Don't Make the Same Mistakes Again and Again: Learning Local Recovery Policies for Navigation from Human Demonstrations

Francesco Del Duchetto[1], Ayse Kucukyilmaz[1], Luca Iocchi[2], and Marc Hanheide[1]

***Abstract*—In this paper, we present a human-in-the-loop learning framework for mobile robots to generate effective local policies in order to recover from navigation failures in long-term autonomy. We present an analysis of failure and recovery cases derived from long-term autonomous operation of a mobile robot, and propose a two-layer learning framework that allows to detect and recover from such navigation failures. Employing a learning by demonstration (LbD) approach, our framework can incrementally learn to autonomously recover from situations it initially needs humans to help with. The learning framework allows for both real-time failure detection and regression using Gaussian processes (GPs). Our empirical results on two different failure scenarios indicate that given 40 failure state observations, the true positive rate of the failure detection model exceeds 90%, ending with successful recovery actions in more than 90% of all detected cases.**

***Index Terms*—Service Robots, Failure Detection and Recovery, Learning from Demonstration.**

I. Introduction

THE ability to detect and recover from errors during navigation is an essential ability for autonomous service robots, which are designed to function in human environments for extended periods of time. However, navigation failures are still common in state-of-the-art systems, where robots get stuck near obstacles or their global navigation plans fail as we show with our analysis of about a year of autonomous operation in a care home. Such navigation errors pose a severe limitation on the long-term autonomy, since they may cause a breakdown of the whole robotic operation.

In order to mitigate operational errors, different recovery strategies have been proposed in the literature. Among these, some take advantage of being in a human-inhabited environment, where the robot can actively consult a nearby human for assistance. Asking questions when plans are ambiguous or presenting the humans with options have been used in human-robot interaction to achieve semi-autonomous operation in real-world scenarios [1], [2], [3], [4], [5]. Most of these studies take on a conversational approach to bring the human in the loop to verbally clarify task goals. In this study, we adopt a learning by demonstration (LbD) approach to teach robots recovery policies by requesting physical human demonstrations.

In our framework, an operator teaches the robot when a failure occurs and demonstrates how to recover from that situation by assuming its control. Once trained for failure detection, the robot can autonomously detect situations that it cannot possibly handle using an existing navigation policy, and either run a previously learned recovery behaviour or ask for a new demonstration depending on the variance of the predicted actions for the current failure.

Fig. 1: Scitos G5

II. Related Work

Robust navigation in long-term scenarios in somewhat dynamic environments remains a challenge in robotics. The "Office Marathon" [6] uses a navigation system based on *move_base*, which is also used in our STRANDS system [7], although only aimed to reach marathon distance. Recently, Biswas and Veloso [8] have traversed over 100 km with their fleet of CoBots. However, they also report that a number of expert interventions were required in these runs, despite optimising the navigation system explicitly for robustness. Similarly, a way to put humans in the loop to increase navigational autonomy is presented in [5], where a mobile robot actively seeks a human when it needs assistance for spatially-situated actions, such as pushing buttons to access an elevator. Also, [9], [10], [11] suggest using the paradigm of LbD to learn navigation policies from human demonstrations.

LbD provides an intuitive way for humans to teach robots physical skills. Kinesthetic teaching has been a popular approach to program new skills on a robot, where the human acts as the *teacher*, who physically demonstrates the task for the robot to generalize from [12], [13], [14]. The key idea behind LbD is to learn a skill with minimum number of demonstrations. Hence, providing new demonstrations where necessary

Manuscript received: February 24, 2018; Revised: June 5, 2018; Accepted: July 11, 2018 .

This paper was recommended for publication by Editor Dongheui Lee upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the EPSRC ("NCNR", EP/R02572X/1) and EU H2020 ("ILIAD", 732737).

[1]F. Del Duchetto, A. Kucukyilmaz and M. Hanheide are with the L-CAS, University of Lincoln, UK. {fdelduchetto, akucukyilmaz, mhanheide}@lincoln.ac.uk

[2]L. Iocchi is with DIAG, University of Rome "Sapienza", Italy. iocchi@dis.uniroma1.it

Digital Object Identifier (DOI): see top of this page.

is crucial to improve skill learning by making the models generalize better to new situations. How to switch between autonomous mode and human demonstration has been a topic of interest in the literature. Lockerd and Breazeal [15] use a Bayesian likelihood method with a fixed threshold to identify low confidence behaviors, which require new demonstrations. Similarly, Grollman and Jenkins [16] determine the uncertainty of actions by identifying low confidence states, again, using a threshold, to request demonstrations. Chernova and Veloso [17] use multiple automatically-calculated thresholds to compute low confidence regions within the task space, and propose a confidence-based autonomy switching technique. In [18], the robot gradually expands its range-of-motion via asking for new demonstrations from the human.

In this study, we take on a LbD approach for robotic navigation, where the robot automatically detects failures and learns recovery trajectories from human demonstrations. We evaluate the variance of the predicted actions as a measure of model confidence, and provide demonstrations where necessary. To our knowledge, ours is the first comprehensive framework for mobile navigation where the robot learns how to deal with abstract navigation failures from human expertise in a realistic scenario, rather than learning specific task descriptions.

As mentioned earlier, we take on a Gaussian process (GP)-based approach in our framework for dealing with navigation errors. Even though LbD is most popularly used for object manipulation tasks, the use of GPs for navigation has been explored by different researchers. Nardi and Stachniss [19] present a GP-based navigation system, designed to be robust against dynamic environments. They propose a planning approach to predict the trajectories of moving agents and adjust local plans to avoid collisions based on previous navigation experience. This technique illustrate the robustness of GPs for generalizing trajectory observations. Hüntemann et al. [20] use a Bayesian approach in conjunction with GPs to model user driving behavior to reason about the user's local navigation plan. In [21], [22], GPs are used to learn assistance policies by demonstration, where a human provides guidance to a user by means of taking over the control of the navigation. In [23], [24], the utility of GPs for learning continuous shared control policies from human demonstrations is proposed to provide active navigation assistance to a wheelchair user. In our study, we extend these ideas to an unexplored field and focus on real-life problems faced in long-term autonomy of social robots. Our focus, therefore is to improve the autonomy of the robots to make them more stable against navigation failures.

III. Analysis of Navigation Failures

In our previous work [7], [25], an autonomous mobile robot (See Fig. 1) was deployed in a care home for a total of just over a year, in the context of the STRANDS project[1]. This experiment was split over three individual deployments, spread over a period of three years. Here, the robot served as a mobile info-terminal and was also engaged in occupational therapy sessions [25]. It was left without any technician or researcher on site; hence, the robustness of navigation in this dynamic and challenging environment was essential. In the rest of this section, we outline some key findings from the of the navigation issues in this long-term deployment.

STRANDS Care Robot Navigation System: The robot's navigation system was based on the *ROS move_base* navigation framework[2], augmented with a topological navigation component [26], which also serves in the architecture proposed in this paper (See Section IV). *move_base* is widely employed by many robot systems and has proven its suitability as a generic navigation framework. However, in real-world applications with unforeseen dynamics, navigation commands sent to the local planner [27] will often fail due to sensing noise, obstacles appearing close to the robot, or control inaccuracies, despite manual and automatic [28] parameter optimisation having been conducted extensively.

Failure Situations: Typical navigation failures encountered during the deployment of the robot were: robot getting stuck on carpet, bumper being pressed and no valid global/local navigation plan being generated. For each of these failure situations, a specific ad-hoc recovery behaviour was defined.

Ad-Hoc Recovery Behaviours: In order to meet the robustness requirement and deal with local planning failures, we initially developed a dedicated recovery system, featuring hard-coded autonomous recovery behaviours. These ranged from a simple (i) *wait and retry* behaviour, which cleared the local cost-map and then re-issued the navigation command, over a (ii) *backtrack* behaviour, which reversed the last 15 (1 second) motion commands sent to the robot to return to a previous position, up to an (iii) *interactive help seeking* behaviour, in which the robot would ask any human in its surrounding –verbally and by screen display– to push it from its current position to a free area. Asking the human was designed as a fall-back behaviour, triggered after all autonomous behaviours had failed to successfully take the robot to its destination.

Analysis of Navigation Issues and Recoveries: Over the three individual deployments, the robot travelled more than $160\ \mathrm{km}$, during which a total of 5591 recoveries (autonomous and involving the human) were required. This averages to almost 35 recoveries per km – or, in other words, one recovery every $28.6\ \mathrm{m}$ travelled, in an environment spanning an area of approximately $54\ \mathrm{m} \times 135\ \mathrm{m}$, as shown in Fig. 2. In many of these cases, the robot could successfully recover using one of the hard-coded autonomous recovery behaviours; yet in 1605 cases (i.e., 28.7% of all cases) *human* help was required, as all autonomous recoveries had failed to succeed. Fig. 3 presents a breakdown of the cases where the robot sought help from humans (e.g., staff, visitors or residents in the care home), in the form of an ordered histogram. It shows the number of requests occurring at each edge within the topological map shown in Fig. 2; hence indicates the spatial distribution of the problem areas. As seen in the figure, the edge where human recovery was requested most is plotted as the leftmost bar with a total of 146 recovery requests. Out of these 146 cases, only in 19 cases, the robot received help. When the robot was

[1] http://strands-project.eu/

[2] http://wiki.ros.org/move_base

Fig. 2: Map of the care home, with superimposed topological map. The annotations indicate specific waypoints relevant for the robot's task and the areas in which video recording was permitted for further analysis in order to comply with local privacy regulations. The analysis presented in this paper involves the navigation errors along all edges in this topological map.

not helped, it aborted its current task and failed the operation, often requiring remote manual recovery by a technician.

Conclusion and Objectives: The conclusions drawn from this study directly lead to our objectives in this paper:

1) *Failures are spatially clustered*
   As Fig. 3 shows, 80% of all error cases are distributed over only about 25 edges in our quite fine-grained topological map (Fig. 2), indicating a strong dependency of failures on location. Indeed, it was observed that the robot repeatedly failed at the very same spot as it was not taking advantage of the learning opportunities. In this paper, this limitation shall be addressed by *learning suitable recovery strategies from few demonstrations given by a helping human*, to avoid making the same mistake again and again.
2) *Human help is rare and precious*
   Physical human assistance is a scarce resource. Overall, out of the 1605 cases where the robot asked for help, only in 257 (16%) it received help that allowed it to continue its operation. Furthermore, the number of cases where the robot was successfully helped by a human varied significantly between edges. This evidences a need to develop recovery models that are not bound to specific locations, but to navigational situations. Hence, another objective of this study is to *correctly classify a known navigational situation and to invoke the correct recovery policy*.

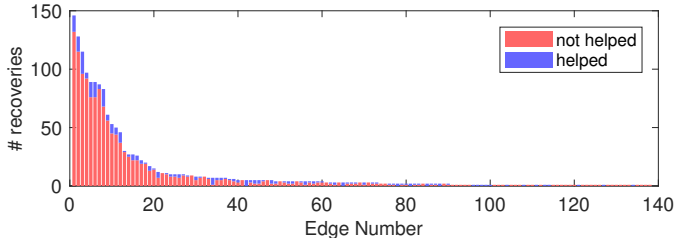


Fig. 3: Edges in topological map requiring human recovery, ordered with respect to the number of issued recovery actions

IV. Gaussian Processes for Learning Local Recovery Policies

In this work we describe a LbD framework[3] that gives a robot the ability to detect failures and recover from them during navigation. During a navigation task, failures are manifested in different ways (such as spinning in place or crashing against an obstacle) and they may happen for different reasons (e.g. local navigation can't generate a plan or due to sensing errors). In our context, we define a *failure situation* as *the situation in which the robot is not able to progress toward the goal, because it is not moving or it is performing some counterproductive behavior*. Some failures are detected by the underlining navigation system, whereas others may not. In order to take into account all such situations independent from the navigation system, we delegate the task of deciding and signaling a failure situation to the humans.

Our local navigation framework makes use of a two-layer cascaded learning approach, where a GP classification model is trained to detect failure situations, and a GP regression model is trained to learn local recovery trajectories. We present an active learning approach, where the human operator presents demonstrations when necessary, to incrementally teach the robot how to handle failure situations. To begin our discussion, we firstly present our overall navigation architecture, followed by the methodology behind GPs, the features and the modelling approach adopted in the experiments.

A. Overall Architecture

Fig. 4 shows the overall software architecture developed in this study. The global navigation system[4] extends the *move_base* package from the ROS navigation stack.

Global task planning is done using a conditional Petri net planner (*PNP*) [29], which provides high level descriptions of actions within a topological map of the environment. A topological map is an undirected graph, where vertices stand for possible waypoints that the robot can navigate to, and

[3]The source code will be available at https://github.com/LCAS

[4]The global navigation system is based on open-source software developed during the STRANDS project [7]

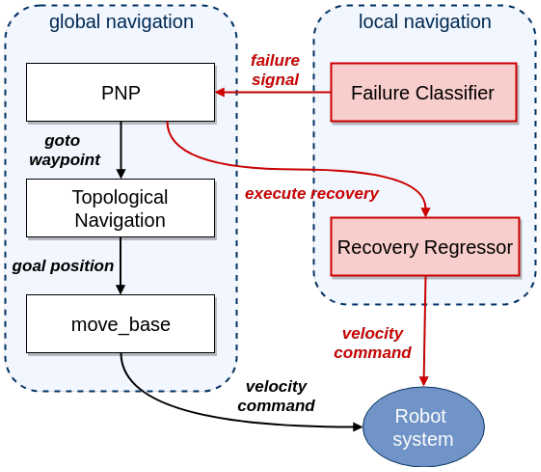


Fig. 4: General software architecture: white boxes indicate the global navigation functionality, whereas the red ones implement our strategy for local recovery policy generation.

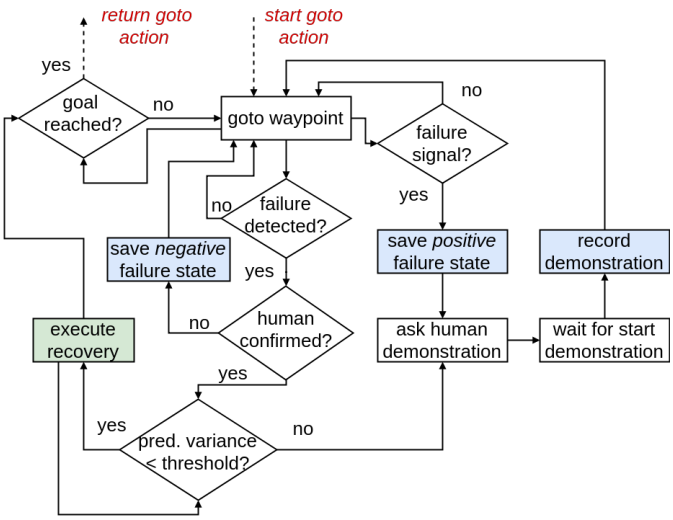


Fig. 5: Flowchart showing the human-in-the-loop approach for collecting failure state observations and recovery demonstrations, as well as recovery execution during *goto* actions

the edges define the connections between waypoints. During navigation, PNP issues *goto* commands to the *topological navigation* node [26], which extracts the goal position and sends that to *move_base* 2D navigation node to compute commanded velocities to drive the robot.

In order to deal with failure situations, PNP plan descriptions are conditioned by *execution rules*, which provide a mechanism for interrupting the navigation in case of failures. For more details on Petri net plans, the reader is referred to [29]. For the approach in this paper, it is sufficient to consider PNP as a way to integrate exception handling and therefore recovery into task planning. Our local recovery framework (red nodes in Fig. 4) makes use of PNP's conditioning property and issues failure signals to override the operation of topological navigation and *move_base*. Specifically, *failure classifier* is responsible for identifying a navigation failure situation, and issuing a signal to PNP to demand a recovery action. Receiving this, PNP interrupts the *goto* action and invokes the *recovery regressor* to run a local recovery policy.

The failure classifier and the recovery regressor constitute the two layers in our interactive LbD framework. Fig. 5 shows the process through which the human supplies new failure state observations and demonstrates recovery trajectories (blue nodes) to get the robot to execute the learned recovery policy (green node). The failure classifier is a binary GP classification model trained with a labelled set of positive and negative observations for failure states. Positive observations represent the environmental context leading to the failure, and are provided by the human operator, who interactively signals for a new failure situation whenever (s)he observes a navigation error. This signalling halts the global navigation and the corresponding data is added to the training set, labelled as a *positive failure*. Once the model is trained for failure detection, the robot can autonomously detect situations that it cannot handle properly, before they occur. When a failure is detected, the human is prompted to confirm the correctness of the detection. In case the human rejects the detected state as a failure, a *negative failure* observation is added to the training set, and the classification model is updated. This action also resumes the global navigation plan.

In our setup, upon receiving a failure signal, the human is prompted to assume control of the robot to demonstrate how to recover from the failure situation. The demonstration is used to train a GP regression model (recovery regressor), to be used as the local recovery policy for failure situations. By using a Bayesian approach we get a variance for the predictions of the regression model. If the human verifies the pertinence of the detection, this variance is used to evaluate the goodness-of-fit for the regression model. In particular, we prevent the execution of the recovery, in situations where the prediction variance is too high, i.e. the model is overfitting the data[5]. In such a case, the human is asked to provide a new demonstration to improve model performance. If the variance is smaller than a predefined threshold τ[6], the GP regression model is used to execute the learned recovery policy.

B. Methodology

In this study, we are interested in 1) detecting a navigation failure in an online fashion, and 2) learning the corresponding local recovery plan for the specific failure situation, i.e. the velocity commands to be issued on the robot depending on the environmental context. For these purposes, we learn two GPs, respectively for classification and regression [30].

GP is powerful non-parametric Bayesian technique that describes prior distributions over functions. Formally: given input space $\mathcal{X}$, $f : \mathcal{X} \mapsto \mathbb{R}$ is a GP, if for every input $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$, s.t. $x_i \in \mathcal{X}$, the output vector $f(x) = [f(x_1), f(x_2), \ldots, f(x_n)]^T$ is normally distributed. The function evaluation in GPs is drawn from a multi-variate Gaussian distribution, fully specified by a zero mean vector μ and a kernel matrix $\mathbf{K}$, s.t. $f \sim \mathcal{N}(\mu, \mathbf{K})$.

[5]This situation can be encountered especially at the beginning of the trials, when the model is trained with many features but has too few observations.

[6]In this study we empirically set $\tau = 0.03$.

For classification and regression models, two different kernel choices were implemented. The failure classifier model was trained using a linear covariance kernel:

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x} \cdot \mathbf{x}' \quad , \tag{1}$$

which required short training time, and observed to be sufficient for separating failure and non-failure states.

The recovery regression model was trained using a squared exponential covariance kernel, which can model short and medium term irregularities in input space [30]:

$$k\left(f(\mathbf{x}), f(\mathbf{x}')\right) = \exp\left\{-\frac{|\mathbf{x} - \mathbf{x}'|^2}{2l^2}\right\} \quad , \tag{2}$$

where l is a free hyper-parameter that defines the characteristic length-scale of the process. The length-scale denotes the minimum distance you have to move in the input space for the function values to be automatically uncorrelated. The initial length-scale of the kernel was set to 40 seconds, which is close to the average duration for a typical demonstration. The final length-scale was optimized using maximum likelihood estimation during model selection. Due to the high-dimensional input space in our task, automatic relevance determination (ARD) [31] was turned off to reduce training time.

For a test instance $\mathbf{x}_*$, the predictive distribution of the target value, y_*, is given by

$$\begin{aligned} &p(y_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}, \hat{\Theta}) = \\ &\mathcal{N}(y_*|\mathbf{k}^\top \mathbf{K}^{-1}\mathbf{y}, k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}^\top \mathbf{K}^{-1}\mathbf{k} + \sigma_n^2), \end{aligned} \tag{3}$$

where $\mathbf{k}^{N\times 1}$ consists of elements of the kernel matrix $[\mathbf{k}]_i = k(\mathbf{x}, \mathbf{x}_i)$, $\mathbf{X}$ and $\mathbf{y}$ are respectively the training observations and the corresponding targets, σ_n^2 is the hyper-parameter denoting noise variance, and $\hat{\Theta}$ denotes the estimated optimal hyper-parameters related to the covariance function and the error variance, estimated by maximizing the marginal-likelihood:

$$p(\mathbf{y}|\mathbf{X}, \Theta) = \mathcal{N}(\mathbf{y}|0, \mathbf{K}). \tag{4}$$

As mentioned in Section IV-A, the GP classifier is used to detect failures. The model is trained with positive and negative examples for failure states, where a positive observation indicates the existence of a genuine failure. We define a *failure state* as the environmental context that leads to a failure situation. In response to detected failure situations, we train a GP regression model which will be used to generate the recovery control actions. Since the failure states and recovery control actions are diverse by nature, our framework requires a rich spatio-temporal representation of the environment, which involves information about the trajectory and obstacles around the robot.

In order to have a good spatial representation and to decouple the performance of the approach on the sensor, we simulate virtual laser scan data surrounding the robot to denote a sparse obstacle map, consisting of N points (See Fig. 6). However, one can consider using directly the laser scan data from the sensor. We infer the virtual laser scan data from the local costmap provided by *move_base*. In order to simulate a 360° cover for laser scans (See Fig. 6 right), we cast N rays in the local costmap, originating from the robot's center pose, where the i^{th} ray propagates in the direction of $\alpha_i = \frac{i\pi}{N}$ radians, $i \in \{1, \ldots, N\}$. Raytracing these rays' intersection with obstacles within a given radius, provides us with N simulated laser scan points.

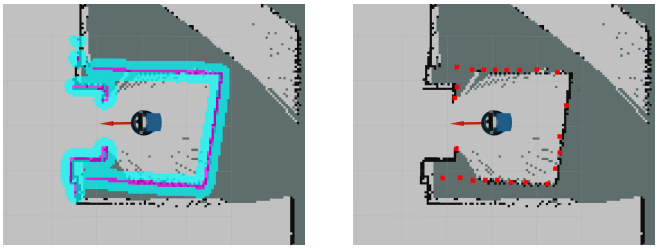


Fig. 6: Left: Local costmap around the robot. Right: Laser scan simulated by raytracing 30 points within the local costmap.

In order to capture the temporal structure of the environment in failure situations, we use sliding windows covering a fixed length of time to populate our feature set. In this way, our features consist of laser scan data spanning the full 360° circle around the robot, occurring inside a past window of data. Our initial observations indicated that the recovery performance under specific failure scenarios is affected by the size of the scan window. Intuitively, a larger window size allows the model to exploit the history of the trajectory, and is suitable for longer demonstrations; whereas a smaller one makes the models depend more on recent samples. The exact window sizes are set empirically for the experimental scenarios in this study, as mentioned in section V.

These spatio-temporal laser scan features are used to train both the GP classification model –for failure detection– and the GP regression model –to learn velocity commands that drive the robot on the recovery trajectory.

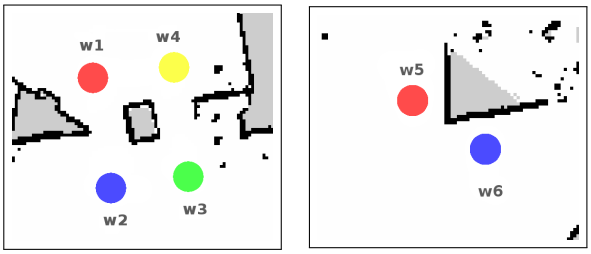
V. Experimental Validation

To evaluate the proposed technique for local navigation recovery, we gathered data using the STRANDS mobile care robot, SCITOS G5, (See Fig. 1) at Lincoln Centre for Autonomous Systems (L-CAS).

A. Experimental Scenarios and Procedure

We evaluated our technique in two different navigation scenarios, where the robot typically fails to reach the goal using ROS *move_base*. Fig. 7 illustrates these scenarios within our experimental area (also see supplementary video).

- *Door passing*: This scenario is encountered when the robot attempts to pass through a narrow passage between two obstacles to reach a goal waypoint (See Fig. 7a). We observed that the robot occasionally gets stuck as it detects a non-existing potential collision, and cannot evaluate the necessary navigation commands.
- *Corner*: In this scenario, the robot needs to turn a corner in order to reach a goal, which is positioned very close to the obstacle (See Fig. 7b). A failure happens when the robot also gets close to the corner and ends up with a continuous spinning motion on its place.



(a) Door passing scenario **(b)** Corner scenario

Fig. 7: Map of the laboratory area used in the experiments and the programmed waypoints for robot navigation.

In each scenario, the robot was programmed to continuously loop between two waypoints w_i and w_j, until the experimenter stopped the operation. As seen in Fig. 7, for the door passing scenario, we tested our approach with two waypoint pairs, $w_1 \leftrightarrow w_2$ and $w_3 \leftrightarrow w_4$; whereas for the corner scenario, we only used a single pair of waypoints, $w_5 \leftrightarrow w_6$. The waypoints were located around the aforementioned failure scenarios; hence it was expected that the robot will occasionally fail its global navigation plan.

The features used in the two scenarios are $N = 30$ laser scan points collected over a temporal window of 10 samples (i.e. 1 second) for scenario 1: *door passing*, and 20 samples (i.e. 2 seconds) for the scenario 2: *corner*.

We designed two experiments to provide insight about the performance of our framework. The first experiment is designed to evaluate the performance of the failure classifier. We collected failure observations while the robot is traversing between the waypoint pairs. In doing so, we created different training sets of size 5, 10, 20, and so on, increasing the number of observations by 10 until we can no longer provide any new positive observations. A GP classification model was trained for each training set size, and the detection performance was evaluated over 20 failure runs through the waypoints (10 in each direction), where the robot was expected to encounter a failure situation. We recorded the number of times a failure is correctly detected out of these 20 situations. The runs, in which the robot could successfully reach the goal waypoint without facing a failure state, were restarted without logging the data.

This experiment allowed us to evaluate the performance of detection with respect to training set size. At the end of the experiment, we selected the optimal training set size regarding the precision of the classifier to correctly classify failure states, and used this well-trained detector to evaluate the recovery policies.

In the second experiment, we followed a similar procedure, and trained the recovery model with different number of demonstrations. The demonstrations were provided by the human via remotely operating the robot using a gamepad; however, without loss of generality, the same commands could have been estimated via physically pushing the robot. Similar to the first experiment, we trained the model with an increasing number of demonstrations; this time, in increments of 5, i.e. 5, 10, 15 up to 35. The training size was limited to 35 in this case due to increased training time. A GP regression model is learned for each training set size; and the number of successful recovery trajectories was recorded over 20 failure runs through the waypoints (10 in each direction). A recovery was considered successful if the robot was able to reach the goal, starting from a failure state, only by using a sequence of velocity commands from the learned recovery model. In this evaluation scenario, we only focused on successful detection cases to isolate the performance of the recovery regressor from that of the failure classifier. In doing so, we discarded all the runs, in which the failure was not detected successfully or the robot didn't encounter a failure at all. The discarded runs were restarted and not counted in the total number of executed runs.

VI. Results and Discussion

This section presents the results of our experimental study to evaluate the performance of the failure detection and local recovery control modules. In total, we trained and evaluated three detection models and three recovery models (one for each waypoint pair $w_1 \leftrightarrow w_2$, $w_3 \leftrightarrow w_4$ and $w_5 \leftrightarrow w_6$). During our experiments we observed that failure detection and recovery was needed in about 82% of the cases, as the robot wasn't able to reach the commanded goal.

A. Failure Detection Performance

In order to provide insight on model performance, we investigated the sensitivity (i.e. the true positive rate) and the precision of the classifier. The sensitivity reports the percentage of correctly identified failures among all failures that are encountered during the experimental trial, whereas precision measures what percentage of the detected failures are genuine.

Fig. 8 (a-b) shows the sensitivity for the door passing and corner scenarios. In both, we observed that the detector sensitivity is maximized when the dataset size approached 40-50 observations. Practically, after that point, it was difficult to collect more positive observations (true failure states) under both scenarios, as most of the possible failure states had already been learned. For this reason, in the door passing scenario the number of training samples is capped at 50.

Fig. 8 (e-f) plots the distribution of positive and negative samples in the dataset used for training the classifier under both scenarios. For the door passing scenario, the number of negative samples –denoting false positives– were consistently low, indicating good overall precision of the model (See Fig. 8 (c)). On the other hand, in the corner scenario, the number of negative observations grew significantly, eventually to out-number the positive samples, as we continued the procedure. This deteriorated the precision of the classifier as shown in Fig. 8 (d).

For both scenarios, we selected the training set size to optimize for sensitivity and precision. As a result, the number of observations to train the optimal detector, which will be used to detect failures when evaluating the recovery control performance, was set to 50 for the door passing scenario, whereas it was set to 40 for the corner case.

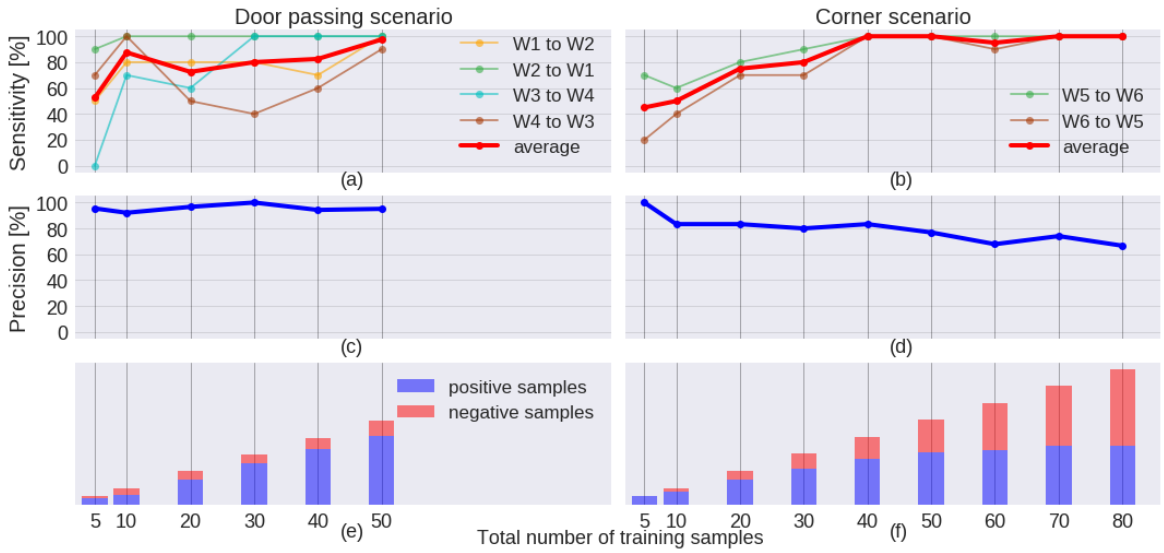


Fig. 8: Evaluation of the failure classifier model and analysis of the dataset composition for the experimental scenarios.

B. Recovery Control Performance

In this section, we report our results to evaluate the accuracy of the learned recovery actions. Similar to the case of failure detection, we investigate the performance of our recovery models with respect to the number of demonstrations used to train the model. In order to assess performance, we investigated the number of successful recovery trajectories.

Fig. 9 shows the percentage of successful recovery trials on both scenarios as a function of the number of demonstrations. Evidently, the recovery performance got better with the number of demonstrations, reaching values over 90% for both scenarios after the model is provided with 30 demonstrations.

Fig. 10 plots the trajectories recorded during our experiments in the corner scenario for the human demonstrations, as well as those generated by the machine learned recovery policy. The figure illustrates that training only one detection model, we were able to distinguish different failure cases. The illustrated recovery trajectories indicate that a single regression model was able to generalize the demonstrations applied by the human to reproduce his/her operation under similar but not identical failure situations.

VII. Conclusion and Outlook

In this work, we proposed a technique to generate local policies for motion-level control recovery, in order to equip robots with the ability to exploit demonstrations provided by humans in long-term scenarios. We highlighted the relevance of such an approach from the analysis of the data gathered from long-term operation in a care home. We used a two-layer cascaded LbD technique using GPs, and adopted an active learning approach, where the system automatically evaluates the relevance of failure states to already seen ones, and demands new demonstrations when necessary.

Our technique works with real or simulated laser scan data to extract the environmental context. Hence, it can be applied to a range of robots with similar sensing capabilities, and provides a platform-independent approach for human-in-the-loop recovery generation for navigation.

We evaluated our approach over two different real-world failure scenarios. Our experiments indicate the utility of the technique to learn active local navigation recovery policies from human demonstrations. However, our technique depends on manually-set parameters for specific failure situations. Our future work will focus on automated parameter selection to generalize for scenario-dependent requirements. Additionally, our active learning approach will be supplemented with an automated decision procedure to select the necessary number of training samples based on model precision and sensitivity to avoid model degradation.

As future work, we intend to generalize our method to work with multiple failure situations and provide a more general framework, suitable for long-term deployment of robots in human spaces. In doing so, we plan on extending the experimental study to test with multiple human demonstrators. Additionally, we plan to identify similar techniques and provide a comparison with other methods.

Finally, in addition to the results we presented in this paper, we collected preliminary data on another robotic platform, MARRtino[7], using a similar approach to learn how to push aside an obstacle from the path that the robot is traversing. Sample results are illustrated in the supplementary video. In order to learn the trajectory, we used a GP regression model, as described in this paper, while an image classifier was in charge of recognizing the type of obstacle. Even though the classifier is different than what we used in the current study, we show that the framework is usable on various platforms and that the regressor model is able to learn different navigation behaviors not necessarily limited to failure recoveries.

References

[1] T. Fong, C. Thorpe, and C. Baur, "Robot, asker of questions," *Robotics and Autonomous Syst.*, vol. 42, no. 3-4, pp. 235–243, 2003.

[7] https://sites.google.com/dis.uniroma1.it/marrtino

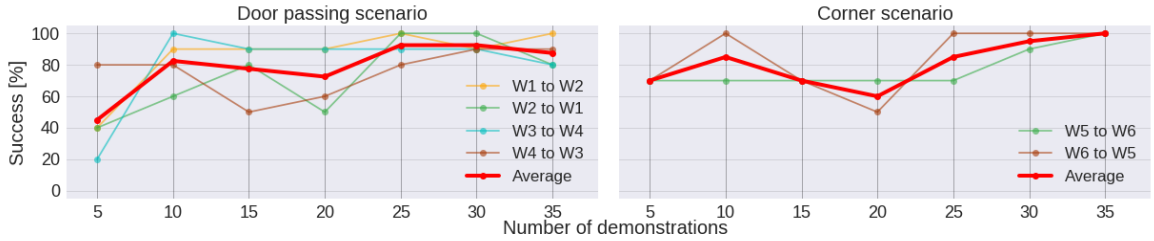
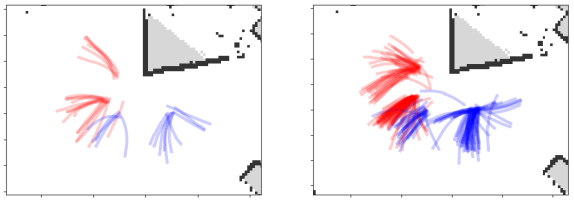


Fig. 9: Evaluation of regression models for experimental scenarios.



(a) Demonstrated trajectories **(b)** Recovery trajectories

Fig. 10: (a) Demonstrated and (b) recovery trajectories for the corner scenario. Red trajectories are recorded when the robot is going in the direction of $w_5 \rightarrow w_6$, and blue trajectories are collected while going in the $w_6 \rightarrow w_5$ direction.

[2] M. K. Lee, S. Kielser, J. Forlizzi, S. Srinivasa, and P. Rybski, "Gracefully mitigating breakdowns in robotic services," in *Proc. of the 5th Int. Conf. on HRI*. IEEE Press, 2010, pp. 203–210.

[3] M. Shiomi, D. Sakamoto, T. Kanda, C. T. Ishi, H. Ishiguro, and N. Hagita, "A semi-autonomous communication robota field trial at a train station," in *3rd Int. Conf. on Human-Robot Interaction*. IEEE, 2008, pp. 303–310.

[4] M. Cakmak and A. L. Thomaz, "Designing robot learners that ask good questions," in *Proc. of the 7th annual Int. Conf. on HRI*. ACM, 2012, pp. 17–24.

[5] S. Rosenthal and M. M. Veloso, "Mobile robot planning to seek help with spatially-situated tasks." in *AAAI*, vol. 4, no. 5.3, 2012, p. 1.

[6] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige, "The Office Marathon: Robust navigation in an indoor office environment," in *Int. Conf. on Robotics and Automation*. IEEE, may 2010, pp. 300–307.

[7] N. Hawes, C. Burbridge, F. Jovan, L. Kunze, B. Lacerda, L. Mudrová, J. Young, J. Wyatt, D. Hebesberger, T. Kortner, *et al.*, "The strands project: Long-term autonomy in everyday environments," *Robotics & Automation Mag.*, vol. 24, no. 3, pp. 146–156, 2017.

[8] J. Biswas and M. M. Veloso, "Localization and navigation of the CoBots over long-term deployments," *The Int. J. of Robotics Research*, vol. 32, no. 14, pp. 1679–1694, dec 2013.

[9] D. Silver, J. A. Bagnell, and A. Stentz, "Active learning from demonstration for robust autonomous navigation," in *Int. Conf. on Robotics and Automation*. IEEE, 2012, pp. 200–207.

[10] M. Zucker, N. Ratliff, M. Stolle, J. Chestnutt, J. A. Bagnell, C. G. Atkeson, and J. Kuffner, "Optimization and learning for rough terrain legged locomotion," *The Int. J. of Robotics Research*, vol. 30, no. 2, pp. 175–191, 2011.

[11] M. Kalakrishnan, J. Buchli, P. Pastor, and S. Schaal, "Learning locomotion over rough terrain using terrain templates," in *Int. Conf. on Intelligent Robots and Syst.* IEEE, 2009, pp. 167–172.

[12] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," in *Springer Handbook of Robotics*, 2008, pp. 1371–1394.

[13] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robot. Auton. Syst.*, vol. 57, no. 5, pp. 469–483, May 2009.

[14] L. Rozo, P. Jiménez, and C. Torras, "A robot learning from demonstration framework to perform force-based manipulation tasks," *Intelligent service robotics*, vol. 6, no. 1, pp. 33–51, 2013.

[15] A. Lockerd and C. Breazeal, "Tutelage and socially guided robot learning," in *Proc. Int. Conf. on Intelligent Robots and Syst.*, vol. 4. IEEE, 2004, pp. 3475–3480.

[16] D. H. Grollman and O. C. Jenkins, "Dogged learning for robots," in *Int. Conf. on Robotics and Automation*. IEEE, 2007, pp. 2483–2488.

[17] S. Chernova and M. Veloso, "Multi-thresholded approach to demonstration selection for interactive robot learning," in *Proc. of the 3rd Int. Conf. on HRI*. ACM, 2008, pp. 225–232.

[18] E. Gribovskaya, F. dHalluin, and A. Billard, "An active learning interface for bootstrapping robots generalization abilities in learning from demonstration," in *RSS Workshop Towards Closing the Loop: Active Learning for Robotics*, vol. 62, 2010.

[19] L. Nardi and C. Stachniss, "User preferred behaviors for robot navigation exploiting previous experiences," *Robotics and Autonomous Syst.*, vol. 97, pp. 204–216, 2017.

[20] A. Hüntemann, E. Demeester, E. Vander Poorten, H. Van Brussel, and J. De Schutter, "Probabilistic approach to recognize local navigation plans by fusing past driving information with a personalized user model," in *Int. Conf. on Robotics and Automation*. IEEE, 2013, pp. 4376–4383.

[21] H. Soh and Y. Demiris, "When and how to help: An iterative probabilistic model for learning assistance by demonstration," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Syst.*, Nov 2013, pp. 3230–3236.

[22] ——, "Spatio-temporal learning with the online finite and infinite echo-state gaussian processes," *IEEE Trans. on Neural Networks and Learning Syst.*, vol. 26, no. 3, pp. 522–536, March 2015.

[23] A. Kucukyilmaz and Y. Demiris, "One-shot assistance estimation from expert demonstrations for a shared control wheelchair system," in *Int. Symp. on Robot and Human Interactive Communication*. IEEE, 2015, pp. 438–443.

[24] ——, "Learning shared control by demonstration for personalized wheelchair assistance," *Trans. on Haptics*, 2018.

[25] M. Hanheide, D. Hebesberger, T. Krajnik, and others, "The When, Where, and How: An Adaptive Robotic Info-Terminal for Care Home Residents A long-term Study," in *Proc Int. Conf. on Human-Robot Interaction*. ACM/IEEE, 2017.

[26] J. Pulido Fentanes, B. Lacerda, T. Krajník, N. Hawes, and M. Hanheide, "Now or later? predicting and maximising success of navigation actions from long-term experience," in *Int. Conf. on Robotics and Automation*, 2015.

[27] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Mag.*, vol. 4, no. 1, pp. 23–33, mar 1997.

[28] J. P. Fentanes, C. Dondrup, and M. Hanheide, "Navigation testing for continuous integration in robotics," in *UK-RAS Conf. on Robotics and Autonomous Syst.*, December 2017.

[29] V. A. Ziparo, L. Iocchi, D. Nardi, P. F. Palamara, and H. Costelha, "Petri net plans: a formal model for representation and execution of multi-robot plans," in *Proc. of the 7th Int. joint Conf. on Autonomous agents and multiagent Syst. -Volume 1*. Int. Foundation for Autonomous Agents and Multiagent Syst., 2008, pp. 79–86.

[30] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*, ser. Adaptative computation and machine learning series. The MIT Press, 2006.

[31] C. M. Bishop, *Pattern recognition and machine learning*, 4th ed. springer New York, 2006.