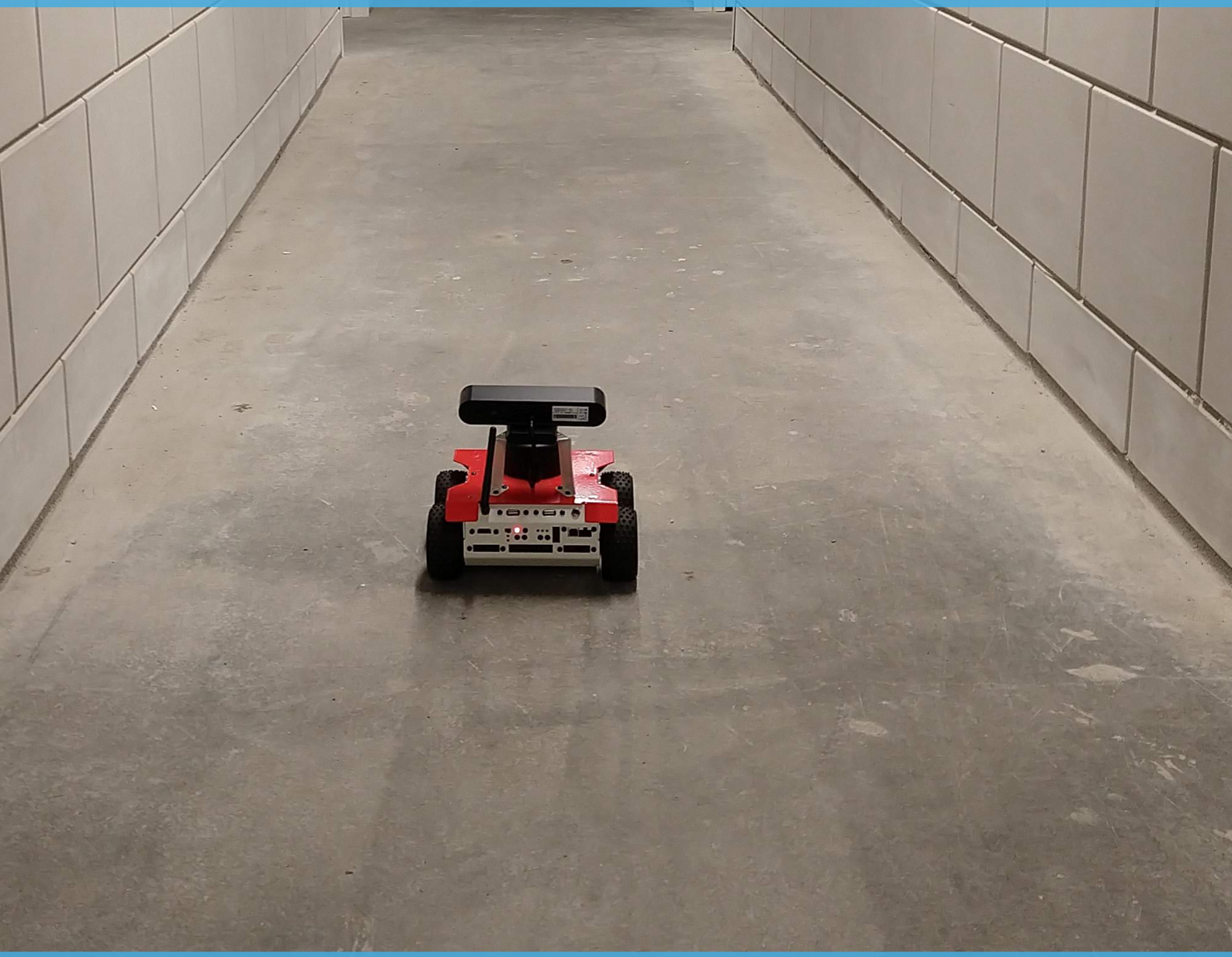
Learning Interactively to Resolve Ambiguity in Sensor Policy Fusion for Robot Navigation

B.G.N. Bootsma



TU Delft

Cognitive Robotics

BOTS

Learning Interactively to Resolve Ambiguity in Sensor Policy Fusion for Robot Navigation

Master of Science Thesis

Author:
Bart Bootsma, 4696883
Technische Universiteit Delft, The Netherlands
Dept: Mechanical Engineering
b.g.n.bootsma@student.tudelft.nl

Supervisors:

Reka Bérci-Hajnovics
Dobots, The Netherlands

Giovanni Franzese
TU Delft, The Netherlands
Dept. Cognitive Robotics

Dr.-Ing. Jens Kober
TU Delft, The Netherlands
Dept. Cognitive Robotics

Acknowledgements

I am grateful to everyone at the Cognitive Robotics department and the robotics company Dobots for the opportunity to work on the fascinating subject of robot learning. Besides the academic challenge and the practical skills gained, I obtained a realistic view of the possible learning approaches for robotics, which I hope to apply in my future work.

I want to thank Jens Kober for his critical input on the project's scope and focus. His clear comments made it easier to separate the good from the bad. Furthermore, I am incredibly grateful for the support of Giovanni Franzese and his limitless patience to educate me on machine learning topics. The amount of suggested literature almost surpassed my reading speed. Without his extensive feedback and suggestions on both the research and the writing, the project would not have been possible, and I would not have enjoyed it as much.

I would also like to thank my colleagues Reka Bérci-Hajnovics and Snehal Jauhri from Dobots. During every weekly meeting their input and fresh eyes, detailed reviews of the written parts, and guidance during the project have been essential.

Contents

Chapter 1

Thesis Structure

This thesis is the completion of the master Mechanical Engineering at the TU Delft. The subject of the thesis can be defined as a combination of mobile robot navigation and machine learning. The main body of this thesis is written in a paper format. Additional information is provided in the appendices. New readers should start with the paper, and move to the appendices for detailed information.

Chapter 2 contains the paper, which introduces the research goal, related works, presents the learning algorithm, experimental results, and conclusions.

Appendix A provides additional information on the robot setup and the implementation during the thesis.

Appendix B explains further details about the neural networks which have a vital role in the architecture.

Appendix C presents an additional investigation into the parameters of the main learning algorithm.

Appendix D describes an addition to the learning algorithm. This addition is excluded from the final algorithm presented in the paper.

Appendix E gives additional information about the main simulation experiment. It describes the expert training, typical scenario's, a brief analysis of data collected during the training, including trajectories.

Appendix F provides additional information on the real-world experiments described in the paper.

Chapter 2

Paper

Learning Interactively to Resolve Ambiguity in Sensor Policy Fusion for Robot Navigation

Bart Bootsma, Giovanni Franzese, and Jens Kober
Department of Cognitive Robotics
Delft University of Technology, Netherlands

***Abstract*—This work applies interactive imitation learning for the navigation of a mobile robot. The algorithm "Learning Interactively to Resolve Ambiguity in Sensor Policy Fusion" (LIRA-SPF) is introduced in the field of machine learning for robot navigation. This algorithm extends on existing methods by allowing the ambiguity-free fusion of existing single-sensor policy behavior using an active and interactive querying of the human expert. The ambiguous situations investigated in this work are due the possible perspective mismatch of each sensor: LIRA-SPF aims to detect these situations and save the correct solution in a new fused policy. As a consequence, we provide an alternative to training a new behavior again from scratch, leveraging the knowledge of existing expert behaviors and reducing the required teacher's effort. The algorithm is tested with different supervised and unsupervised disambiguation strategies thanks to its modular implementation. This paper summarizes multiple simulated and real robot tests, showing the advantages of the proposed disambiguation module on state of the art approaches. In particular, the analysis underlines the necessity of less human-robot interaction during the training process. Finally the conclusions reveal the missing blocks of the approach and how this could be beneficial in the sensor fusion procedure.**

I. Introduction

The use of robotics in assisting humans and performing several different tasks can change our lives. Unfortunately, for the execution of each new task, a slow reprogramming of the robot is necessary. This approach limits the adaptability of robots to new conditions. Ideally, robots can quickly adapt to unknown situations and can learn directly from humans. What if even a child could teach a robot to drive autonomously? Or if a non-expert user could transfer her skills to the robot via a kinesthetic demonstration? Learning from Demonstration (LfD) methods allow a human to teach robots this way [1].

Within LfD, there are different approaches to the training of a new behavior. For example, Behavioural Cloning (BC) collects all the demonstration data and then trains the robot policy to clone that desired behavior. It can, however, be challenging to foresee which demonstrations lead to the desired behavior [2]. A solution is to have the human in the learning loop and supervise the robot with corrective feedback. Instead of providing all demonstrations at the start, the human is asked to supervise the training and provide corrective feedback during multiple iterations. This is the field of Interactive imitation learning (IIL) methods. Examples like DAgger [3], HG-DAgger [4], TAMER [5], COACH [6], have shown promising results in an easier, safer, and faster way of teaching robots. IIL methods teach the learner in an online fashion, in contrast to the offline BC.

This work focuses on learning navigation behavior for an autonomous mobile robot. Mobile robots often use a combination of sensors to navigate through the environment safely. As each sensor modality has its strengths, they are often responsible for a specific behavior. Combining different sensor modalities is the field of sensor fusion. A survey, by W. Elmenreich [7], describes sensor fusion as "the combining of sensory data or data derived from sensory data such that the resulting information is in some sense better than would be possible when these sources were used individually." What if we can leverage these different sensor modalities to help the human demonstrate the desired behavior? With existing sensor-specific behaviors that acts as experts, our methods assists the human teacher by detecting ambiguous situations and applies the actions of these experts behaviors to reduce the input a human teacher has to give.

We named our method LIRA-SPF (Learning Interactively to Resolve Ambiguity in Sensor Policy Fusion). During their lifetime robots operate in different environments and fulfill different tasks. LIRA-SPF is based on the assumption that during their lifetime different

behaviors are developed. Instead of defining a complete new behavior for each application, we can use these existing behaviors to speed up the implementation process. Examples of the expert behaviors are: 'follow the line', 'avoid obstacles', and 'drive through a narrow opening'. We assume the expert behaviors are a black box and only provide an action for the robot. This makes our approach applicable to a wide range of existing behaviors. Based on their action LIRA-SPF assigns priority to one of the expert behaviors during the training process. It is however likely that the existing behaviors are dependent on different sensors. These different sensor modalities can result in ambiguous situations. The training approach specifically detects and resolves these ambiguities. LIRA-SPF essentially fuses sensor-specific policies into a novice behavior which depends on the full observation.

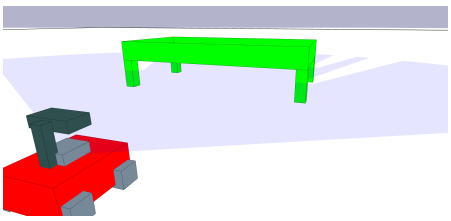


Figure 1: When a LIDAR perceives table legs to the left and right side, the LIDAR-based policy might deem it safe to drive ahead. However, the camera can see the connecting tabletop and decides it is not high enough to drive underneath.

The scenario displayed in Figure 1 is an example of ambiguity the robot might encounter. Often mobile robots use multiple sensors to observe the environment. However, each sensor perceives different features. For example, an RGB monocular camera cannot infer the distance measurements from an obstacle, but a 2D laser scan can. On the other hand, a 2D laser rangefinder cannot detect color and textures, but a camera can. Based on these perception mismatches, ambiguous situations can occur in the policy fusion when the single actions are not coherent: which sensor perceives the feature that matches the teacher intention?

We define an ambiguous situation as the scenario where only one sensor can sense the relevant feature, and the output decision is not consistent among the different sensor policies. We aim to detect those ambiguous situations and actively query a human expert to teach a sensor-fusion novice policy in these situations. The experiments in Section IV and V-A use both a camera and a 2D laser rangefinder which we will refer to as a LIDAR in the remaining text.

Research Question. *Could Interactive Learning be successfully applied to the learning of sensor policy fusion, leveraging the knowledge of the teacher on the task execution for the automatic inference of the different sensor's relevance in ambiguous/conflicting situations? Is learning sensor policy fusion more data-efficient and user-friendly than learning everything again from scratch?*

The main contributions of this paper are:

1) A learning approach that ingrates existing expert policies in a human-robot interactive learning framework.
2) A validation of the learning approach on simulated and real-world navigation tasks.
3) A comparison between LIRA-SPF and a state of the art interactive imitation learning approach.

In Section II related work is discussed. Section III explains the learning algorithm. Section IV focuses on the simulated experiments, followed by the real world validation in Section V. Section VI discusses the method and concludes the findings of this study.

II. Related Works and Background

LIRA-SPF is a variation on the work in [8], which proposed the method named 'Learning Interactively to Resolve Ambiguity in Reference Frame Selection' (LIRA-RFS). LIRA-RFS already uses human-robot interaction to resolve ambiguous situations. The approach presented in this paper touches on different research fields. At its core, LIRA-SPF is an IIL approach, but it is also related to active learning and sensor fusion.

A. Interactive Imitation Learning

While Behavioural Cloning separates the demonstration from the policy execution, Interactive Imitation Learning places the human teacher in a fused learning-execution loop where the feedback directly affects the robot's decisions. The quality of the teacher's feedback determines the performance of the learner. Therefore, recent studies focused on the type of feedback and interaction between human teachers and learners. The DAgger algorithm [9] records the current policy roll out so a human teacher can add feedback for the recorded states. As the control never fully switches to the human, it can be hard to provide the correct feedback and lead the exploration. HG-DAgger [4], therefore, switches completely during training from the novice to the human teacher when he/she wants to take control. Figure 2 is a schematic visualization of HG-DAgger. TAMER [5] does not require feedback in the action space but asks the teacher to assign a reward for

the novice's actions. However, it is not always an obvious choice, making it hard to give the correct feedback. The approach of [10] extends on HG-DAgger. In combination with the explicit feedback, it also leverages the implicit feedback. They show a reduction of demonstrations required from the teacher.

Another approach, COACH [6], uses a human interpretation model to translate the feedback from a human teacher to policy corrections. The human gives feedback in the form of directional corrections. As the method does not need perfect action labels from the human, COACH allows even unskilled teachers to demonstrate complex tasks. LIRA-SPF extends on these existing IIL methods, specifically HG-DAgger, by assisting the human teacher during the training phase. The existing sensor-specific expert behaviors are used to detect and highlight ambiguous situations, so the human can be warned to pay extra attention. In addition, LIRA-SPF benefits from the experts' knowledge and reduces the feedback required from the human teacher.

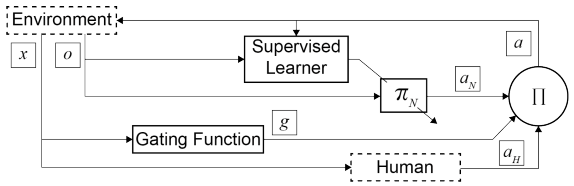


Figure 2: During HG-DAgger, the observation $\mathcal{O}$ is fed to the novice policy π_N, which output is an action a_N. The human often has a complete view of the state x, and chooses when to take control and supply an action a_H. During the training phase of HG-DAgger, the human acts as a gating function by taking control. During the test-phase, a novelty detection based on the uncertainty of π_N determines whether additional a_H is required. The combination of a_H and $\mathcal{O}$ is used as input for a supervised learning algorithm to update the π_N.

B. Active Learning

It is likely that a trained behavior has not seen the complete task environment. New areas where the novice is not confident might lead to dangerous situations. As used in [11]–[14], active learning aims to detect when the novice is not confident about its actions and requires more feedback from the human. Similar to the uncertainty measure used in [13], HG-DAgger uses the variance between a committee of trained networks to detect novel situations in the test phase. Interesting is that the training phase's interaction defines the threshold to determine the confidence of the trained networks. LIRA-SPF explicitly targets conditions where one of the sensors is blind to specific features in the environment. In these situations, the difference between single-sensor expert policies can be used to detect the ambiguity. This decision ambiguity can also result from the intrinsic difference between the expert policies. In both scenarios, all expert policies can be confident but contradicting. In contrast to the active learning methods that apply an epistemic uncertainty metric to determine the novice's confidence in the test-phase, LIRA-SPF uses the ambiguity between expert policies during the training to actively signal the human teacher.

C. Sensor Fusion

LIRA-SPF aims to learn navigation policies efficiently that fuse multiple sensors. Previous LfD studies mostly used a single sensor modality [4], [9], [15]–[20]. There are examples of sensor fusion inside neural networks. The LfD method applied in [21], fuses range and image data inside the neural network. The approach in [22] combines RGB and Depth image into a single feature which acts as input for the function approximation. A comparison in [23] shows that a camera and laser range finder perform better while learning to drive autonomously. The LfD methods in [19], [24]–[26] fuse sensor input with state measurements (e.g., speed, position) and higher-level goal commands in the same network. These methods use the fusion of different data sources to increase the complexity of the learned task.

Sensor fusion can also actively increase the performance by leveraging the sensor characteristics. In [27], sensor fusion increases robustness in an end-to-end driving policy. By applying Sensor Dropout during the training process, the policy is less sensitive to one specific modality. In [28], a confidence metric determines whether a second camera is needed to resolve the ambiguity. Similarly, LIRA-SPF actively leverages the differences between the sensor modalities. However, instead of adding an extra sensor, the human teacher is queried for more feedback when an ambiguity is detected.

Some fusion attempts focus on the fusion of separate networks. In [29], two separate networks are available on different sensors. The camera-based network can determine mid-long term actions, while the LIDAR-based networks are better equipped for close range obstacle avoidance. The networks are fused on a decision level with a simple rule: 'when the LIDAR sensor readings are within a minimum range, its network takes over control for a preset number of time steps.' The disadvantage of this approach, aside from the difficulties in scaling to more sensors, is that the camera is disregarded when the LIDAR sensor is active.

The method in [29] uses a fixed parameter to switch between sensor-specific network modalities. While in the MAMMOTH method of [30], the fusion of two sensor-specific networks is trained on a third novice network. The method is compared to a 'Monolithic' architecture. [30] state that MAMMOTH architecture gives the teacher more control over which sensor-modality is essential in different situations. Similarly, LIRA-SPF allows a general sensor fusion approach by learning the fusion of the sensor-specific networks in an additional network. However, in contrast to [30], LIRA-SPF actively calls for the human teacher's attention, which makes it easier to supply the right demonstrations.

III. LIRA-SPF

This section describes the LIRA-SPF training approach, as visualized in Figure 3 and Algorithm 1. End-to-end navigation architectures map sensor data directly to an executable command for the mobile robot. LIRA-SPF trains a new behavior (novice) to map the output of multiple sensors to an action command. The sensor combination will change depending on the task, and instead of learning from scratch, LIRA-SPF leverages existing policies during the training. The only constraint for the expert policies is that they are able to predict continuous action commands for the robot. LIRA-SPF judges the relevance of the expert policies based only on their action output. This feature makes it easier to implement different sensors and experts, as it treats all experts equally without the need for prior knowledge about expert behavior.

A LIRA-SPF training loop requires at least two expert policies Π^1, Π^2. While it is possible to extend to more experts, the algorithm is described based on two experts for the explanation. At the start, an empty novice buffer $\mathcal{D}^N$, and short term memory $\mathcal{S}_j$ are initialized. The short-term memory stores the expert policy actions for the last j steps. Each time step, an observation $\mathcal{O}$ is retrieved from the robot sensors. Each loop, the expert policies predict an action a^1, a^2. Sometimes the human provides feedback in the form of an action a^H. Eventually the action a is executed on the robot and demonstrations labels $d(a, \mathcal{O})$ are added to $\mathcal{D}^N$.

LIRA-SPF starts with the $\mathcal{O}$, see line 3 in Algorithm 1. The actions predicted by Π^1, Π^2 (line 4) are added to the finite $\mathcal{S}_j$ (line 5). The expert policies only use their specific subset of the observation. If the human expert takes control (line 7) and provides an a^H, the demonstrated label is added to the $\mathcal{D}^N$. The human can always decide to take control during the training to prevent unwanted or dangerous situations. If the human expert is not in control, three possible situations may occur.

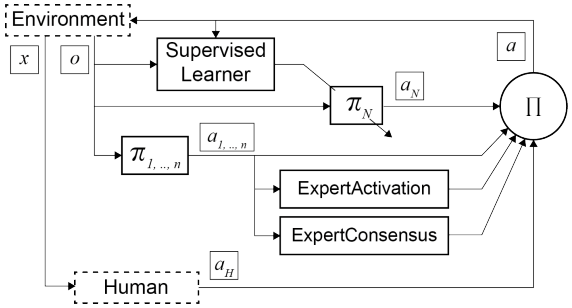


Figure 3: LIRA-SPF control loop starts with the observation $\mathcal{O}$ as input for the novice policy π_N, and expert policies $\pi_{1,..,n}$, which all output an action a_N, and $a_{1,..,n}$, respectively. The human has a complete view of the state x, and chooses when to take control and supply an action a_H. The experts actions $a_{1,..,n}$ is fed into the ExpertActivation function and the ExpertConsensus function. The output these functions determines which policy should have control and if additional human feedback is desired. The supervised learning algorithm updates the π_N with the collected feedback.

Algorithm 1 Basic LIRA-SPF training loop

Require: Π^1, Π^2
1: **Init:** $\mathcal{D}^N = []$, $\mathcal{S}_j = []$
2: **for** $t = 1, 2, ...$ **do**
3: **Get** $\mathcal{O}$
4: a^1, a^2, $a^N = \Pi^1(\mathcal{O})$, $\Pi^2(\mathcal{O})$, $\Pi^N(\mathcal{O})$
5: $\mathcal{S}_j \leftarrow a^1, a^2$
6: **Get** ExpertActivation(S_j)
7: **if** human gives feedback **then**
8: $a = a^H$
9: $\mathcal{D}^N \leftarrow d(a, \mathcal{O})$
10: **else if** Π^1 or Π^2 is activated **then**
11: **Call** for attention
12: $a = a^1$ or a^2
13: $\mathcal{D}^N \leftarrow d(a, \mathcal{O})$
14: **else if** Both Π^1 and Π^2 are activated **then**
15: **Get** ExpertConsensus(a^1, a^2)
16: **if** Experts agree **then**
17: $a = mean(a^1, a^2)$
18: $\mathcal{D}^N \leftarrow d(a, \mathcal{O})$
19: **else**
20: **Call** for feedback
21: $a = a^N$
22: **else**
23: $a = a^N$
24: **Execute** a
25: **Train** Π^N on $\mathcal{D}^N$ **if** m new labels in $\mathcal{D}^N$

In line 6, Π^1 and Π^2 are evaluated on their **temporal** activation. The experts' activation determines whether an expert is relevant in this specific state of the environment. The activation is computed based on their last j actions. If a sensor is effectively blind to features in the environment, the accompanying expert's output will not change. When the predicted action change is large enough, we assume that the accompanying sensor is picking up relevant features in the environment. By defining the activation as a temporal change in action, we can distinguish where the expert behavior is blind and where it isn't.

If a single expert is activated, it receives priority as we assume the other expert is blind in this situation (line 10). However, a call for attention is sent to the human (line 11) to notify that one expert is now in control, and feedback might be necessary. The notification is important because this makes it easier for the human to determine when extra focus is required.

If more than one expert is activated, the similarity is computed with the ExpertConsensus function (line 15). Both expert policies may be activated on different features in the environment and agree on the action. When the experts predict the 'same' action (line 16), their action is set as a and added to $\mathcal{D}^N$. The similarity between Π^1, Π^2 is computed based on the difference of their action. The other option is that both Π^1, and Π^2 are activated and disagree about the action. In that case, we assume there is an ambiguous situation that requires human feedback. During the experiments, the speed of the robot is reduced in case of an ambiguous situation. The reduced speed allows the human to observe the current novice policy and take over control when needed. In all other cases, the novice action is executed (line 23).

A. ExpertActivation

The ExpertActivation function receives $\mathcal{S}^i$ as input, which is a matrix with the last J actions a_k for expert policy i. The output is a measure of activation for Π^i. The activation is measured by summing the standard deviation for the most recent actions of the expert policy. The actions have a dimension k, depending on the task. The activation for expert i is

$$\sum_k \sqrt{\frac{\sum_{j=1}^{J}(\mathcal{S}^i_{j_k} - \bar{\mathcal{S}^i_k})^2}{J}}. \tag{1}$$

The output of the ExpertActivation function is thus a measure of the temporal change for each expert policy.

B. ExpertConsensus

The ExpertConsensus function receives a_t^1 and a_t^2 as input. The result is true if

$$|a_t^1 - a_t^2| < \mathcal{T}. \tag{2}$$

The threshold $\mathcal{T}$ defines the boundary of the consensus area. A large $\mathcal{T}$ allows big differences between the expert's actions. This difference can lead to destructive behavior and imperfect labels for the novice buffer. If the threshold is set small, ambiguous situation are declared more often except for identical behaviours. Empirically we found that setting $\mathcal{T}$ to 10% of the action dimension is a good starting point.

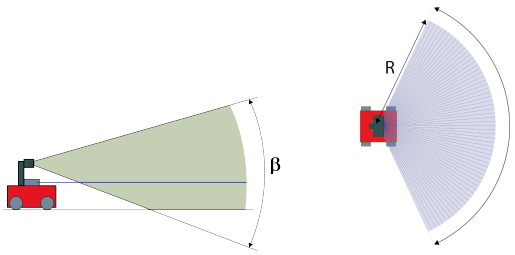
C. Redundant Demonstrations

Every time the experts are activated, they supply an action for the novice buffer. It is, however, possible the novice is already capable in this situation from a previous encounter. To make sure redundant demonstrations are not stored to the novice's buffer, the action supplied by the experts is compared with the novice's action. After line 8, 12, and 17 in LIRA-SPF, an additional comparison is made between the novice action a^N and the current action a supplied by one of the experts. The comparison is similar to the ExpertConsensus function, only with the input being a^N and a. If they are similar the label is not added to the demonstration buffer. We name the threshold $\mathcal{L}$, and assume the novice is already capable in that state if the difference is lower then $\mathcal{L}$.

IV. Simulated World Validations

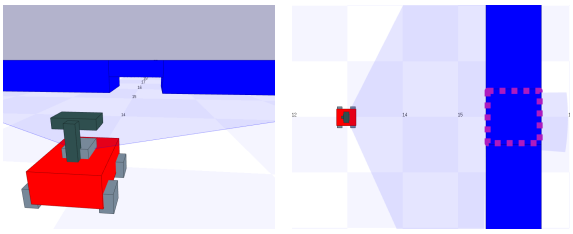
The player stage environment [31] is used to simulate the world. The Robot Operating System (ROS) and Tensorflow (TF) library are used for the implementation. The robot is equipped with a camera and a LIDAR, as visualized in Figure 4. A 32x32 pixel RGBa image is the output from the camera. As color differences will not influence the applications, the RGBa is converted to a grayscale image. The 2D LIDAR, placed at a heigth of 26cm, produces 101 distance measurements, evenly spaced in a forward-facing 130°angle. This angle is similar to the camera field of view, which allows them to see the same obstacles.

There are two expert policies, one for each sensor. The expert policies use a simple end-to-end architecture to predict a two-dimensional action. The camera-based expert (CamExpert) uses convolution layers to extract the features from the image input. While the LIDAR-based expert (LidExpert) only consists of fully connected



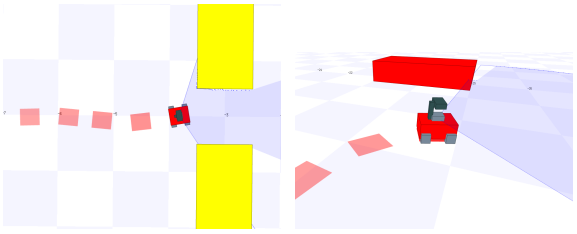
(a) Side-view simulated robot. (b) Top-view simulated robot.

Figure 4: Both the camera and the LIDAR look in a 130° horizontal angle α. With a range R of 4m. While the LIDAR is 2D, the camera looks in a 40° vertical angle β.



(a) Third person simulated robot. (b) Top view.

Figure 5: In the 'narrow passage' scenario, the LIDAR is loo low to perceive it as an obstacle; instead, it sees a narrow opening. The camera, however, is capable of seeing the whole obstacle. The narrow opening is indicated with dashed purple line.



(a) Top view of the narrow corridor. (b) Obstacle situation.

Figure 6: The the expert behaviors are trained to pass through narrow corridors and void obstacles.

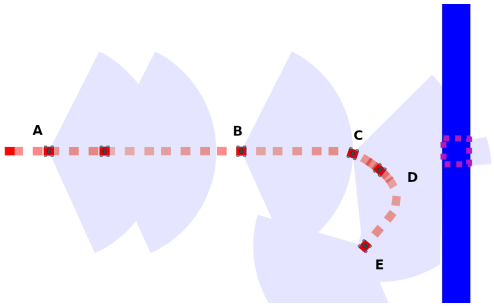


Figure 7: The training trajectory is visualized in a top-view. The distance between the red footprint squares indicates the speed of the robot. In position D, it is clearly visible that the robot drives with reduced speed. The narrow opening is indicated with dashed purple line.

layers (FC). The expert networks are trained interactively for each task with HG-DAgger.

The sensor data is normalized before it is fed into the networks. The expert policies and novice predict a linear and angular velocity, ranging between -1.0 and 1.0. A simulated on-board controller converts these motor commands. The environment contains a ground plane where obstacles are added for the specific tasks.

A. Driving Through a Narrow Passage

The goal of this scenario is to validate that LIRA-SPF can detect and learn to resolve an ambiguous situation. The first ambiguous situation, is visualized in Figure 5. The LIDAR is placed too low to perceive the obstacle and sees a suitable opening. On the other hand, the camera does discern the whole dimension of the opening and can judge the feasibility of going through it. Both expert behaviors are trained to avoid obstacles and pass through narrow corridors, as visualized in Figure 6. If no obstacles or corridors are present, the experts are trained to drive straight ahead. In Table I the demonstration labels and policies updates are summarized.

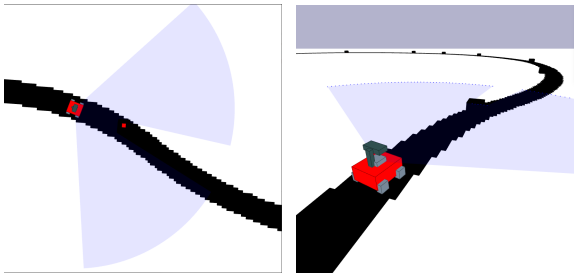
The resulting trajectory, during the experiment, is displayed in Figure 7. The robot starts in position A, with initial input from the human to drive straight. In position B, the robot is able to drive straight after a first policy update. In position C, the LidExpert picks up the wall opening and wants to steer to the left. The camera, however, wants to avoid the wall and steers to the right. As both experts are activated and disagree, the speed is reduced, and the human is called for feedback. The human expert gives a clockwise steering action as feedback in position D. After the feedback, the novice drives straight towards position E. The novice can reproduce this behavior after a policy update. The reduced speed and detection of the ambiguous situation make it easy for humans to give correct demonstrations.

B. Learning Line-Following with Collision Avoidance

The goal of this scenario is two-fold. In the previous example, the novice only learned a dominant camera situation, while in this experiment, both a camera and LIDAR have to be operative. Furthermore, the test aims to prove that it is possible to use LIRA-SPF to detect and

Policy	Expert Labels	Training Step
LidExpert	210	7
CamExpert	270	9
Novice	30	1

Table I: The buffer size and policy updates during the 'narrow passage' scenario. A single training step consist of 50 policy updates with the collected data set. A mini-batch gradient descent approach, randomly selects 32 labels from the complete set for a single policy update.



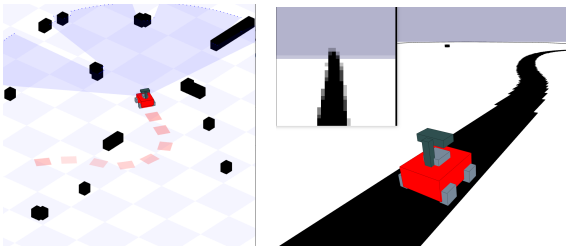
(a) Top-view. (b) Third-person perspective.

Figure 8: The top-view and perspective orientation of the robot while it encounters an obstacle along the black line. A red dot is placed in the obstacle to make it easier for the human to recognize. From the camera perspective, the obstacle blends with the road below. The LIDAR does see the obstacle, yet it cannot see the black line.

solve several ambiguous situations. The performance is compared to the training-phase approach of HG-DAgger, see Figure 2.

The robot is placed in an environment with a black line. However, black obstacles are placed along the line, as visualized in Figure 8. The environment's design is such that the camera will not see the difference between the black line and the obstacles. The LIDAR is only capable of seeing the obstacles. The CamExpert is trained to follow a black line, as seen in Figure 9b. The LidExpert is trained in an obstacle forest to drive straight or steer away from the obstacle (Figure 9a). This experiment aims to teach the novice to avoid the obstacles while following a black line. The assumption is that the human is queried when the robot encounters an obstacle on the black line.

Results displayed in Table II are the averages from ten repetitions. The novice is trained until it avoids all obstacles and does not lose the black line. If the final policies are placed in a situation, which they did not encounter during training, they will likely fail. While the HG-DAgger algorithm learns the desired behavior with a smaller buffer size, the number of demonstrations provided by the human is almost three times the amount of demonstrations during LIRA-SPF. The number of



(a) The Forest environment. (b) The Black-line environment.

Figure 9: Both the LidExpert and the Camexpert are trained with HG-DAgger to get the desired behavior. The CamExpert is also trained to steer back to the black line actively.

Training method	Buffer size		Human labels		Human interventions	
	1^{st}	final	1^{st}	final	1^{st}	final
LIRA-SPF	158	273.5	50	59	7	9.5
HG-DAgger	167	228	96.5	156.5	11	19.5

Table II: The results are averaged over ten repetitions for both LIRA-SPF and HG-DAgger. The activity threshold is set to 3%, and the redundancy threshold $\mathcal{L}$ is set to 5% for LIRA-SPF. The numbers for the first lap and after the final policy is learned are displayed.

human interventions is double for HG-DAgger. This means that during HG-DAgger, the human provides 1.5 times longer trajectories during a take over compared to LIRA-SPF. This means during LIRA-SPF, the human avoids the obstacle but trusts the experts to steer the robot back to the black line as the danger has already passed.

C. Obstacle Race: LIRA-SPF vs HG-DAgger

In the 'obstacle race' scenario, a 120m long black line runs straight to the finish, with 21 black obstacles placed randomly along the path as seen in Figure 10. Similar to the scene in Section IV-B, the black obstacles are hard to distinguish from the black line based on the camera image. However, the 'obstacle race' environment has a higher obstacle density, with obstacles on and besides the black line. The CamExpert and LidExpert used in Section IV-B can also apply this adjusted scenario. This experiment compares LIRA-SPF to the training-phase approach of HG-DAgger, see Figure 2. The training continues until the novice reaches the finish, without any additional feedback. The desired novice policy has learned to avoid obstacles and closely follow the black line while driving as fast as possible.

The results, as summarized in Figure 11 and 12, show that LIRA-SPF requires less interventions and labels from the human teacher. During the training process of LIRA-SPF, the CamExpert and the LidExpert supply

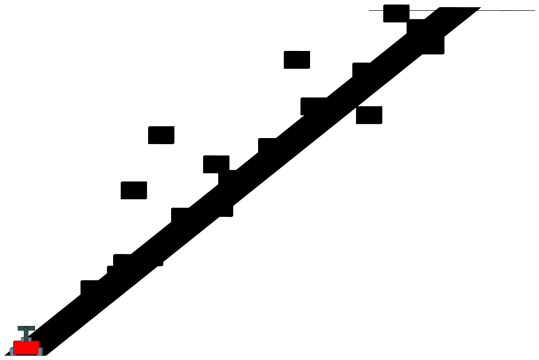


Figure 10: While the robot follows the 120m long black line towards the finish, it encounters 21 black obstacles. The robot needs both the camera and the LIDAR to reach the finish safely. The finish line is visible in the top right corner of the figure.

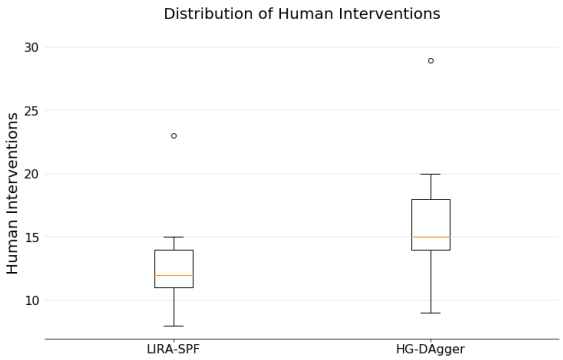


Figure 11: During the training process, the human teacher has to intervene and give feedback. The experiments are repeated ten times for both LIRA-SPF and HG-Dagger. To get a policy capable of reaching the finish line on its own, LIRA-SPF requires fewer interventions than HG-DAgger.

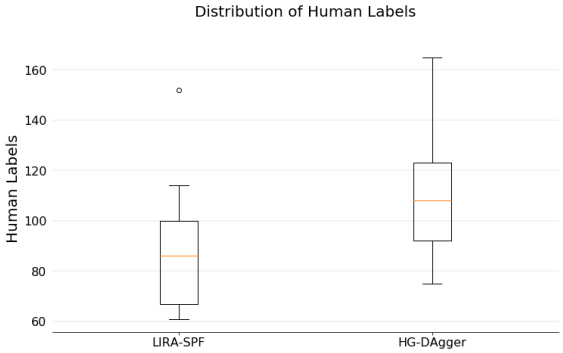


Figure 12: There is a large variance in the number of human labels required for a capable policy between repeated experiments. The number of human-provided labels can be twice as large between repeated experiments for LIRA-SPF and HG-DAgger. Overall, LIRA-SPF requires less input from the human teacher compared to HG-DAgger.

useful corrections, reducing the input the human has to give. An unforeseen ambiguous situation occurred because the novice sometimes actively wanted to steer away from the black line as it saw the black obstacles as pieces of a black line. Often, the camera's influence would be larger when features in the environment are further away, compared to the LIDAR influence. This influence difference can be explained by the fact that as soon as an obstacle enters the field of view, the pixels immediately change from neutral to black, while the LIDAR only experiences a small change from the maximum range to a little less. The large variance and outliers visible in Figures 11 and 12 are the result of imperfect demonstrations: during the experiment the human teacher made errors in the feedback, and moved the robot in a less desirable direction. The correction of the mistakes required extra demonstrations from the human teacher, which resulted in the large variance between experiments. This happened during both LIRA-SPF and HG-DAgger.

V. Real-World Validation

Similar to the simulated setup, ROS and Tensorflow are used to implement the method. The ROSbot connects with a laptop through wifi. The laptop receives sensor data from the ROSbot and sends back velocity commands. The images are transformed to grayscale, and the resolution is reduced to a 32x32 pixel size. The LIDAR measurements are reduced to a maximum of 1 meter and clipped from a full 360°to a 130 °forward-looking angle. With a resolution of one degree per measurement, the resulting array contains 130 measurements. In Figure 13 the ROSbot is displayed. The networks for the CamExpert and the Lidexpert is very similar to the simulation. The convolution layers in the CamExpert use 32 filters instead of 16. Input layer of the LidExpert receives 130 measurements instead of 101. Linear and angular velocity commands are sent to the ROSbot range between -1.0 and 1.0. The onboard controller translates them to wheel speeds—the maximum linear velocity set to 0.20 m/s, and the maximum angular velocity to 210 deg/s.

A. Invisible Obstacle

To validate if the LIRA-SPF method applies to a real-world environment, the ROSbot is placed in a tiled environment displayed in Figure 14. The CamExpert is trained to follow the steel edge border along the blue

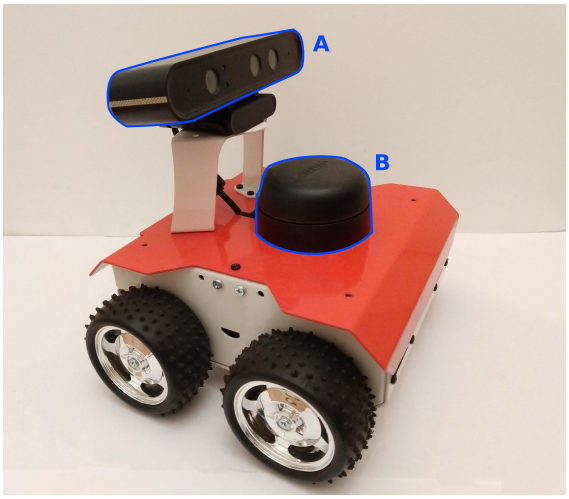


Figure 13: The ROSbot 2.0 is equipped with multiple sensors. In this work, the Orbbec Astra RGB-D camera (A) and the RP LIDAR A2 (B) are used.

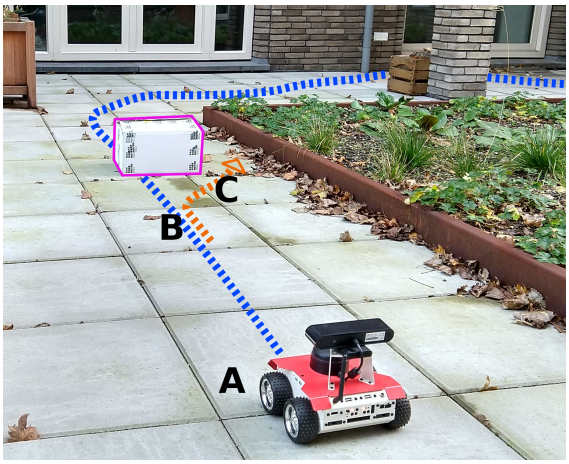
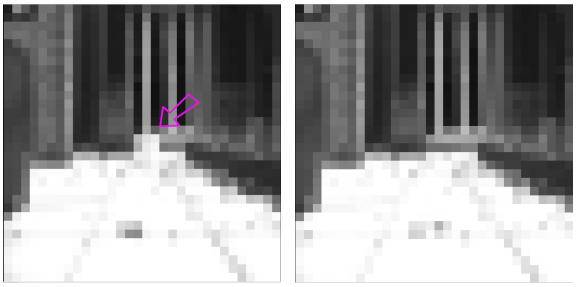


Figure 14: The ROSbot placed in a tiled garden environment. The blue path displays the desired path of the CamExpert. The orange path is the desired path of the LidExpert. The White box with the purple edge is hardly visible for the CamExpert. However, it is easily detected by the LidExpert.

path (without the obstacle). The LidExpert is trained in presence of the obstacle and learns to avoid the obstacle by steering clockwise.

For the training of the sensor fused policy with LIRA-SPF, the white box is placed again along the expected trajectory. As seen in Figure 15, the white box blends with the tiles in the camera image, which make it difficult for the CamExpert to notice the danger.

The resulting trajectory follows the blue path in Figure 14 between position A and B. Between positions



(a) ROSbot camera view with a white box.

(b) Rosbot camera view without a white box.

Figure 15: The purple arrow point to the white box placement. The white box is difficult to recognize for the CamExpert.

A and B, the CamExpert teaches the novice to drive on the blue path. From position B the LidExpert takes control for five steps along the orange path. In position C, the human is queried for feedback. The novice follows the path up to position C after two repetitions of this training cycle. It should be noted that the white box blends with the tiles due to the specific lightning during that time of the day and the greatly reduced resolution of the camera image.

B. Obstacle Corridor

This test the ROSbot has to drive through a corridor while avoiding obstacles. The position of the obstacles is displayed in Figure 16. The floor is filled with white areas that look similar to the real foam obstacles. Figure 17a shows an example setting of obstacles and the accompanying camera in Figure 17b. The CamExpert is trained to drive in the center of the corridor towards the end. During the training, the real obstacles are not present. The LidExpert is trained to avoid obstacles in an open space. During the LIRA-SPF training loop, the CamExpert will try to steer the ROSbot back to the center of the corridor, while the LidExpert detects the real obstacles.

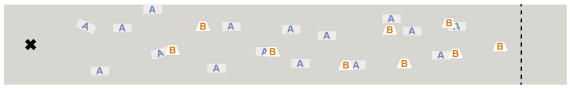
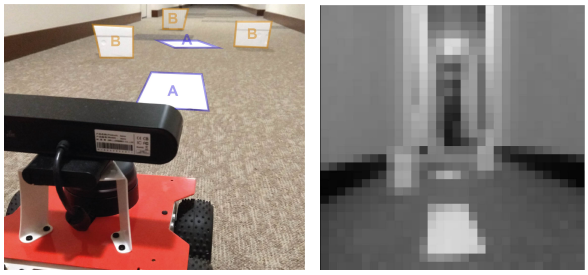


Figure 16: The ROSbot has to move through an obstacle-filled corridor. The A-type obstacles are fake and really white spots on the ground; the B-type are real obstacles in the form of white foam blocks. The camera cannot distinguish the two types, while the LIDAR detects correctly only the B-type. The ROSbot starts at the black cross on the left side and finishes behind the black dashed line on the right side.

The results displayed in Table III are averaged over three repetition of the experiment. After a single training



(a) Third person view. (b) ROSbot camera view.

Figure 17: The two types of obstacles are seen from the ROSbot perspective. The obstacles look indistinguishable in the camera image.

run through the obstacle corridor, the trained novice had varying results for both LIRA-SPF and HG-DAgger The trained novices had an average of 1 collision for both methods. The number of demonstration collected with LIRA-SPF is almost double that with HG-DAgger. However, only roughly a third is given by the human teacher. This difference is also visible in the number of human interventions, which is more than double for HG-DAgger.

Training method	**Buffer size**	**Human labels**	**Human interventions**
	1^{st} lap	1^{st} lap	1^{st} lap
LIRA-SPF	161.7	52.7	6
HG-DAgger	93.7	9.7	13.3

Table III: The buffer size and human input were collected after a single training run through the obstacle corridor. The results are averaged over three repetitions.

VI. Discussion

LIRA-SPF is evaluated in different scenarios. It can detect ambiguities between single-sensor expert policies and learn to resolve them by training a multi-sensor novice. The expert policies are always predicting actions; the ExpertActivation function aims to detect when their actions are relevant. Currently, the function sums the standard deviation for each action dimension. The current threshold to determine whether an expert is active would have to scale with a change in the action dimension. The expert policies used during the experiments are trained for specific purposes. It is, however, possible to use already existing behaviors or programs to act as an expert. This allows a modular approach for the same robot, where a different composition can quickly be used to learn a desired behavior. When more than one expert is active, an error threshold is used to determine whether a situation is **ambiguous**. Based on the properties of the action, this threshold can easily be estimated. However, if the thresholds are not well set, and the human does rely too much on the method, it can result in dangerous situations.

We see two significant improvements for our approach. The first improvement focuses on the application of the expert policies. The current expert policies are assumed to be sufficiently trained for the task environment. During the experiments, the expert were strategically trained to perform well in the task environment. This approach is, however, not always feasible in a practical implementation of LIRA-SPF. If the expert policies can provide a confidence metric about their prediction, LIRA-SPF can choose to ignore uncertain predictions or even signal the human teacher that the robot has entered a novel state for the experts.

The second improvement involves the novice policy. Each time the robot encounters a previously seen ambiguous situation, the human teacher is notified, and the speed is reduced. To prevent the human from giving the same feedback again, the human teacher can choose to trust the novice. In future work, we like to automate this to reduce the human teacher's input further. There are two main solution directions we want to explore. The simplest adjustment would be to increase the required expert activation during the training. This way, the expert policies are quickly activated at the start of the training but lose importance later as the novice likely learned a capable behavior. However, a more general approach would be to compute the novice's confidence during the training. When the robot encounters an ambiguous situation, LIRA-SPF can choose to trust the novice based on its confidence. Because the current LIRA-SPF implementation requires only a reasonable small amount of demonstration labels, gaussian processes are a possible candidate. Another option is the approach of [32] that computes the confidence based on neural network ensembles.

During the experiments, the human used a Ps3 controller to teleoperate the robot. Taking control when the human expert is queried for feedback can be difficult. Giving feedback to go straight forward is easy, yet a steering action becomes somewhat more challenging. The human expert will likely under- or oversteer with this controller. Another aspect is the communication with the human. Currently, a computer display is used to signal the human. This can be improved by using haptic or sound signals. These signals will reduce the strain on the human even further.

We encountered that the imperfect demonstrations

have a significant influence on the speed of the learning process. Flawed demonstrations are a known problem in LfD [33], and remain an open question in our work. Because the experiments used the same human teacher, this teacher can be regarded as skilled. Further investigation should also involve unskilled teachers to compare the influence of LIRA-SPF better.

Conclusion

This study shows that a combination of single sensor-based experts can help teach a multi-modal learner a more complex task by detecting ambiguous novel situations and providing demonstrations. We showed that it is possible to reduce the human strain while teaching a mobile robot navigation behavior. The approach was evaluated in simulation and validated in real-world experiments.

REFERENCES

[1] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, 2009. [Online]. Available: http://dx.doi.org/10.1016/j.robot.2008.10.024

[2] D. Silver, J. A. Bagnell, and A. Stentz, "High performance outdoor navigation from overhead data using imitation learning," *Robotics: Science and Systems*, vol. 4, pp. 262–269, 2009.

[3] S. Ross, G. J. Gordon, and J. A. Bagnell, "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, vol. 15, 2011, pp. 627–635. [Online]. Available: http://proceedings.mlr.press/v15/ross11a/ross11a.pdf

[4] M. Kelly, C. Sidrane, K. Driggs-Campbell, and M. J. Kochenderfer, "HG-DAgger: Interactive imitation learning with human experts," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 8077–8083. [Online]. Available: https://arxiv.org/abs/1810.02890

[5] W. B. Knox and P. Stone, "TAMER: Training an Agent Manually via Evaluative Reinforcement," in *2008 7th IEEE International Conference on Development and Learning*, Monterey, CA, 2008, pp. 292–297. [Online]. Available: https://ieeexplore.ieee.org/document/4640845

[6] C. Celemin and J. Ruiz-del Solar, "An Interactive Framework for Learning Continuous Actions Policies Based on Corrective Feedback," *Journal of Intelligent & Robotic Systems*, vol. 95, pp. 77–97, 2019. [Online]. Available: https://doi.org/10.1007/s10846-018-0839-z

[7] W. Elmenreich, "An introduction to sensor fusion," *Austria: Vienna University Of Technology*, no. February, pp. 1–28, 2002. [Online]. Available: http://www.vmars.tuwien.ac.at/documents/intern/805/elmenreich{_}sensorfusionintro.pdf

[8] G. Franzese, C. Celemin, and J. Kober, "Learning Interactively to Resolve Ambiguity in Reference Frame Selection," no. CoRL, 2020.

[9] S. Ross, N. Melik-Barkhudarov, K. shaurya Shankar, A. Wendel, D. Dey, j. andrew Bagnell, and M. Hebert, "Learning monocular reactive UAV control in cluttered natural environments," in *2013 IEEE International Conference on Robotics and Automation*. Karlsruhe: IEEE, 2013, pp. 1765–1772. [Online]. Available: https://ipvs.informatik.uni-stuttgart.de/mlr/13-SeminarRobotics/presentations/LearningMonocularReactiveUAVControl.pdf

[10] J. Spencer, S. Choudhury, M. Barnes, M. Schmittle, M. Chiang, P. Ramadge, and S. Srinivasa, "Learning from Interventions : Human-robot interaction as both explicit and implicit feedback," *Robotics: Science and Systems (RSS)*, 2020.

[11] D. H. Grollman and O. C. Jenkins, "Dogged learning for robots," *Proceedings - IEEE International Conference on Robotics and Automation*, no. April, pp. 2483–2488, 2007.

[12] S. Chernova and M. Veloso, "Confidence-Based Policy Learning from Demonstration Using Gaussian Mixture Models," vol. 5, pp. 1315–1322, 2007.

[13] D. Silver, J. A. Bagnell, and A. Stentz, "Active learning from demonstration for robust autonomous navigation," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 200–207, 2012.

[14] M. Laskey, S. Staszak, W. Y. S. Hsieh, J. Mahler, F. T. Pokorny, A. D. Dragan, and K. Goldberg, "SHIV: Reducing supervisor burden in DAgger using support vectors for efficient learning from demonstrations in high dimensional state spaces," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2016-June, pp. 462–469, 2016.

[15] A. Giusti, J. Guzzi, D. C. Ciresan, F. L. He, J. P. Rodriguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. D. Caro, D. Scaramuzza, and L. M. Gambardella, "A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 661–667, 2016. [Online]. Available: https://ieeexplore.ieee.org/document/7358076

[16] K. Ofjall, M. Felsberg, and A. Robinson, "Visual autonomous road following by symbiotic online learning," in *IEEE Intelligent Vehicles Symposium, Proceedings*, no. Iv. Gothenburg, Sweden: IEEE, 2016, pp. 136–143. [Online]. Available: https://ieeexplore.ieee.org/document/7535377

[17] M. Bojarski, D. Del 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. [Online]. Available: https://arxiv.org/abs/1604.07316

[18] L. Tai, S. Li, and M. Liu, "A deep-network solution towards model-less obstacle avoidance," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, no. October. Daejeon, South Korea: IEEE, 2016, pp. 2759–2764. [Online]. Available: https://doi.org/10.1109/IROS.2016.7759428

[19] Y. Pan, C. A. Cheng, K. Saigol, K. Lee, X. Yan, E. A. Theodorou, and B. Boots, "Imitation learning for agile autonomous driving," *The International Journal of Robotics Research*, vol. 39, no. 2-3, pp. 286–302, 2019. [Online]. Available: https://doi.org/10.1177/0278364919880273

[20] S. Hornauer, K. Zipser, and S. X. Yu, "Learning to Roam Free from Small-Space Autonomous Driving with A Path Planner," pp. 1–14, 2017. [Online]. Available: http://arxiv.org/abs/1709.10512

[21] D. a. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," *Advances in Neural Information Processing Systems 1*, pp. 305–313, 1989. [Online]. Available: http://papers.nips.cc/paper/95-alvinn-an-autonomous-land-vehicle-in-a-neural-network.pdf

[22] Z. Huang, C. Lv, Y. Xing, and J. Wu, "Multi-modal Sensor Fusion-Based Deep Neural Network for End-to-end Autonomous Driving with Scene Understanding," *IEEE Sensors Journal*, no. May, pp. 1–1, 2020.

[23] I. Kang, R. Cimurs, J. H. Lee, and I. Hong Suh, "Fusion Drive: End-to-End Multi Modal Sensor Fusion for Guided Low-Cost Autonomous Vehicle," pp. 421–428, 2020.

[24] M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart, and 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). IEEE, 2017, pp. 1527–1533. [Online]. Available: https://ieeexplore.ieee.org/document/7989182

[25] M. Pfeiffer, S. Shukla, M. Turchetta, C. Cadena, A. Krause, R. Siegwart, and J. Nieto, "Reinforced Imitation: Sample Efficient Deep Reinforcement Learning for Mapless Navigation by Leveraging Prior Demonstrations," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4423–4430, 2018. [Online]. Available: https://doi.org/10.1109/LRA.2018.2869644

[26] F. Codevilla, M. Miiller, A. Lopez, V. Koltun, and A. Dosovitskiy, "End-to-End Driving Via Conditional Imitation Learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. Brisbane, QLD: IEEE, 2018, pp. 4693–4700. [Online]. Available: https://ieeexplore.ieee.org/document/8460487

[27] G.-H. Liu, A. Siravuru, S. Prabhakar, M. Veloso, and G. Kantor, "Learning End-to-end Multimodal Sensor Policies for Autonomous Navigation," no. CoRL, pp. 1–13, 2017. [Online]. Available: http://arxiv.org/abs/1705.10422

[28] A. S. Pourya Hoseini, M. Nicolescu, and M. Nicolescu, "Handling Ambiguous Object Recognition Situations in a Robotic Environment via Dynamic Information Fusion," *Proceedings - 2018 IEEE International Conference on Cognitive and Computational Aspects of Situation Management, CogSIMA 2018*, pp. 56–62, 2018.

[29] X. Zhou, Y. Gao, and L. Guan, "Towards goal-directed navigation through combining learning based global and local planners," *Sensors (Switzerland)*, vol. 19, no. 1, 2019.

[30] I. L. Davis and A. Stentz, "Sensor fusion for autonomous outdoor navigation using neural networks," *IEEE International Conference on Intelligent Robots and Systems*, vol. 3, pp. 338–343, 1995.

[31] "stage_ros - ROS Wiki." [Online]. Available: https://wiki.ros.org/stage{_}ros

[32] K. Menda, K. Driggs-Campbell, and M. J. Kochenderfer, "EnsembleDAgger: A Bayesian Approach to Safe Imitation Learning," *IEEE International Conference on Intelligent Robots and Systems*, no. 2, pp. 5041–5048, 2019.

[33] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent Advances in Robot Learning from Demonstration," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. 1, pp. 297–330, 2020.

Appendix A

Robot Setup

This Appendix provides a brief description of the implementation during the thesis. The robotic operating system (ROS) framework is used to implement the different modules. ROS supports a modular approach, making it possible to easily switch between a simulated and a physical robot setup. The experiments are executed on a notebook with an Intel i5-7200U CPU (2.50GHz), 16 GB ram, and integrated HD Graphics 620. Figure A.1 provides an overview of the main modules in the architecture. The simulated experiments are fully executed on the notebook, while during the real-world experiments, the notebook communicates with the ROSbot over a WiFi protocol. The information gathered by the robot sensors is placed on ROS topics. The training loop runs with 5Hz. The speed is limited by the transportation of the images from the ROSbot to the notebook. The current speed of the training loop is sufficient for the low velocities during the experiments. The human teacher controls the linear and angular velocity through a joystick gamepad.

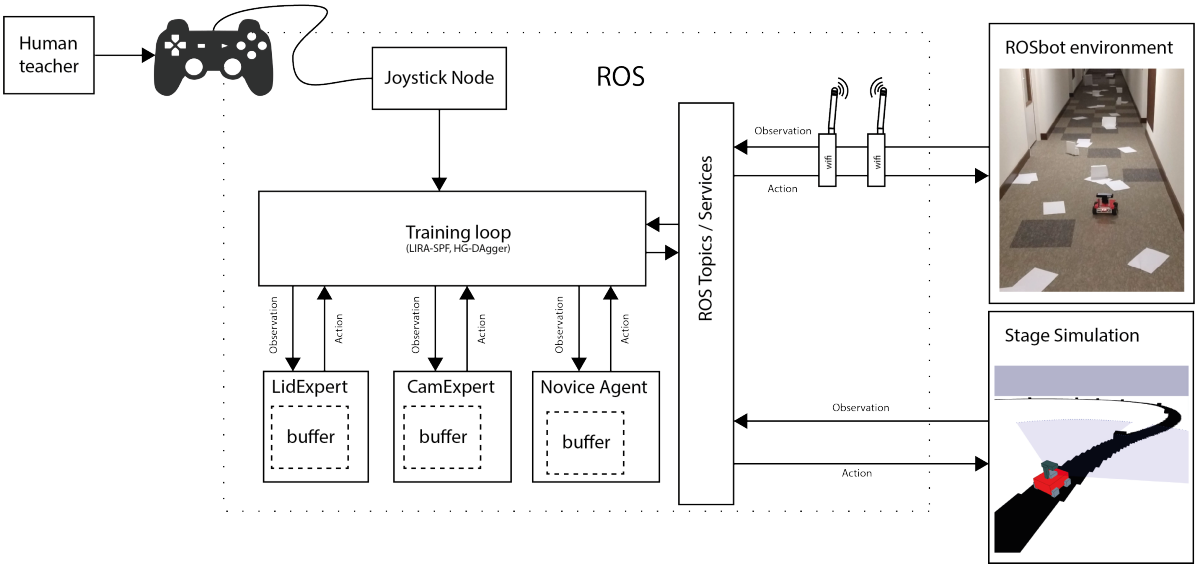


Figure A.1: The same architecture is used for the simulated and real-world experiments. The sensor data is posted on ROS topics and used by the main training loop to update the agents. The action from the agents or human teacher is then used to control the robots.

Appendix B

End-to-End Networks

LIRA-SPF uses an end-to-end architecture for the application to mobile robot navigation. End-to-end architectures map raw sensor data to useful commands for a robot [1]. Figure B.1 displays the approach used during the experiments described in the paper. This section describes the neural networks that facilitates end-to-end learning based on different sensors. The networks are implemented with the Tensorflow 2.0 API, which provides an easy to use interface. This appendix is divided into three sections. Section B.1 describes the network used by the CamExpert, and Section B.2 explains the network used by the LidExpert. The novices' network uses both the camera and LIDAR information. Section B.3 describes a multimodal network that fuses the sensor information. All networks are trained with an experience replay buffer similar to the buffer used in [2]. The replay buffer allows efficient use of the collected demonstrations. The Adam optimizer updates the neural networks, in mini-batches of 32 samples, with a mean squared error loss function to calculate the losses, and a learning rate set to 1e-3. The dimensions of the networks are the result of fine-tuning during preliminary tests. The networks are sufficient for the current application. It is, however, likely that the networks can be further optimized.

Figure B.1: The end-to-end architecture directly maps the output from the sensors to action commands for the robot. The image data is reduced in resolution and channels to keep the networks small and easily trainable.

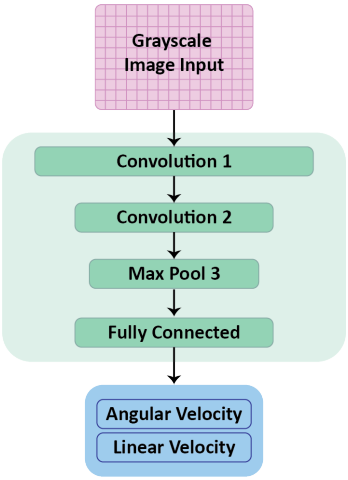


Figure B.2: Overview of the dedicated image network.

Layer	Properties	Activation
Input	Size 32x32x1	
CCN 1	16 or 32 Filters, 3x3 kernel	Leaky ReLU
CNN 2	16 or 32 Filters, 3x3 kernel	Leaky ReLU
MaxPool	2x2 kernel	
FC	Output 2x1	Hyperbolic Tangent

Table B.1: The most important properties of the image network are listed for each layer.

B.1 Camera Network

The camera-based network expects a single channel 32x32 pixel image. The network properties are summarized in Table B.1 and visualized in Figure B.2. Similar to the work of [3], convolution layers are expected to extract the relevant features from the image input. The network was first tuned for the simulated environment and then adjusted for the real-world tests. The simulated tests use convolution layers with 16 filters, while for the real-world tests, the convolution layers were extended to 32 filters. As real-world environments have a higher complexity than the simulated environment, the higher filter level was expected to capture a higher abstraction level. A preliminary test showed that the adjusted setting allowed the network to learn simple 'corridor following' behaviors faster. A fully connected layer maps the convolution output to a two-dimensional action after a pooling operation to reduce the output dimension. The hyperbolic tangent activation function is used for the output layer to ensure the output values stay between -1 and 1.

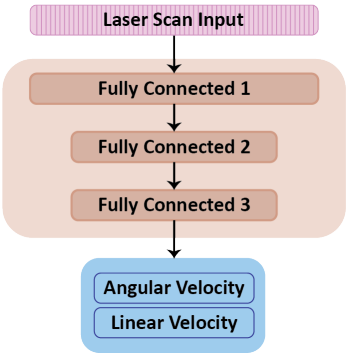


Figure B.3: Overview of the dedicated laser scan network.

Layer	Properties	Activation
Input	101x1 or 130x1	
FC 1	Ouput 64x1	Leaky ReLU
FC 2	Ouput 64x1	Leaky ReLU
FC 3	Output 2x1	Hyperbolic Tangent

Table B.2: The most important properties of the laser scan network are listed for each layer. The input size varies between 101 during the simulated experiments and 130 during the real-world experiments.

B.2 LIDAR Network

The output of the laser scanner is an array with range measurements. The array's size differs between the simulated and real-world robot, with 101 values for the simulation and 130 for the ROSbot. The network consists of three fully connected layers in sequence. The hidden layers have a size of 64 neurons. The hyperbolic tangent activation function is used for the output layer to ensure the output values stay between -1 and 1.

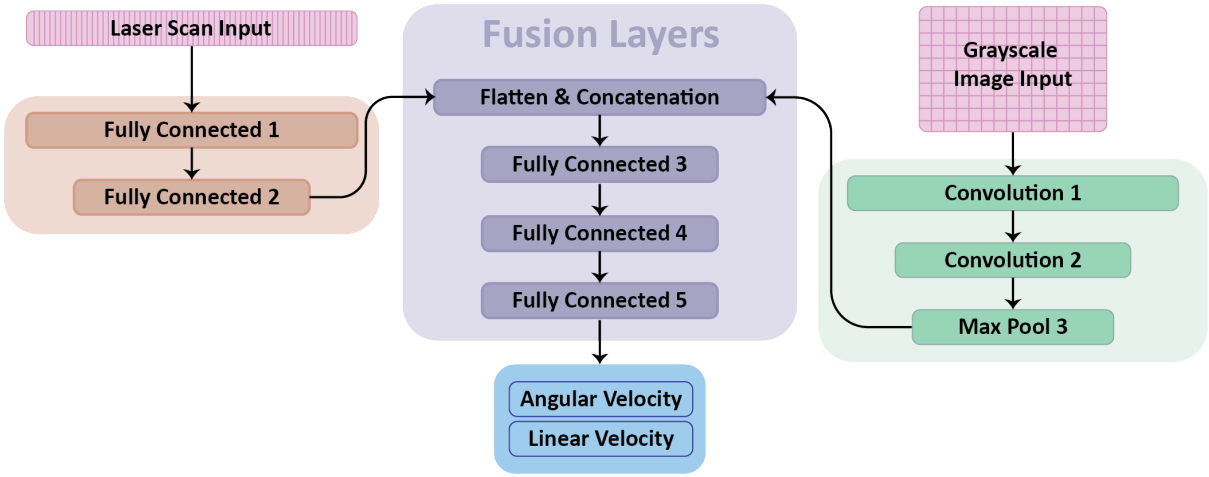


Figure B.4: The first layers of the fusion network are similar to the single-sensor networks. The sensor information is combined in 3 fully connected fusion layers to predict a linear and angular velocity.

Function	Layer	Properties	Activation
Feature Extraction LIDAR	Input	101x1 or 130x1	
	FC 1	Ouput 64x1	Leaky ReLU
	FC 2	Ouput 64x1	Leaky ReLU
Feature Extraction Camera	Input	32x32x1	
	CNN 1	16 or 32 Filters, 3x3 Kernel	Leaky ReLU
	CNN 2	16 or 32 Filters, 3x3 Kernel	Leaky ReLU
	MaxPool	2x2 Kernel	
Sensor Fusion	FC 3	Output 128 x1	Leaky ReLU
	FC 4	Output 128x1	Leaky ReLU
	FC 5	Output 2x1	Hyperbolic Tangent

Table B.3: The properties of the fusion network are displayed for each layer. The network is divided into two separate functions: feature extraction for the camera and LIDAR, and sensor fusion in final layers.

B.3 Fusion Network

The fusion network combines two separate single-sensor branches into a fusion network as visualized in Figure B.4. The layer-specific properties are described in Table B.3. The output of the sensor-specific layers is flattened and concatenated in a single tensor. The final fully connected layers map the sensor information to a two-dimensional action. The most straightforward way to train the fusion network is from scratch with the collected demonstration labels. An advantage of LIRA-SPF, however, is that the expert networks are already available. Those networks are already trained to process camera and LIDAR data. The first layers of those networks can be

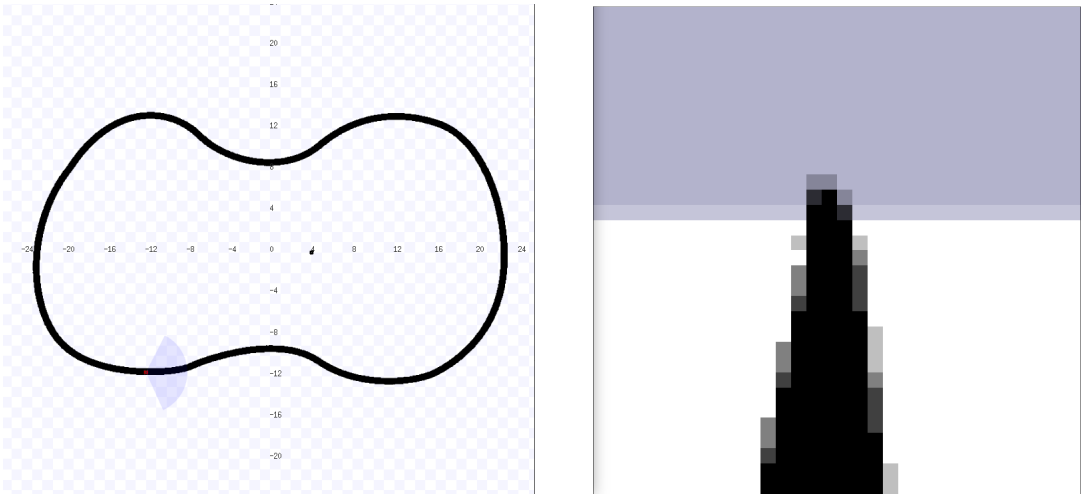
used to reduce the size of the fusion network. In [4] a similar architecture is used. The first layers of the expert networks are designed to extract features from raw sensor data. The output of the feature extraction layers is then used as input for a novice network. During LIRA-SPF only the final fusion layers have to be updated. The simulated test showed that the reduced fusion network performed similarly to a complete new fusion network. However, a full comparison of the methods is beyond the scope of this research.

Appendix C

LIRA-SPF Parameter Tuning

Several thresholds need to be set in the LIRA-SPF framework. This section describes a preliminary investigation to estimate the influence of those thresholds. The investigation evaluates settings for the ExpertActivation function and the additional 'redundant demonstrations' module (Chapter 2). The same experiment is repeated with different settings to evaluate the behavior of the algorithm. The settings are then compared based on the number of demonstration collected and how well the novice learned the desired task. The robot is placed in an environment with a black line. The aim is that the novice learns to follow the line with maximum speed. The camera does see the black line, while the LIDAR does not. The track is visualized in Figure C.1a, and the perception of the camera in Figure C.1b. The LidExpert is trained to drive straight unless it sees an obstacle. The CamExpert learned to steer towards and follow the line with maximum speed.

Since the CamExpert is the only one that will be activated, LidExpert is ignored. During the experiment, the human can step in if the robot does not show the desired behavior. First, the settings for the ExpertActivation function are changed with the redundant demonstration module off. Second, the experiment is repeated with relevant settings for the activation function and the redundant demonstration module. During all settings, the policy is automatically updated after 30 new demonstration labels.



(a) Top-view of the track.

(b) Camera view of the robot.

Figure C.1: The simulated 'follow the line' environment consist of a single black line. The track has a eight figure shape with bends in different directions.

#	Activation threshold	Buffer size		Policy updates	
		1^{st} lap	2^{nd} lap	1^{st} lap	2^{nd} lap
1	1%	570	998	18	33
2	3%	77	124	2	4
3	6%	30	30	1	1
4	10%	30	30	1	1

Table C.1: The activation function computes the standard deviation over the last five actions of the LidExpert and CamExpert. A fixed threshold is used to determine when the experts are active. The threshold is set between 1% - 10% of the action range. The amount of demonstration labels and novice policy updates are counted after the 1^{st} and 2^{nd} lap.

Results of first setting are displayed in Table C.1. As seen in Table C.1 the number of demonstration labels stored reduces with a higher activation threshold. With the activation threshold set to 1% and 3%, the novice learns the desired policy. With the activation threshold set to 6%, the novice learns to follow the black line. However, it does follow slightly off-center at times. The difference is that the lower thresholds allow smaller adjustments from the CamExpert while following the black line. When the activation threshold is set to 10%, the novice roughly learns to follow the black line but often drives besides the line. The speed is also half the maximum speed. To get the correct behavior, the human has to add 50 extra labels with five interventions. The CamExpert is quickly activated with an activation threshold of 1%. This results in many similar demonstration labels.

#	Activation threshold	$\mathcal{L}$	Buffer size		Policy updates	
			1^{st}	2^{nd}	1^{st}	2^{nd}
1	1%	5%	83	99	3	3
2	1%	3%	167	228	5	7
3	3%	5%	52	61	1	2
4	3%	3%	54	73	1	2

Table C.2: The activation threshold is changed from 1% to 3% of the output dimension. The redundant demonstration module compares each expert action to the current novice policy's action, with the absolute difference. The error threshold $\mathcal{L}$ is set to 3% and 5%. The amount of demonstration labels and novice policy updates are counted after the 1^{st} and 2^{nd} lap.

Results with the additional redundant demonstration module are displayed in Table C.2. The activation threshold is set to 1% and 3%, as the higher thresholds already stop accumulating demonstrations. The redundant demonstration threshold $\mathcal{L}$ is set to 3% and 5%. The use of this additional comparison is most beneficial for the lowest activation threshold (1%), but also, with the activation threshold set to 3%, the module reduces the number of demonstration labels. Already after two laps, the difference becomes significant. The performance of the learned novice policy is not distinguishable from the policies learned without the module. During the experiments, the human did not have to take control. This is due to the placement and task the novice had to learn. After the first 30 labels, the novice policy is updated. With this experiment, the first 30 labels are sufficient to get a basic line following behavior. During the experiment, the novice policy will not deviate very far from the black line.

Appendix D

Intention Reading in Human Feedback

This Appendix describes an additional investigation made during the research. This addition aims to further reduce the required input from the human teacher. Section D.1 describes the approach, and Section D.2 applies the approach to a simulated test.

D.1 Approach

In LIRA-SPF, the human is called for feedback. The assumption is that the human will often give feedback similar to one of the expert policies. Therefore, the expert policies can provide feedback, which reduces the number of demonstration labels the human has to give. The additional comparison is added in Algorithm 1. When the human is in control, the human's actions, expert policy actions, and novices actions are stored in a short-term buffer $\mathcal{R}$ (line 5). If the human gives back control, the FeedbackSimilarity function is called. The FeedbackSimilarity assigns priority based on the lowest mean squared error between the human and the expert or novice predictions. The result determines whether a single expert or novice gains priority for u steps. The human can still take back control if the priority results in undesirable behavior.

Algorithm 1 LIRA-SPF with Shared Feedback

Require: Π^1, Π^2
1: **Init:** $\mathcal{D}^N = []$, $\mathcal{S}_j = []$, $\mathcal{R} = []$
2: **for** $t = 1, 2, ...$ **do**
3: ...
4: **if** human gives feedback **then**
5: $\mathcal{R}_n \leftarrow a^1, a^2, a^N$
6: ...
7: **else if** at t-1 human was in control **then**
8: **Get** FeedbackSimilarity($\mathcal{R}_h$)
9: **if** feedback is similar to a single expert **then**
10: follow similar expert policy for u steps
11: **end if**
12: **else**
13: ...
14: **end if**
15: ...
16: **end for**

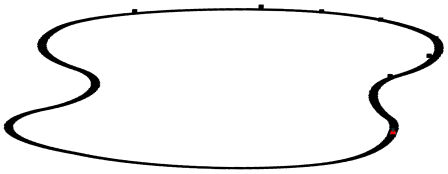


Figure D.1: The test is executed on the figure-eight track. The track is populated with black obstacles, which the robot has to avoid.

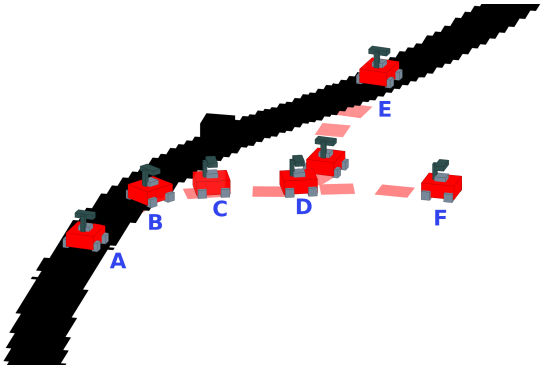


Figure D.2: The robot notices the obstacle in position A. The LidExpert's preferred trajectory runs from position A-F. When the robot encounters the obstacle, the human teacher is called to resolve the ambiguity. This results in positions B, C, D. In position D, the human gives back control. The preferred position is now E. However, based on the low error with the human feedback, the LidExpert receives priority and moves the robot to position F.

D.2 Validation

This test aimed to validate whether it is possible to share the feedback burden between a human and an expert policy. The setting is similar to the 'Learning Line-Following with Collision Avoidance' experiment, as visualized in Figure D.1. The same trained LidExpert and CamExpert are used as in the Learning 'Line-Following with Collision Avoidance' experiment. The LidExpert avoids obstacles, and the CamExpert steers towards the black line. A fixed and flexible setting are compared. In the fixed setting, the expert takes control from the human after two steps. The flexible setting waits for the human to resign control.

Setting	Buffer size	Policy updates	Human labels	Human interventions
Fixed	296	9	50	25
Flexible	236	8	92	16

Table D.1: The shared feedback module is tested with a fixed and flexible setting. During the fixed setting, the experts take over after two feedback actions from the human, while in the flexible version, the human decides when to resign control. The numbers are averaged over ten repeated experiments.

The results are summarized in Table D.1. In the fixed setting, the expert takes over control after two steps. However, often the feedback was not similar enough to one of the experts. Therefore the number of human interventions is much higher for the fixed setting, as the teacher had to retake control. The amount of labels provided by the human is less for the fixed setting because it is hard to determine when an expert policy can take over. In the case of the flexible setting, this results in an extended length of the demonstration. The main problem is visualized in Figure D.2. The task during the experiment requires a quick switch between the LidExpert and the CamExpert. Because the expert receives priority based on the history of the human feedback, this switch is missed. The human teacher can take over if the wrong expert receives priority. However, this results in confusing situations and imperfect demonstrations. Additionally, the priority to the wrong expert will also result in imperfect demonstration labels for the novice.

Appendix E

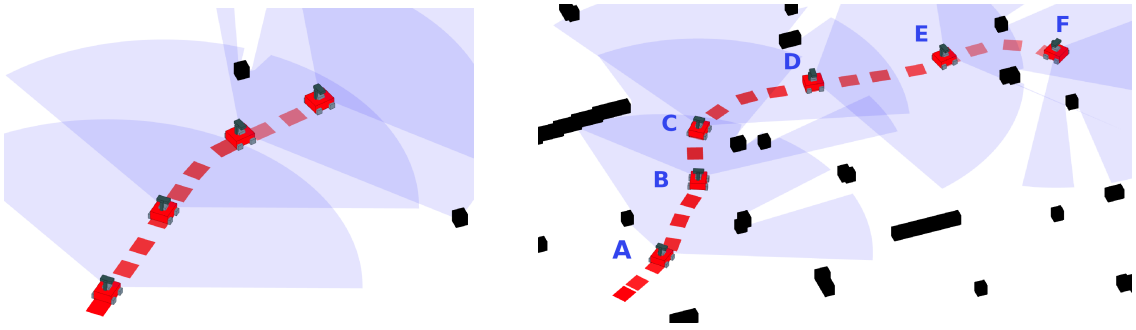
Additional Information Obstacle Race Experiment

This appendix provides additional information about the 'obstacle race: LIRA-SPF vs. HG-DAgger' experiment, described in the paper. After a summary of the experiment, Section E.1 gives further details about the training of the expert policies used by LIRA-SPF. Section E.2 aims to explain the ambiguous situation as it occurs during the experiment. The buffers collected with both HG-Dagger and LIRA-SPF are compared in Section E.3. Section E.4 contains the trajectories of the final policies during the experiment.

During the simulated 'obstacle race' experiment, the novice is trained to follow a black line while avoiding obstacles. The results presented in the paper show that LIRA-SPF requires fewer interventions and demonstrations from a human teacher compared to HG-DAgger. The novice is trained until it can reach the finish line without additional feedback.

E.1 Expert Policies

The expert policies used in this experiment are referred to as the CamExpert for the camera-based policy and LidExpert for the LIDAR based policy. Both experts are trained in their own environment. The LidExpert and CamExpert learn to predict a linear and angular velocity.

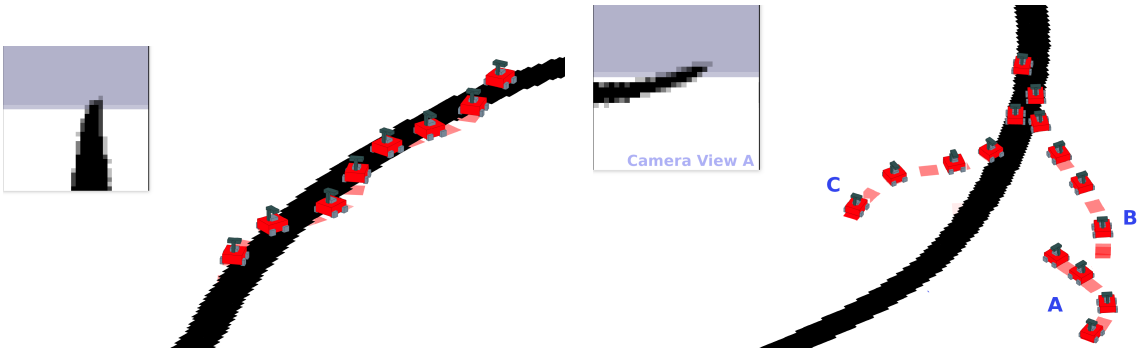


(a) Sparse obstacle area.

(b) Dense obstacle area.

Figure E.1: The training starts in a sparse area of the environment. The robot receives the initial demonstration as driving straight and steer away from the obstacle. After the initial demonstration, the robot wanders into more densely populated areas. Occasionally it receives feedback from the human teacher. The human teacher is instructed to steer the robot away from collision paths and obstacles closest to the robot. It can, however, be challenging to judge the right direction in the dense obstacle area. Based on the instruction, the move from position C-D is wrong.

The LidExpert is trained to avoid obstacles in the 'forest' environment, which contains randomly placed obstacles. Figure E.1 provides a summary of the scenarios the robot encountered during the training of the LidExpert. The LidExpert learns to predict both the linear as angular velocity. The training starts by driving straight until the robot encounters an obstacle, see Figure E.1a. After 30 new labels, the policy is updated. During the training, a human teacher is instructed to drive straight until an obstacle enters the LIDAR range. Based on the position of the obstacle, the teacher steers the robot away from the obstacle. The training starts in a position with a single block to give an initial demonstration Figure E.1a. After an initial policy update, the robot will wander into areas of the environment that are denser populated with obstacles. Along the trajectory seen in Figure E.1b, the robot receives feedback from the human to help it navigate the area safely.



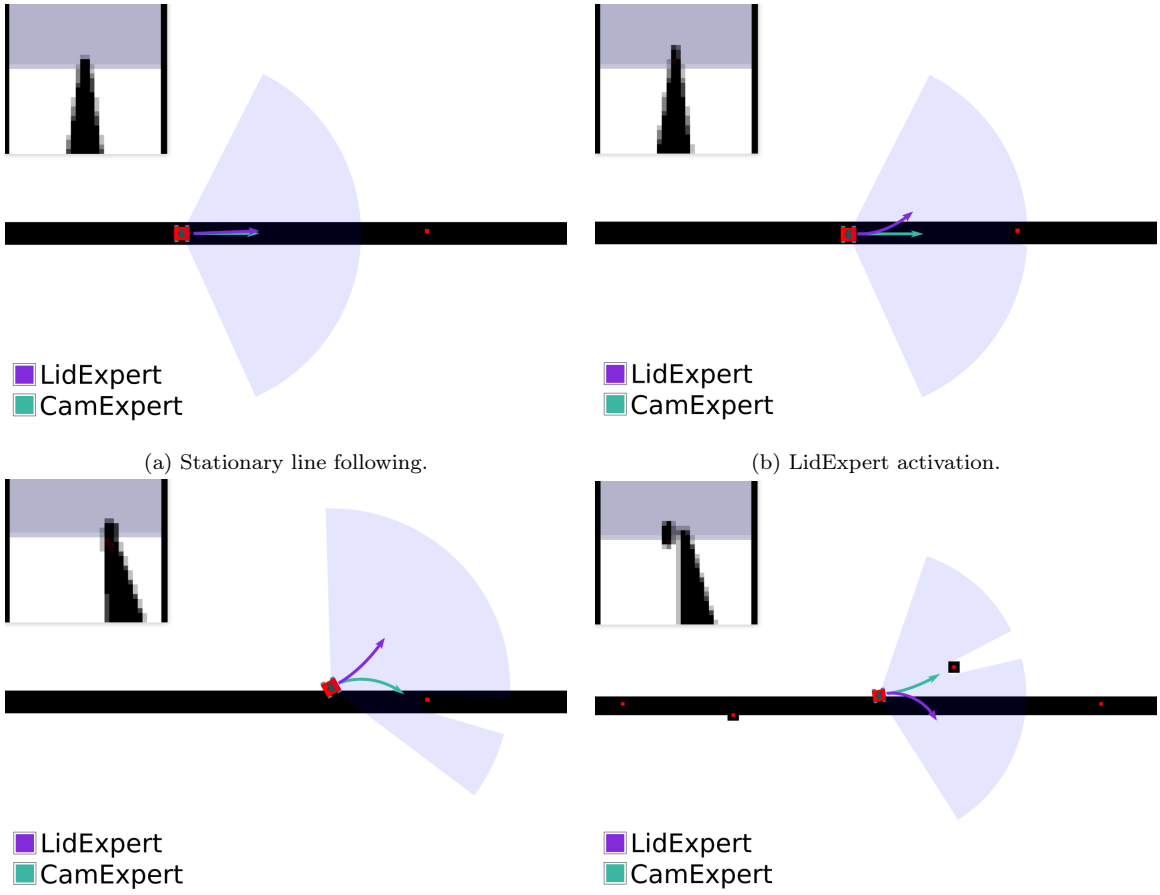
(a) Initial demonstrations. (b) Manual scenarios.

Figure E.2: The robot starts the training on the black, initially it has no clue and will therefore receive feedback to stay on the black line. During the bends in different directions, the robot requires more feedback. During the 'obstacle race' experiment, the robot also ventures besides the black line. Therefore the robot is placed in positions beside the black line to teach the CamExpert a recovery behavior.

The CamExpert is trained to follow the black line. The robot starts on the black line, and receives feedback when it steers away from the line, see Figure E.2a. When the robot capable of following the curved line from start to end, the robot is also placed in position beside the line. This teaches the robot to steer back to the line when it sees the line from the corner of the camera view, see Figure E.2b.

E.2 Ambiguous Encounters

During the sensors policy fusion with LIRA-SPF, the robot encounters ambiguous situations. In this section, a brief overview is given of the situations encountered in the obstacle race experiment. In Figure E.3 several steps are visualized of an ambiguous situation. Initially, the obstacle is not visible, Figure E.3a. When the obstacle comes into the LIDAR range, the LidExpert wants to avoid the obstacle, see Figure E.3b. From the camera perspective, nothing changed; hence its action stays the same. The LidExpert receives priority and steers the robot away from the line to avoid the obstacle. However, the CamExepert notices the black line is now moving away from its center and starts steering to the right Figure E.3c. Both experts are activated, and thus the LIRA-SPF calls for the human teacher's attention to resolve the situation. An additional ambiguous situation occurred because the CamExpert was not trained in an environment with obstacles besides the line. In FigureE.3d the robot encounters an obstacle close to the line. In this situation, the CamExpert wants to steer somewhere in the middle between the obstacle and the line.



(a) Stationary line following.

(b) LidExpert activation.

(c) CamExpert activation.

(d) Wrong CamExpert activation.

Figure E.3: The Ambiguous situation occurs when the LidExpert steers the robot away from the black line to avoid the obstacle, and the CamExpert is activated as it wants to steer back to the black line. An unforeseen ambiguity occurs when the CamExpert is wrongly activated as it wants to steer towards an obstacle, and the LidExpert steers to avoid the obstacle.

E.3 Buffer Intensity

This section gives an insight into demonstration labels collected during the 'obstacle race' experiment. During LIRA-SPF, both the human teachers and the experts can add demonstration labels to the novice's buffer. Figure E.4 displays two density plots that visualize the linear and angular velocity action labels collected with LIRA-SPF. Most action labels contain the maximum linear velocity. The human and experts account almost equally for the number of demonstrations. The second plot shows, however, a large difference between the human teacher and the experts. The human teacher either drives straight or applies maximum steering velocity, while the experts give much more mid-range angular velocities.

Figure E.5 displays the difference between the actions in the buffer collected by HG-Dagger and LIRA-SPF. The action labels are very similar for both methods. Only some small differences are noticeable. The buffer collected by HG-DAgger also contains demonstrations of driving backward. This indicates that during HG-DAgger, the human teacher rescued the robot from positions too close to the obstacles. The angular velocity is almost similar. However, throughout ten repetitions, HG-DAgger required more demonstrations from the human, hence the height difference.

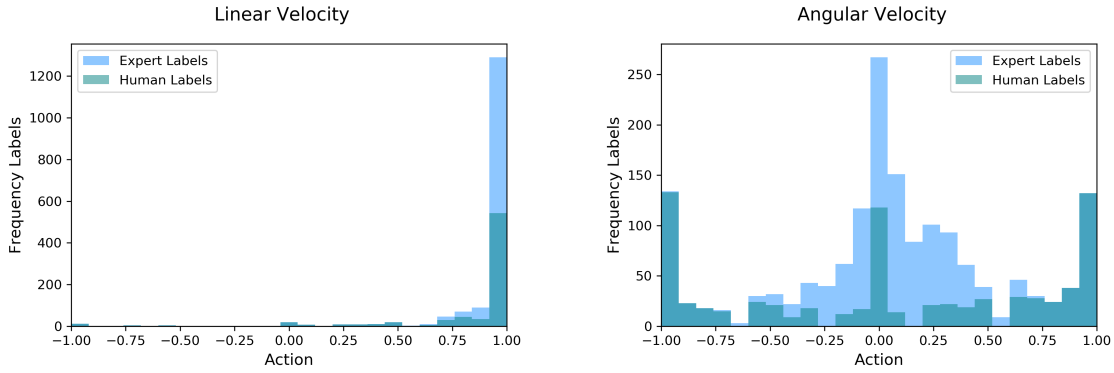


Figure E.4: There is a clear difference between the frequency of the mid-range angular velocities provided by the experts and the human teacher.

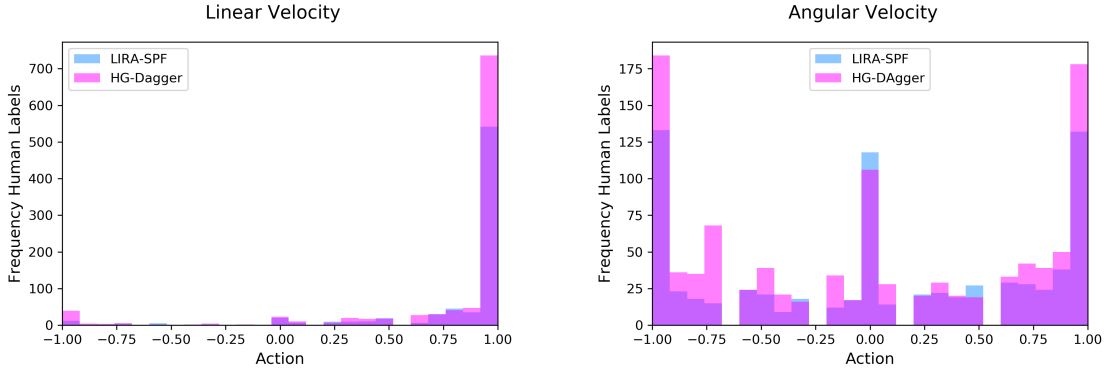


Figure E.5: The collected action labels are very similar between LIRA-SPF and HG-DAgger.

E.4 Trajectory Plots

The final trajectories from the novice policies are visualized in Figure E.6. The full 120-meter long trajectory is divided into three equal parts for an accurate visualization.

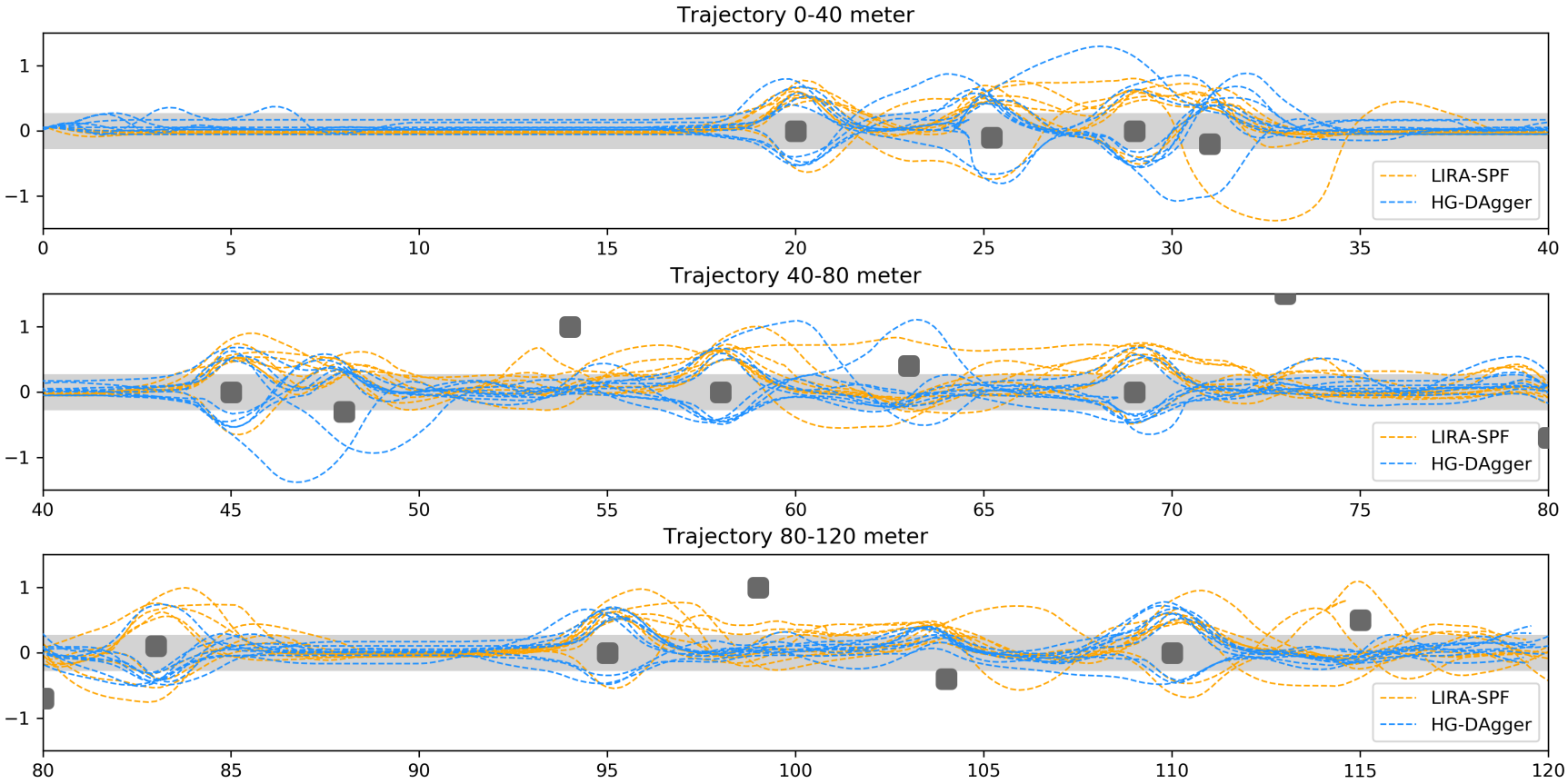


Figure E.6: The final trajectories from both the LIRA-SPF and the HG-DAgger approach are visualized.

Appendix F

Additional Information Obstacle Corridor Experiment

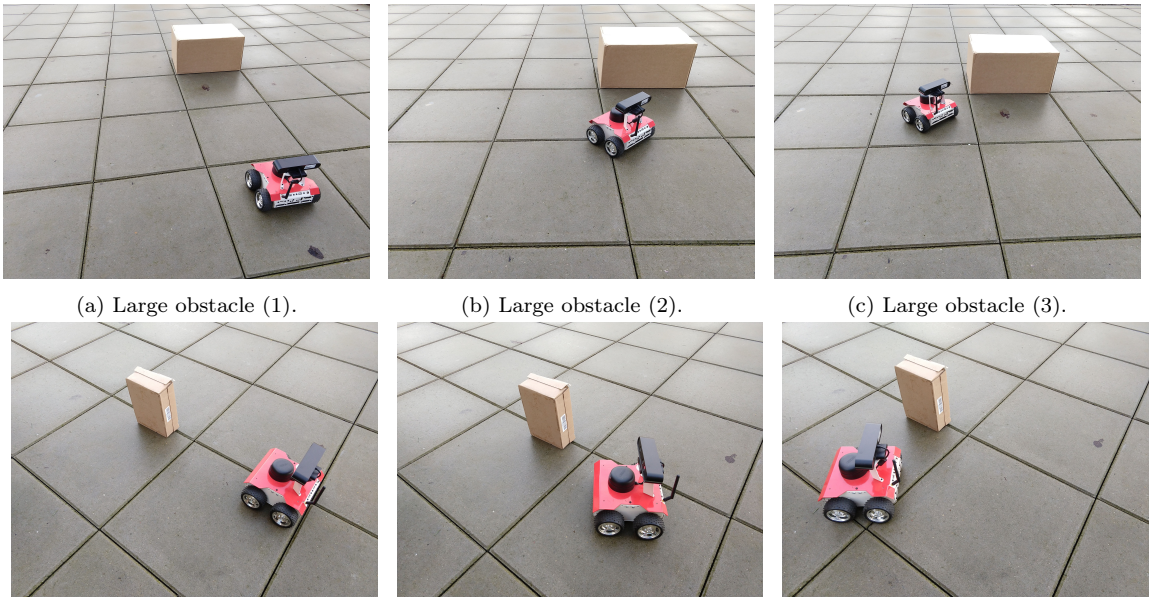
This appendix provides additional information about the real world 'obstacle corridor' experiment. After a summary of the experiment, Section F.1 explains the used experts' training. Section F.2 gives a short description of the ambiguity during the experiment. The 'obstacle corridor' experiment aims to train a novice to drive along a corridor. However, the corridor is populated with real and fake obstacles. The real obstacles are foam blocks, and the fake obstacles are just white spots on the floor, which look similar to foam blocks from the camera's point of view. Based on both the camera and the LIDAR, the novice is trained to avoid the obstacles and drive towards the end of the hallway.

F.1 Expert Training

For this experiment, both the LidExpert and the CamExpert were trained to predict a robot's linear and angular velocity command. This section describes how the expert are trained.

The LidExpert was already available from the 'invisible obstacle' experiment and could be used again for this experiment. Figure F.1 shows a collection of steps in the training process. A human teacher placed the robot in various situations before an obstacle and provided feedback for the robot to safely avoid the obstacles. Obstacles of different sizes were used to generalize the LidExpert for multiple situations.

The CamExpert is trained in the same corridor as the final experiment. Only fake obstacles are placed on the floor during the training. Figure F.2a shows the training environment of the Camexpert. During the 'obstacle corridor' experiment the actual obstacles are added, see Figure F.2b. The robot is reset to the same initial position until the CamExpert can drive to the end of the hallway while staying in the center.



(a) Large obstacle (1). (b) Large obstacle (2). (c) Large obstacle (3).

(d) Narrow obstacle (1). (e) Narrow obstacle (2). (f) Narrow obstacle (3).

Figure F.1: The LidExpert is manually placed in different positions before different obstacles.

(a) CamExpert training environment. (b) Obstacle corridor environment.

Figure F.2: The CamExpert is trained to drive down the same corridor as the 'obstacle corridor' experiment. The training environment is vacant of 'real' obstacles and only contains 'fake' obstacles.

F.2 Ambiguous Situation

This section describes the ambiguous situation as encountered by the robot during the 'obstacle corridor' experiment. Figure F.3 visualizes two positions of the robot in the environment and the accompanying actions predicted by the experts. When the robot encounters the first obstacle, the LidExpert is activated and steers the robot away from the obstacle. The CamExpert is now also activated as the robots moves away from the center of the corridor. LIRA-SPF now calls the human teacher to resolve the situation with additional demonstrations. Between obstacles, the CamExpert also assists with demonstrations to steer the robot to the corridor's center. These demonstrations reduce the number given by the human compared to HG-Dagger.

Figure F.3: The robot encounters the first obstacle. The actions predicted by booth the LidExpert and CamExpert are visualized.

Bibliography

[1] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent Advances in Robot Learning from Demonstration," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. 1, pp. 297–330, 2020.

[2] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 2016. [Online]. Available: https://arxiv.org/abs/1509.02971

[3] Y. LeCun, U. Muller, J. Ben, E. Cosatto, and B. Flepp, "Off-road obstacle avoidance through end-to-end learning," in *Advances in Neural Information Processing Systems 18*, Y. Weiss, B. Schölkopf, and J. C. Platt, Eds. Vancouver, British Columbia, Canada: MIT Press, 2006, pp. 739–746. [Online]. Available: http://yann.lecun.com/exdb/publis/pdf/lecun-dave-05.pdf

[4] I. L. Davis and A. Stentz, "Sensor fusion for autonomous outdoor navigation using neural networks," *IEEE International Conference on Intelligent Robots and Systems*, vol. 3, pp. 338–343, 1995.