FAILURE RESILIENCE IN LEARNED VISUAL NAVIGATION CONTROL

by

WANG ZISHUO
(B.E., Harbin Institute of Technology)

A THESIS SUBMITTED FOR THE DEGREE OF

MASTER OF COMPUTING

in

ARTIFICIAL INTELLIGENCE

in the

GRADUATE DIVISION

of the

NATIONAL UNIVERSITY OF SINGAPORE

2024

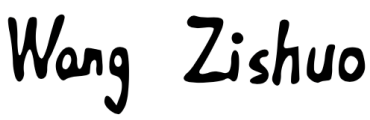
Supervisor:
Professor David Hsu

Examiners:
Professor Harold Soh

Declaration

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.



Wang Zishuo

June 2024

Acknowledgments

I want to convey my deepest gratitude to my supervisor, Professor David Hsu. His support, guidance, and profound insights in the field of robotics have made this thesis an inspiring journey for me. This unique experience has not only helped me complete my thesis but has also strengthened my skills and determination to continue my research journey in robotics.

I am also grateful to my friend Joel for his invaluable collaboration and patient support. His consistent assistance has provided both help with research and comfort throughout this challenging process.

I extend my sincere thanks to all the members of the AdaComp lab. Their expertise and kindness have greatly enriched my research experience and daily life.

Lastly, I want to express my heartfelt gratitude to my parents. Their unwavering support has been a constant source of motivation and encouragement throughout my master's journey.

Contents

Acknowledgments **i**

Abstract **iv**

List of Figures **v**

List of Tables **vi**

1 Introduction **1**

2 Literature Review **4**

2.1 Learned visual navigation 4

2.2 Failure detection with learned policies 5

2.3 Failure handling with learned policies 5

3 Method **7**

3.1 Problem formulation 7

3.2 System overview 8

3.3 Anomaly detection 9

3.4 Anomaly localisation 12

3.5 Failure handling 12

3.5.1 Self-recovery 13

3.5.2 Requesting human intervention 14

4 Experiments **15**

4.1 Experimental setup 15

4.2 Taxonomy of failures 15

4.3 Anomaly detection and localisation performance 17

4.4 Influence of VIB regularisation on policy 21
4.5 Navigation failure detection and handling performance 22

5 Conclusion **26**

Bibliography **28**

Abstract

While learning policies that navigate with rich visual information is a promising approach to robot navigation, it remains a challenge to robustly *detect* and *handle* their failures. Prior works often fail to (1) directly detect task-relevant failures induced by the policy, or (2) to handle the failure in a robust and informed manner. To this end, we propose *FaRe*, a framework for failure-resilience with learned visual navigation policies, that augments such policies with task-relevant failure detection and informed failure handling. Key to this is a novel unsupervised anomaly detection and localisation approach that can be embedded in such policies during training, with only mild changes to their architecture. We show through extensive real world experiments on a representative, off-the-shelf behavioural learned policy that our approach imbues it with the ability to robustly handle failures spanning a range of scenarios, both inside and outside its training distributions.

List of Figures

1.1 *FaRe*-DEC running on the Spot robot. 2

3.1 Overview of our failure-resilience framework 8
3.2 The neural network architecture of *FaRe*-DEC 9
3.3 Visual explanation of our VIB heuristic. 10
3.4 Anomaly localisation with *FaRe*-DEC 11
3.5 Self-recovery module 13

4.1 Taxonomy of failures for learned visual navigation policies 16
4.2 Taxonomy of failures in terms of data 16
4.3 Sampled failure cases from offline tests for question (1) and (2). 18
4.4 The navigation route of Task iii 22
4.5 Qualitative analysis of *FaRe*-DEC 25

5.1 Potential improvement of our system 27

List of Tables

4.1 Performance of anomaly detection and localisation 19
4.2 Performance of anomaly detection and localisation with different β factor 20
4.3 Performance of DECISION vs *FaRe*-DEC(policy) 21
4.4 Real world performance of *FaRe*-DEC 24

Chapter 1

Introduction

Visual navigation is an appealing approach to robot navigation, since it has the potential to harness rich visual information about the environment [1, 24] from readily available and cost-effective visual sensors [4]. In particular, end-to-end deep learning has become a key means of implementing effective visual navigation policies, because of its ability to capture task-relevant semantic information from visual inputs [1], and its robust performance [20, 26]. However, such learned policies are often black boxes that can make unreliable predictions without warning, especially when presented with novel situations outside their training distribution. This can induce navigation failures, by leading the robot into states from which the learned policy is unable to autonomously complete its navigation task.

To address this shortcoming, we propose a simple yet effective approach to imbuing learned visual navigation policies with *failure-resilience* (*FaRe*), which we define as the ability to *detect* and *handle* failures. A major cause for failures under data-driven learned policies is their unreliable predictions on inputs that lie outside the distribution learned from the training set, *i.e. anomalies* [24, 8, 29]. A common approach to detecting failures, which *FaRe* also takes, is using anomaly or novelty detection [24, 7, 29]. However, prior works often rely on detectors external to the policy that lack direct knowledge of what inputs are anomalies under the policy, and are generally unaware of which parts of the inputs are relevant to the navigation task, leading to task-irrelevant anomaly predictions and excessive pessimism [21]. To handle failures, most existing works terminate the task and request human intervention [21, 14, 16] with some executing set prior behaviours to prevent or recover from the failure [24, 29]. However, such failure-handling strategies often do not exploit knowledge of the anomalies faced to take informed actions for recovery.

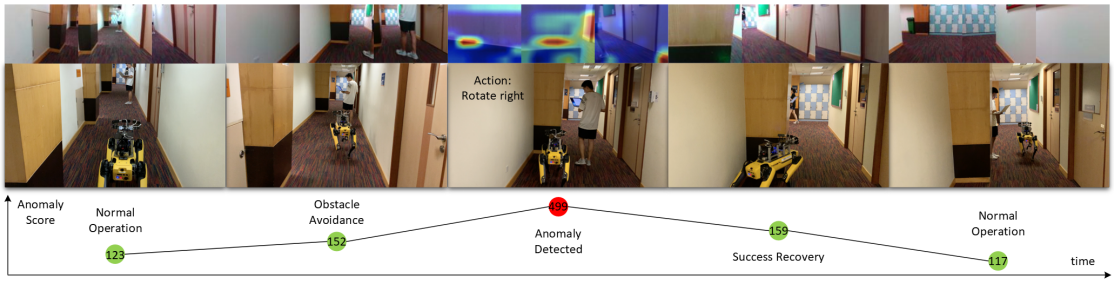


Figure 1.1: *FaRe*-DEC running on the Spot robot. From top to bottom are robot observations, whole scenarios, and the corresponding anomaly score. From left to right are five time steps extracted from a real navigation scenario to illustrate how our *FaRe* approach works. Our robot first tracks the path and avoids the human in normal operation. Unfortunately, after avoiding the human, it turns into a corner and is blocked by walls because the corridor is too narrow. Our method successfully detects this situation as a potential navigation failure and executes the informed recovery strategy guided by the anomaly localisation information, returning back to normal operation automatically.

FaRe seeks to address both these issues with a system that robustly detects and handles failures in an informed and predictable way. *FaRe* focuses on handling navigation failures arising from anomalies, and its core building blocks are a novel unsupervised anomaly detector and localiser that operate on the policies' latent space. Our key observation is that efficient and effective anomaly detection and localisation can be enabled by well-structured latent representations of sensory inputs, obtained by regularising policies with a variational information bottleneck objective [3]. Such representations discard task-irrelevant factors and induce a structured latent space in which distances are meaningful [22], which are properties that allow us to identify potential navigation-related failures due to anomalies, *i.e.* potential failures exhibit a greater distance from the prior distribution of regularisation compared to normal inputs. *FaRe* also handles detected failures, by attempting recovery strategies that exploit anomaly detection and localisation information to intelligently perturb the robot locally or backtrack, before escalating to requesting human intervention if needed.

We show that our approach of embedding anomaly detection and localisation into learned policies enables more effective identification of anomalies that have an impact on navigation. Through extensive real-world experiments, we find that *FaRe* imbues a representative, off-the-shelf behavioural navigation policy DECISION [1]

with the ability to handle failures spanning different scenarios, both inside and outside their training distributions.

Chapter 2

Literature Review

2.1 Learned visual navigation

Visual navigation involves navigating with rich, high-dimensional image inputs. Classical robot systems need complex handcrafted pipelines to handle such inputs and extract relevant geometric information from them [4]. In contrast, learning visual navigation avoids manual algorithm design and better captures semantic information from images [1]. Prior works explore learning collision predictors [24] or traversability maps [13] which are then combined with classical planners for navigation, or end-to-end policies that directly map visual inputs to control [1, 26].

Classical robot systems' explicit, interpretable sensory representations and states enable detection of certain types of navigation failures - *e.g.* inspecting a robot's map and pose lets us detect if it has gotten stuck in a corner and cannot reach its goal. This is challenging with the black box latent representations of many learned policies, motivating our approach to better structure the latent space and equip it with a meaningful distance metric that provides information on the policy's uncertainty about completing a task. In particular, we focus on end-to-end learned policies that are highly opaque owing to their lack of modularity. We apply *FaRe* to one of the latest work DECISION [1], a state-of-the-art behavioural navigation policy with a complex architecture that handles multimodality and temporal information.

2.2 Failure detection with learned policies

Detecting the failure of learned policies often comes down to quantifying the uncertainty of the policy in its ability to complete the task [14, 24]. Various approaches explored range from joint supervised learning of value functions together with policies [14], to unsupervised anomaly or out-of-distribution detection. The latter is a widely used method, which aims to identify if sensor inputs are drawn from a distribution different from that of the training data. This can be achieved by modelling the training distribution with generative models like normalising flows [28, 6] and using their likelihoods as a metric, or by learning autoencoders and thresholding on their reconstruction errors [24, 29]. In many of these works, the detectors are trained apart from the policy using objectives unrelated to navigation. Thus they may not accurately capture anomalies under the distribution learned by the policy and may detect task-irrelevant anomalies like visual differences from the training data that have no bearing on navigation [21]. Some other works attempt to design policies that can estimate the uncertainty of their predictions using ensemble methods [12] and Bayesian neural networks [21]. However, these methods can be computationally expensive to run, and are meant to quantify variance in training data rather than accurately estimating uncertainty for inputs far from the training data [24].

FaRe is designed to enable lightweight and efficient detection of navigation-related anomalies, without requiring any additional supervision apart from the data used for learning policies with Imitation Learning (IL). In particular, *FaRe* detects anomalies in the policy's latent space, which is learned through IL and captures navigation-related features, making detected anomalies likely to be failures related to navigation. Detecting anomalies in the latent space without requiring additional decoders [18] or other detector networks [24, 29] ensures that *FaRe* is lightweight and practical for real-world onboard usage.

2.3 Failure handling with learned policies

In addition to detecting failures, several works also consider how to handle failures. Among these, the common approach is to terminate the episode and

possibly request human intervention [14, 16, 21]. Other works consider how to take corrective actions to prevent or recover the robot from a failure, such as by falling back to a predefined "safe" prior behaviour [24] or returning to a default position [29]. Notably, select systems combine both corrective action and termination followed by interventions: MoMaRT first attempts recovery then requests human intervention if needed, while [11] performs online learning of recovery policies based on demonstrations accumulated from requested human interventions.

FaRe is a comprehensive approach to failure-handling that combines autonomous corrective actions with requests for human intervention as a last resort. In contrast to existing works, its corrective actions are not simply predefined default "safe" behaviours, but are intelligent recovery strategies that attempt to prevent or recover the robot from its failure state. In particular, we implement backtracking and local perturbation behaviours which use anomaly detection and localisation information to guide the robot away from identified anomalies in a predictable manner.

Chapter 3

Method

3.1 Problem formulation

We consider the problem of detecting and handling navigation failures induced by learned visual navigation policies. Similar to prior works, we cast this as a problem of detecting and handling anomalous inputs that lie outside the training data. Specifically, we seek to build a system that uses learned policies for visual navigation until an anomalous situation is detected. For example, a robot equipped with DECISION [1] experiences a sensor malfunction, encounters a dead end where no traversable path exists in the observation, or a dynamic object suddenly appears very close to the robot. Since learned policies may yield unreliable predictions that can cascade into navigation failures in such situations, the system should instead execute recovery strategies that can prevent the robot from entering, or extricate the robot from the anomalous situation. If such strategies fail to address the anomalous situation, the system should terminate the navigation episode and seek manual intervention from humans.

Our system should be able to augment existing learned visual navigation policies with the ability to detect and handle anomalous situations. For a given navigation policy π at timestep t, we seek to augment it to $\hat{\pi}(o_t) = (a_t, D_t, L_t)$, where o_t is the RGB observation input, a_t is the control command output, $D_t \in \{0, 1\}$ is an indicator describing the occurrence of an anomaly, and L_t is a heatmap showing where the corresponding anomaly is located in the current observation. We desire a system Π containing $\hat{\pi}$ and a set of recovery strategies R, such that $\Pi(o_t) = a_t$

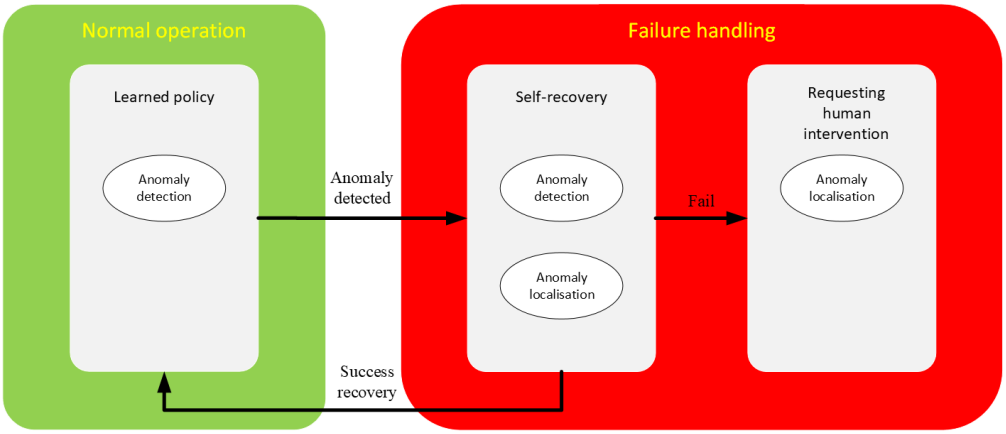


Figure 3.1: Overview of our failure-resilience framework

if input is not anomalous to $\hat{\pi}$ (*i.e.* $D_t = 0$), and $\Pi(o_t) = R(L_t) = a'_t$ if input is anomalous (*i.e.* $D_t = 1$). If the recovery strategies R do not succeed and an anomalous situation continues to be detected, the robot should terminate the episode and seek human intervention.

3.2 System overview

Our objective is to develop a framework for failure resilience with learned visual navigation policies π. The system Π comprises two parts: normal operation with augmented learned policy $\hat{\pi}$ and failure handling to address abnormal situations where the learned policy might fail. To detect and handle task-relevant anomalies effectively, we embed two key components, anomaly detection and localisation into the learned policy by regularising the policy with a variational information bottleneck, without requiring additional supervision.

In normal operation, $\hat{\pi}$ will concurrently yield an anomaly score each steps while navigating. This enables the robot to transition seamlessly to failure handling mode upon encountering anomalous states. In the failure handling phase, robot will initially attempt self-recovery, executing recovery strategies R guided by anomaly localisation to navigate out of anomalous situations. The anomaly score serves as a metric for the success of recovery strategies. Robot will switch back to normal operation upon successful recovery or eventually request human intervention if the recovery remains unsuccessful after a predefined number of attempts.

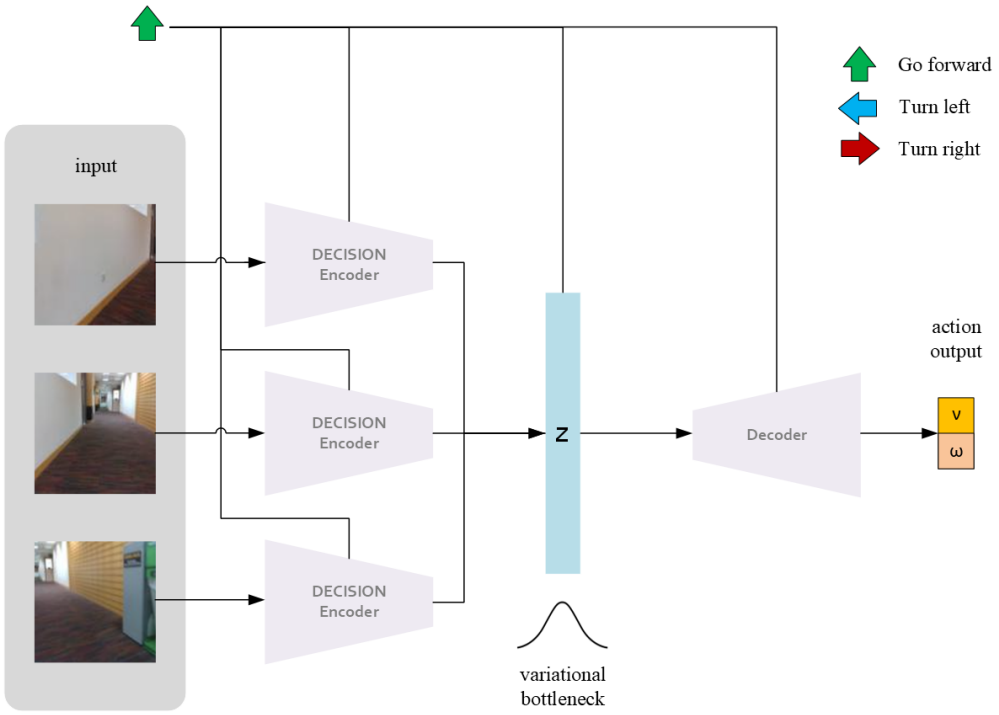


Figure 3.2: The neural network architecture of *FaRe*-DEC

3.3 Anomaly detection

We use anomaly detection to identify situations potentially leading to navigation failures. The explicit detection of anomaly is crucial for switching from normal operation to failure handling and measuring the success of self-recovery. Our approach involves structuring the latent representation of visual inputs by slightly modifying the network architecture of a learned policy, to create a latent space with meaningful distances that augments the policy with the ability to classify normal inputs and anomalies. Similar approach has been used in classification task [2]. In this way, we embed the anomaly detection into a learned policy that ensures the detector is learning the same data distribution as the policy, *i.e.* task relevant, and does not add too much computational cost compared with building detectors external to the learned policy.

The latent space is formed by regularising the policy using a variational information bottleneck (VIB) [19] objective, which use Kullback–Leibler (KL) divergence D_{KL} to constrain the variational posterior to approximate a prior distribution during training. Given a learned policy $\pi(a|o)$ with input image observations o, we regularise

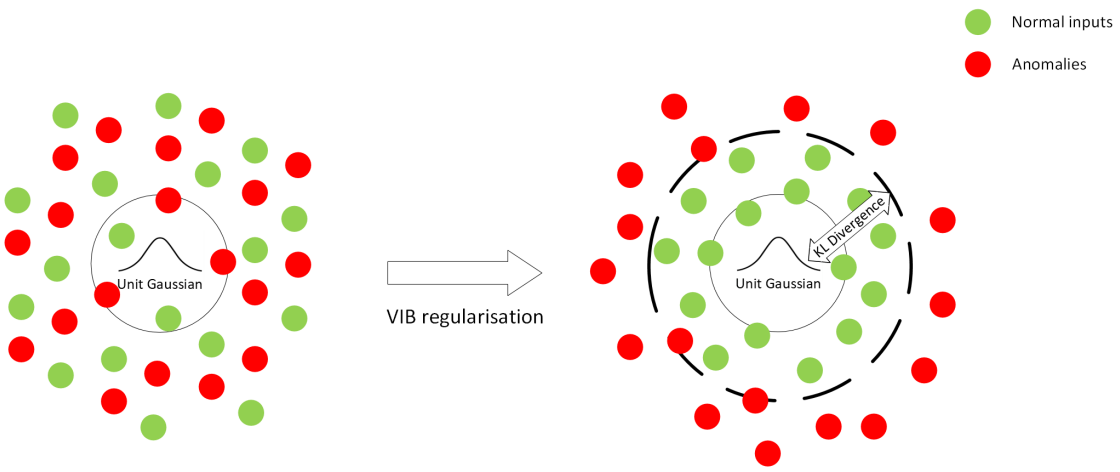


Figure 3.3: Visual explanation of our VIB heuristic. After adding VIB regularisation, a structured latent space is formed where we can apply a decision boundary measured by KL divergence to classify normal inputs and anomalies

it to $\hat{\pi}(a|o)$ by inserting a VIB after the convolutional feature extraction backbone of $\pi(a|o)$ to achieve task-relevant anomaly detection. We train the new policy $\hat{\pi}(a|o)$ in β-VAE [17] form. We denote the regularised latent space as z, encoder of $\hat{\pi}(a|o)$ as $p(z|o)$, the decoder of $\hat{\pi}(a|o)$ as $q(a|z)$, the loss for training the original policy $\pi(a|o)$ as $\mathcal{L}_{policy}$. The KL regularisation constrains the variational posterior $p(z|o)$ to approximate an isotropic unit Gaussian distribution prior $\mathcal{N}(0,1)$ during training:

$$\mathcal{L} = \mathcal{L}_{policy} + \beta D_{KL}(p(z|o) \parallel \mathcal{N}(0,1)) \tag{3.1}$$

Consequently, inputs lying inside the training distribution may be projected closer to the Gaussian prior while anomalous inputs deviating from the training distribution may be projected further, which can be measured by D_{KL} in this newly structured latent space z. A visual explanation 3.3 regarding the VIB regularisation is provided. Heuristically, anomalies may have a higher D_{KL} values. We leverage this property to use D_{KL} directly as our anomaly score. Moreover, inputs that have a relatively high D_{KL} represent that they are task-relevant anomalies for valid actions generation, rather than naive anomalies based on image pixel space.

The total change of filtered anomaly score over a fixed window length is used as anomaly detection criterion during deployment. At each timestep t the augmented policy $\hat{\pi}$ will yield an anomaly score $d_t = D_{KL}$ together with action output a_t in normal operation. We pass d_t through a Kalman filter initialised with identity matrix

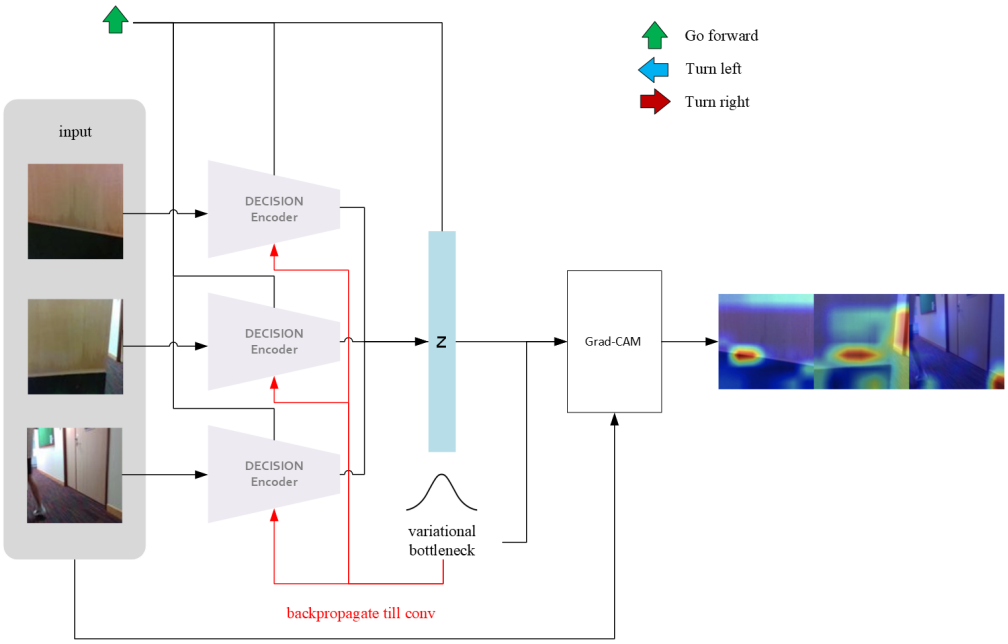


Figure 3.4: Anomaly localisation with *FaRe*-DEC

to get filtered anomaly score $\hat{d}_t$ and set the total change of $\hat{d}_t$ over a fixed window length t_d, $\hat{d}_t - \hat{d}_{t-t_d}$, as the anomaly detection criterion. The detection threshold ϵ and t_d are set as hyperparameters which can be tuned to balance sensitivity and robustness. This criterion is better than applying a hard threshold on the anomaly score since our VIB regularisation operates as a heuristic that lacks a strict unit Gaussian constraint. Consequently, the D_{KL} values, which distinguish anomalies from normal inputs, may exhibit slight variations across different environments. Our difference-based criterion is designed to adapt to these variations, providing a more robust threshold. Our design is based on the assumption that robot always start in a normal condition with $D_{t=0} = 0$ and environment will change smoothly during normal navigation process.

$$\text{if } D_{t-1} = 0, D_t = \begin{cases} 0 & \text{if } \hat{d}_t - \hat{d}_{t-t_d} \leq \epsilon \\ 1 & \text{if } t \geq t_d, \hat{d}_t - \hat{d}_{t-t_d} > \epsilon \end{cases} \tag{3.2}$$

3.4 Anomaly localisation

Merely detecting anomalies is insufficient for ensuring failure resilience. It is essential for robot to pinpoint the locations in current observations causing potential policy failures. Therefore, we incorporate anomaly localisation to perform a more predictable, efficient and safer failure handling.

Since the whole structure of our regularised learned policy is differentiable 3.2, if an anomaly is detected, *i.e.* $D_t = 1$, at each timestep we can backpropagate the high anomaly score d_t to the last convolutional layer and apply Grad-CAM[25] to obtain a heatmap L_t representing anomaly localisation.

Specifically, after the forward pass and backpropagation we can obtain feature map activations of the convolutional layer $A \in \mathbb{R}^{n \times h \times w}$ and gradients of anomaly score with respect to the activations $\frac{\partial d_t}{\partial A^k}$, where A^k is the k^{th} feature channel of A. Then we compute neuron importance weights α_k by global average pooling:

$$\alpha_k = \frac{1}{Z} \sum_{i=1}^{h} \sum_{j=1}^{w} \frac{\partial d_t}{\partial A_{ij}^k} \tag{3.3}$$

where $Z = h \times w$. This α_k captures the importance of feature channel k for our anomaly score. Finally we perform a weighted combination of A^k and apply ReLU to it since we are only interested in the features that have a positive influence on d_t. After scaling to the same spatial size as the input, we get a heatmap L_t that represents task-relevant anomaly localisation.

$$L_t = Scaling(ReLU(\sum_{k=1}^{n} \alpha_k A^k)) \tag{3.4}$$

3.5 Failure handling

The failure handling aspect of *FaRe* is designed to address abnormal situations where the learned policy might fail. It is comprehensive and encompasses two phases, self-recovery and requesting human intervention, guided by the information provided by our anomaly detection and localisation components.

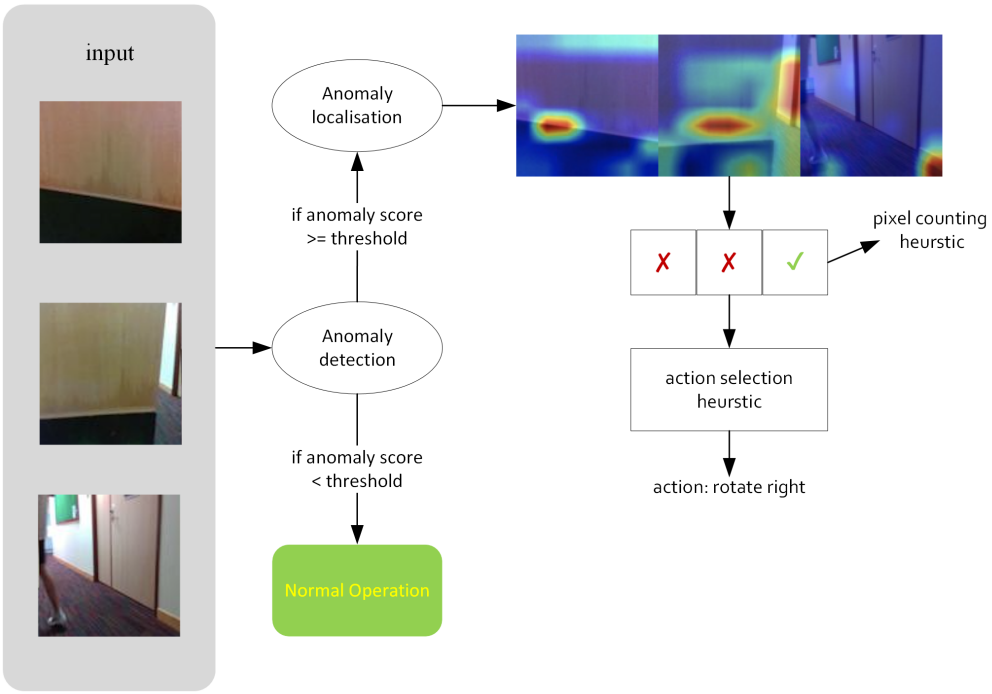


Figure 3.5: Self-recovery module

3.5.1 Self-recovery

Autonomous prevention or recovery from failure states is the initial objective after encountering anomalies ($D_t = 1$). Our self-recovery strategy R aims to avoid anomalous regions in current observation o_t that might cause navigation failure given L_t. We divide the anomaly localisation heatmap horizontally into three equal-sized bins, and measure the anomaly belief of each bins b_{al}, b_{am}, b_{ar} through a pixel counting heuristic. L_t is converted into a binary image using Otsu's method [23] and we count the number of highlighted pixels of each bins c_{al}, c_{am}, c_{ar}. If the number is above a threshold a, the corresponding bin is marked as anomalous.

$$b_{ap} = \begin{cases} \frac{c_{ap}}{a} & \text{if } c_{ap} \leq a \\ 1 & \text{if } c_{ap} > a \end{cases}, p \in [l, m, r] \tag{3.5}$$

An action selection heuristic is employed based on the anomaly belief of each region, determining the appropriate recovery action from predefined options: *rotate left*, *rotate right*, *stepping*, and *backtrack*. Action *rotate left*, *rotate right* and *stepping* are hard-coded and angularly constrained to perform perturbation without affecting the high-level task. Action *backtrack* is implemented with a fixed length action cache

Algorithm 1: Recovery strategies $R(L_t)$.

Input: anomaly localisation heatmap: L_t
Output: recovery_action
Function `R(`L_t`):`
  b_l, b_m, $b_r \leftarrow$ pixel counting heuristic(L_t)
  if $b_l = b_m = b_r = 1$ **then**
    recovery_action $\leftarrow$ *backtrack*
  else if $\min(b_l, b_m, b_r) = b_l$ **then**
    recovery_action $\leftarrow$ *rotate left*
  else if $\min(b_l, b_m, b_r) = b_r$ **then**
    recovery_action $\leftarrow$ *rotate right*
  else
    recovery_action $\leftarrow$ *stepping*
  return *recovery_action*

that records past t_a steps of action. The detail of our action selection heuristic is shown in Algorithm 1

During self-recovery, the anomaly score d_t is continuously monitored each timestep to measure the success of R. In normal operation, the robot records past t_d steps of anomaly score. When an anomaly is detected, given our anomaly detection criterion it represents that our robot gets into an anomalous state from normal. Thus, the first t_n steps of recorded anomaly scores are able to stand for the anomaly score value in normal state. Our success recovery criterion is when d_t drops back to the value level observed in normal operation. Suppose the robot detects an anomaly at timestep t_0, the transition is stated as 3.6. In this way, *FaRe* is able to execute a series of intelligent self-recovery actions in an informed and predictable way while encountering anomalies.

$$\text{if } D_{t-1} = 1, D_t = \begin{cases} 0 & \text{if } d_t < \frac{\sum_{p=t_0-t_d}^{t_0-t_d+t_n} d_p}{t_n} \\ 1 & \text{if } d_t \geq \frac{\sum_{p=t_0-t_d}^{t_0-t_d+t_n} d_p}{t_n} \end{cases} \tag{3.6}$$

3.5.2 Requesting human intervention

If the robot fails to self-recover within T steps or *backtrack* action exceed t_a steps, *FaRe* will terminate the self-recovery and request human intervention. Human experts will be informed with the current observation, anomaly localisation heatmap and anomaly beliefs to take over the irrecoverable situation or improve the system.

Chapter 4

Experiments

We seek to answer the following questions: (1) How well can *FaRe* detect and localise anomalies? (2) Do *FaRe*'s design choices enable more efficient and effective detection and localisation of navigation-related anomalies than prior methods? (3) How does *FaRe*'s VIB regularisation affect performance of a learned visual navigation policy? (4) Can *FaRe* enable a learned policy to robustly detect and handle navigation failures?

4.1 Experimental setup

We run our experiments on a Boston Dynamics Spot robot equipped with 3 Intel RealSense D435i cameras with a combined FoV of 140°, with all software running onboard an Nvidia Orin AGX. Our tests focus on augmenting a representative, off-the-shelf learned policy DECISION [1] with *FaRe*, *i.e.* *FaRe*-DEC. DECISION is a behavioural navigation policy which has an architecture to capture multimodal behaviours with temporal information. We choose it to demonstrate that *FaRe* is effective even on models with recurrent elements and a complicated structure.

4.2 Taxonomy of failures

We first introduce a taxonomy of failures 4.1 and a specific illustration of our training data 4.2 to provide a clearer understanding for evaluating our method.

Failures are categorised into 1) failures of input, and 2) failures of learned policy. The latter is further divided into failures attributable to inductive bias of the policy model and those stemming from training data. Our experiments focus

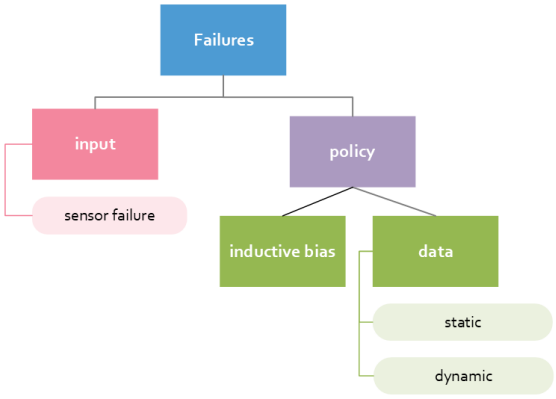


Figure 4.1: Taxonomy of failures for learned visual navigation policies

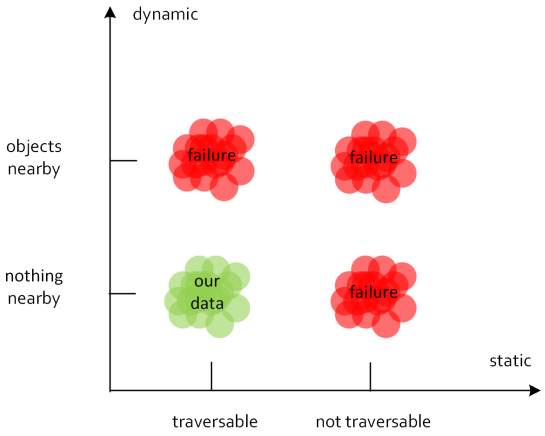


Figure 4.2: Taxonomy of failures in terms of data

on the input and data aspects in the taxonomy without taking inductive bias into consideration, since failures due to inductive biases are rare in our navigation setting. For instance, the specified number of time steps for the ConvLSTM backbone within the DECISION policy is an inductive bias. However, this setting is unlikely to lead to failures because it has been set large enough to accommodate the necessary historical information required by the policy to handle partial observability in navigation tasks.

Input failures in visual navigation correspond to sensor malfunctions, for example, camera dead or extreme lighting conditions. For the data aspect, we analyse the characteristics of training data for learned visual navigation policy along two axis, static and dynamic. As depicted in 4.2, along the static axis from left to right

are observations contain traversable path to do not contain traversable path, along the dynamic axis from bottom to top are observations do not contain dynamic objects nearby the robot to contain objects nearby. In order to train a behavioural policy conditioned on high level intentions capable of tracking traversable paths and avoiding obstacles in advance, all the data that have been collected for DECISION training adhere to the rule of including traversable paths while maintaining a clear space around the robot. Then all the other areas in this static-dynamic plain represent potential failures given the training dataset.

4.3 Anomaly detection and localisation performance

We aim at answering question (1) and (2) in this section. We collect 120 trajectories featuring different failure cases in various environment to evaluate our method and two baseline approaches offline. Given our taxonomy, there are 40 failure cases of each of the three types: not traversable, traversable but have dynamic objects nearby, and sensor failure. For each type, we collect 20 trajectories from within the training distribution and 20 from outside. Each trajectory comprises approximately 8 seconds of normal data followed by about 2 seconds of the corresponding type of failure data with a frequency of 10 Hz. We record the initial time frame t_1 when a potential navigation failure occurs, and the binary anomaly status (anomalous or normal) of the three bins of observation (as described in the last chapter) conceptually as the ground truth.

We utilise these trajectories to quantify the false positive (fp), true positive (tp) and false negative (fn) results for anomaly detection and localisation. For anomaly detection, if an anomaly is detected before t_1, we label the trajectory as fp. If an anomaly is detected after t_1, we label the trajectory as tp. If no anomaly is detected until the end of the trajectory, we label the trajectory as fn. For anomaly localisation, if the ground truth of a bin is anomalous and the predicted belief is 1, we label this bin as tp. If ground truth is anomalous but the predicted belief is less than 1, we label this bin as fp. If ground truth is normal, the predicted belief is 1,

Figure 4.3: Sampled failure cases. The left column of images are failure cases occurred inside training environments, the right column are outside. From top to bottom are three failure types, not traversable, traversable but have dynamic objects nearby, and sensor failure.

we label this bin as fn. Precision and recall metrics are then calculated as follows:

$$Precision = \frac{tp}{tp + fp} \quad Recall = \frac{tp}{tp + fn} \tag{4.1}$$

Our baseline methods include:

- **VAE(Recon)**: A Variational Auto Encoder [19] model trained with image reconstruction loss and KL divergence regularisation aside from the learned policy. Anomaly score is derived from image reconstruction error, and anomaly localization heatmap is based on pixel error map.
- **VAE(KL)**: Utilises the same VAE model as VAE(Recon) but employs KL divergence as the anomaly score. Anomaly localization is carried out similarly to *FaRe*-DEC.

We select VAE(Recon) as a representative of a variety of anomaly detection and localisation methods leveraging reconstruction-based generative models [24, 29]. Additionally, we include VAE(KL), which employs the same anomaly detection and gradient-based localisation method as *FaRe*-DEC, to assess the efficacy of detectors and localisers external to the learned policy in identifying task-relevant anomalies compared to our embedded approach.

Table 4.1: Performance of anomaly detection and localisation

Method		InD		OoD		Add. params	Add. execution time
		Precision	Recall	Precision	Recall		
VAE(Recon)	Det	0.31	0.41	0.36	0.30	123.46M	0.016
	Loc	0.71	0.49	0.69	0.40	123.46M	**0.020**
VAE(KL)	Det	0.74	0.60	0.69	0.51	123.46M	0.016
	Loc	0.87	0.47	0.83	0.41	123.46M	0.071
FaRe-DEC	Det	**0.93**	**0.74**	**0.86**	**0.69**	**6.16M**	**0.006**
	Loc	**0.91**	**0.57**	**0.86**	**0.48**	**6.16M**	0.049

All models, including our method and the baselines, are trained on the same dataset in β-VAE form. For the baseline VAE model, we adopt ResNet50 [15] as the encoder and a series of deconvolutional layers as the decoder. For all models, the dimension of the variational bottleneck to set to 1024, with a balancing factor β=1e-6. The hyperparameters are tuned to simultaneously uphold satisfactory control performance for *FaRe*-DEC while ensuring effective reconstruction of image observations for VAE(Recon) and VAE(KL). One trick for *FaRe* training is to make the policy loss converge to a similar value as the original policy training. All the other training details are consistent with those outlined in the DECISION paper.

We compare four metrics, precision, recall, additional parameters and additional execution time (s) of all methods for anomaly detection and localisation respectively. The last two metrics mean that how many additional parameters or execution time is needed to perform anomaly detection or localisation. All tests are done in one RTX 2080Ti GPU.

As shown in the table 4.1, our *FaRe*-DEC achieve the best performance in terms of navigation related failure detection and localisation both inside and outside training distribution. We observe that for VAE(Recon), the false positive rate is extremely high because of detecting task-irrelevant anomalies, for instance, a fire extinguisher at one side of the corridor. And the reconstruction method can reconstruct navigation related anomalies that have simple visual feature well, for example a plain wall ahead of the robot, resulting in false negative. VAE(KL) perform generally better than VAE(Recon), which indicates that KL divergence could be a better metric than

Table 4.2: Performance of anomaly detection and localisation with different β factor

Method		InD		OoD	
		Precision	Recall	Precision	Recall
$\beta = 1$	Det	0.14	0.25	0.06	0.08
	Loc	0.49	0.20	0.47	0.18
$\beta = 1e-3$	Det	0.89	0.71	0.86	0.52
	Loc	0.87	0.51	0.82	0.38
$\beta = 1e-6$	Det	**0.93**	**0.74**	**0.86**	**0.69**
	Loc	**0.91**	**0.57**	**0.86**	**0.48**
$\beta = 1e-9$	Det	0.13	0.19	0.07	0.17
	Loc	0.57	0.38	0.59	0.47

reconstruction error in navigation context. Furthermore, *FaRe*-DEC requires the least additional parameters and is the most efficient method for anomaly detection, which is required for every timestep. Although it is more time consuming compared with using pixel error to localise anomalies since *FaRe*-DEC needs backpropagation, anomaly localisation is not as frequently used as detection and is still able to satisfy the real-time requirement in real world deployment.

The experiments prove that *FaRe* is efficient and is able to detect and localise task-relevant anomalies effectively. Our *FaRe*-DEC outperforms the baselines in terms of both detection and localization, exhibiting superior precision and recall while requiring fewer additional parameters and little additional execution time.

We conduct another ablation study to show how the regularisation factor β affects the anomaly detection and localisation performance in table 4.2. We train our *FaRe*-DEC model with different values of β, and compute their precision and recall rate on the offline dataset. Results show that when $\beta = 1e-9$, the effect on the latent representation is barely noticeable, as the regularisation factor is too small. When $\beta = 1e-6$, *FaRe*-DEC achieves the best performance, and $\beta = 1e-3$ shows a comparable result. If the regularisation is too strong, *i.e.* $\beta = 1$, the performance becomes much worse due to posterior collapse, which is commonly observed in previous literature [9, 5, 27, 30].

Table 4.3: Performance of DECISION vs *FaRe*-DEC(policy)

Method	Task i		Task ii		Task iii
	SR	CR	SR	CR	No. of human interventions
DECISION	90	98.8	90	98	6
FaRe-DEC (policy)	80	98	80	96	8

4.4 Influence of VIB regularisation on policy

We aim at answering question (3) in this section. In these experiments we **disable** the anomaly detection and failure handling part of *FaRe*-DEC in order to compare only the performance of learned policy before and after VIB regularisation. Three tasks are designed to evaluate the ability of DECISION and *FaRe*-DEC(policy):

- **Task i**: Adversarial pedestrian avoidance. All settings are the same as in the DECISION paper [1].
- **Task ii**: Blind-spot object avoidance. All settings are the same as in the DECISION paper.
- **Task iii**: Long range navigation, where the robot is tasked with covering a distance of approximately 600 meters outdoor.

We compute the same metrics as in the DECISION paper for Task i and Task ii, success rate (SR %) and completion rate (CR %), while Task iii is quantified based on the number of human interventions required.

From the results 4.3, we can see that *FaRe*-DEC(policy)'s performance is slightly worse than DECISION, indicating that our regularisation method may impact policy performance to some extent. However, it is important to note that this test solely aims to examine the impact of VIB on policy performance without incorporating failure detection and handling. While the observed performance degradation is minimal and unlikely to be noticeable during real world deployment, the subsequent section demonstrates that *FaRe*-DEC is proficient in detecting and handling various failures that DECISION may struggle to address effectively and efficiently.

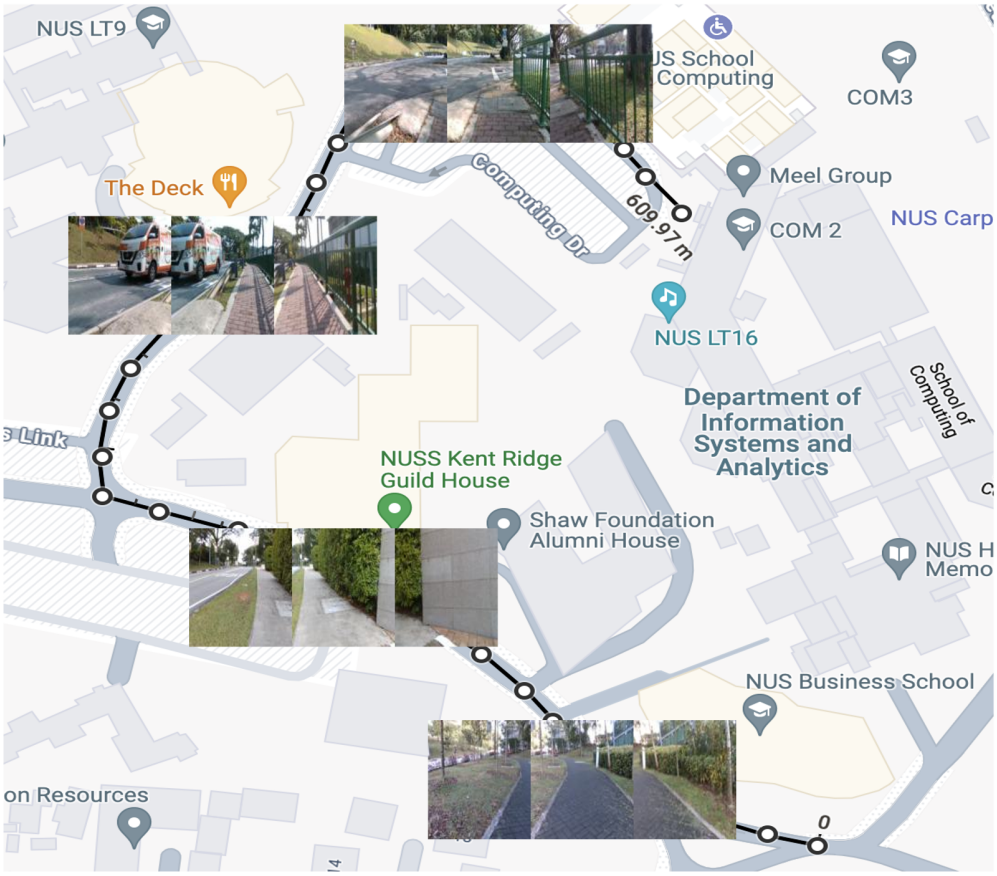


Figure 4.4: The navigation route of Task iii

4.5 Navigation failure detection and handling performance

We aim at answering question (4) in this section. Most importantly, we want to show how robustly and efficiently can *FaRe*-DEC detect and handle navigation failures in the real world. For quantitative analysis, similar to question (1) we choose three types of failures from taxonomy to ensure our experiments have a decent coverage of failure situations. More than question one, we give each type of failures a label of recoverable or irrecoverable, for better evaluation of failure handling performance. For each label of each type of failures, we pick one scenario and repeat it 20 times, recording the success rate (SR %) of failure detecting, failure handling and handling time (s). Successful detection is defined as detect the anomaly within 2 seconds after a potential failure occurs without collision. Successful handling for

recoverable failures is defined as prevent or recover from a navigation failure and complete the original task without collision. Successful handling for irrecoverable failures is defined as eventually requesting human intervention without collision. Furthermore, when evaluating failure handling, we only focus on successful detection cases in order to isolate the performance of the failure handling from that of the failure detecting. If the failure is not detected in one run, we discard it and restart. The discarded runs are not counted in the total number of executed runs. We design a 10-meter test route for each type of failures, where the robot initiates from the starting point and endeavors to reach the goal point. Along the navigation task, our robot encounters the following failures listed below.

- **Sensor failure**: Sensor failures are irrecoverable. We simulate the sensor failure with all zero inputs.
- **Not traversable**: An observation that does not contain traversable path may be either recoverable or irrecoverable. For recoverable cases, we choose to place an obstacle 0.2m in the direction of robot's heading to block its view of any traversable area while navigating and stay still. Ideally, our robot could detect the failure, avoid the obstacle and return to the path using a combination of backtracking and perturbation manoeuvres without collision. For irrecoverable cases, we choose to drive the robot towards a closed door. Ideally, our robot could request human intervention without collision.
- **Traversable but have dynamic objects nearby**: A observation that contains dynamic objects nearby is recoverable since there is still traversable path in robot's observation. We choose to place an obstacle 0.2m in the direction of robot's heading to block half of the robot's view. Ideally, our robot could detect the failure, avoid the obstacle and return to the path by perturbation.

The experiment results 4.4 show high failure detecting and handling success rate and reasonably low time to handle several types of failures that DECISION encounters.

The typical examples of success failure detection and handling for each type of failures are presented in Figure 4.5. The regular RGB image indicates normal

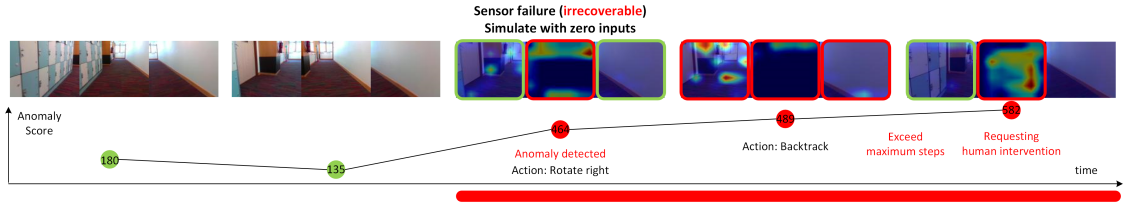
Table 4.4: Real world performance of *FaRe*-DEC

Test cases		Detecting SR	Handling SR	Handling Time
Sensor failure (irrecoverable)		100	100	18.35
Not traversable	recoverable	90	80	5.10
	irrecoverable	100	90	14.78
Trav. but objects nearby (recoverable)		85	85	11.11

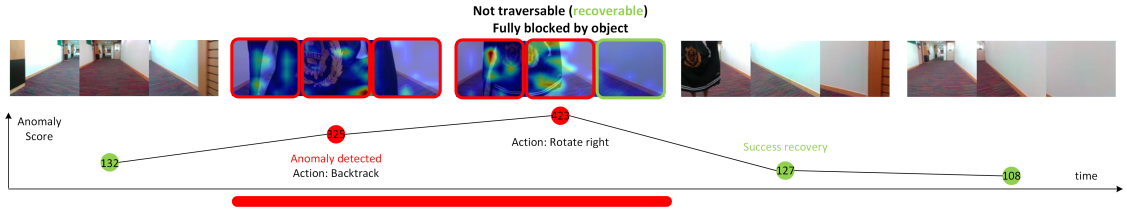
operation, whereas the heatmap image, representing anomaly localization, indicates failure handling. The corresponding recovery strategies generated from heatmap are also listed in the figure. The first 4.5(a) and third row 4.5(c) depict two irrecoverable failure cases: sensor failure and encountering a closed door ahead of the robot. *FaRe*-DEC successfully detects the failures and resorts to requesting human intervention, as neither self-recovery action can restore the robot to normal operation. The second and 4.5(b) fourth row 4.5(d) illustrate two recoverable failure cases: the robot's view is obstructed by objects, fully blocking any traversable area, and the observation contains traversable areas but with dynamic objects nearby. *FaRe*-DEC successfully detects the failures and self-recover through a series of backtracking or perturbation strategies, generated from heatmaps which highlight the task-relevant anomalous regions in the observation. Our robot finally returns to normal operation and reach the navigation goal point without collision. The typical failure scenario in our failure handling arises when *FaRe*-DEC switches back to normal operation but still fails to entirely avoid the anomalies, potentially resulting in collisions at the robot's periphery.

We have also conducted another qualitative experiment, to mimic failures that may frequently happen in real navigation scenarios. We can observe that *FaRe*-DEC is able to help with robot visual navigation in real scenarios 1.1.

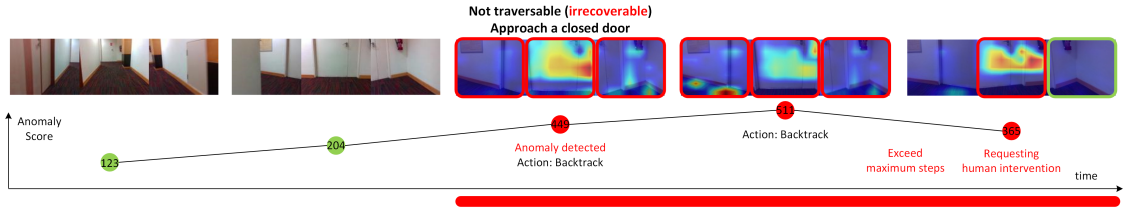
In conclusion, *FaRe*-DEC has the ability to detect and handle various failures in the real world robustly and efficiently. Moreover, *FaRe*-DEC enables more robust navigation by successfully recovering from failures along the way in real navigation scenarios.



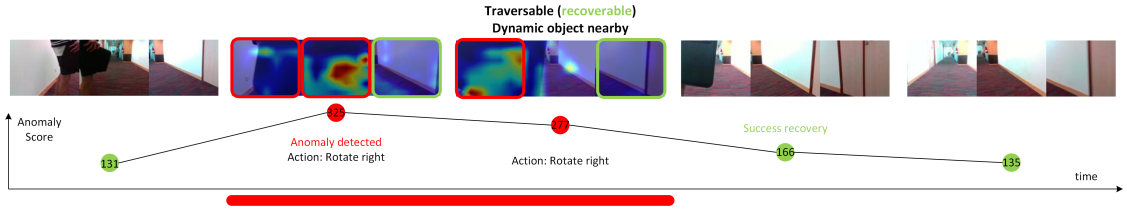
(a) Sensor failure (irrecoverable)



(b) Not traversable (recoverable)



(c) Not traversable (irrecoverable)



(d) Trav. but objects nearby (recoverable)

Figure 4.5: Qualitative analysis of *FaRe*-DEC. From top to bottom are the four real world test cases corresponding to Table 4.4, arranged in the same order. From left to right are five time steps down-sampled from recorded real-world experimental result. Robot observation, anomaly score, anomaly localisation result, system status and recovery strategy of each step are presented. In the anomaly localization results, bins where the anomaly belief equals one are highlighted with a red circle, indicating them as anomalous. If not all bins are anomalous, the bin with the minimum anomaly belief is highlighted with a green circle.

Chapter 5

Conclusion

In this work we propose *FaRe*, a framework for failure-resilience with learned visual navigation policies and demonstrate its performance on *FaRe*-DEC. Through extensive ablation study and real world experiments, we have shown that our unsupervised, embedded anomaly detector and localiser excels at identifying navigation-related failures. Moreover, *FaRe* efffectively enables DECISION to robustly and efficiently detect and handle real world navigation failures. Our work introduces a novel system design paradigm with learned policy for visual navigation, which also opens up several avenues for future research into failure resilience. For example, while the current approach is unsupervised and the failure concept for learned policies rely on the training dataset, extending it to combine human ground truth feedback to update the failure concept could be a good way to enhance human-robot interaction and fortify failure resilience. Additionally, extending the core idea of this framework to encompass the latest imitation learning algorithms, such as diffusion policy [10], holds promise for further advancements.

A potential improvement of our work is to replace the pixel counting heuristic in failure handling with a lightweight, learnable classifier as shown in figure 5.1. In this way, the recovery strategy can be chosen more desirably, even under the circumstances where the heatmap is slightly noisy. Since this is a mapping from grey-scale heatmap to binary classification result, it could be learned with little data labelling and generalised to different failure situations compared to directly learning a mapping from raw observation to control, without losing the anomaly information from the original learned policy. Moreover, this classifier can be continuously updated and learned if the failure handling fails.

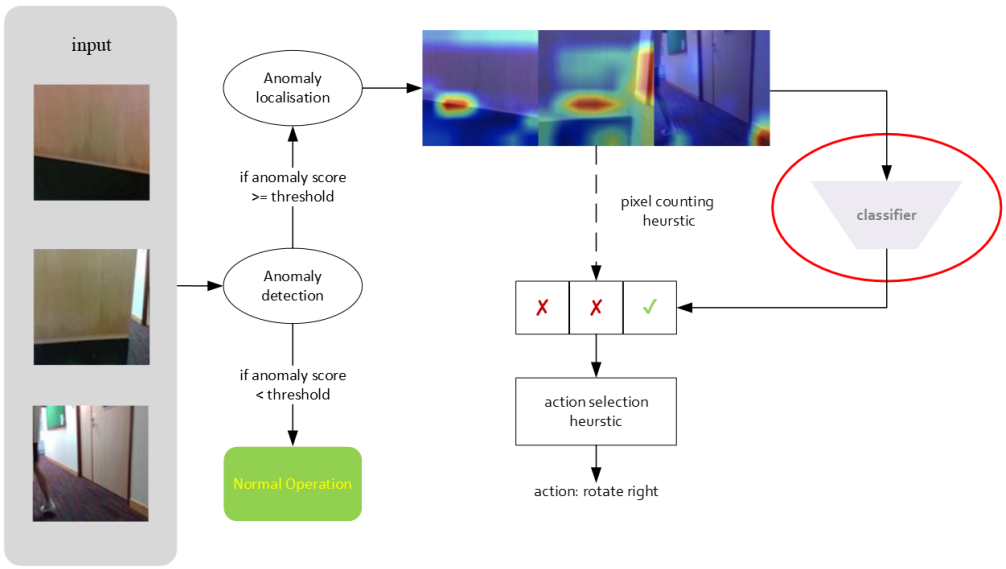


Figure 5.1: Potential improvement of our system

Our future plans also include extending this framework to accommodate a wide range of CNN-based learned policies. We aim to apply *FaRe* to policies with diverse structures and intended for various navigation tasks, including but not limited to the General Navigation Model (GNM) [26], which incorporates image goal conditioning for object-goal navigation.

Bibliography

[1] B. Ai, W. Gao, D. Hsu, *et al.*, "Deep visual navigation under partial observability", in *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022, pp. 9439–9446.

[2] A. A. Alemi, I. Fischer, and J. V. Dillon, "Uncertainty in the variational information bottleneck", *arXiv preprint arXiv:1807.00906*, 2018.

[3] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy, "Deep variational information bottleneck", in *International Conference on Learning Representations*, 2017.

[4] F. Bonin-Font, A. Ortiz, and G. Oliver, "Visual navigation for mobile robots: A survey", *Journal of Intelligent and Robotic Systems*, vol. 53, pp. 263–296, Nov. 2008.

[5] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space", *arXiv preprint arXiv:1511.06349*, 2015.

[6] X. Cai, S. Ancha, L. Sharma, P. R. Osteen, B. Bucher, S. Phillips, J. Wang, M. Everett, N. Roy, and J. P. How, "Evora: Deep evidential traversability learning for risk-aware off-road autonomy", *arXiv preprint arXiv:2311.06234*, 2023.

[7] O. Çatal, S. Leroux, C. De Boom, T. Verbelen, and B. Dhoedt, "Anomaly detection for autonomous guided vehicles using bayesian surprise", in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020, pp. 8148–8153.

[8] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey", 2019. arXiv: `1901.03407 [cs.LG]`.

[9] X. Chen, D. P. Kingma, T. Salimans, Y. Duan, P. Dhariwal, J. Schulman, I. Sutskever, and P. Abbeel, "Variational lossy autoencoder", *arXiv preprint arXiv:1611.02731*, 2016.

[10] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion", 2024. arXiv: 2303.04137 [cs.RO].

[11] F. Del Duchetto, A. Kucukyilmaz, L. Iocchi, and M. Hanheide, "Do not make the same mistakes again and again: Learning local recovery policies for navigation from human demonstrations", *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4084–4091, 2018.

[12] A. Filos, P. Tigkas, R. McAllister, N. Rhinehart, S. Levine, and Y. Gal, "Can autonomous vehicles identify, recover from, and adapt to distribution shifts?" In *International Conference on Machine Learning*, PMLR, 2020, pp. 3145–3153.

[13] J. Frey, M. Mattamala, N. Chebrolu, C. Cadena, M. Fallon, and M. Hutter, "Fast traversability estimation for wild visual navigation", in *Proceedings of Robotics: Science and Systems*, 2023.

[14] C. Gokmen, D. Ho, and M. Khansari, "Asking for help: Failure prediction in behavioral cloning through value approximation", in *2023 International Conference on Robotics and Automation (ICRA)*, IEEE, 2023, pp. 5821–5828.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition", *CoRR*, vol. abs/1512.03385, 2015. arXiv: 1512.03385. [Online]. Available: http://arxiv.org/abs/1512.03385.

[16] S. Hecker, D. Dai, and L. Van Gool, "Failure prediction for autonomous driving", in *2018 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2018, pp. 1792–1799.

[17] I. Higgins, L. Matthey, A. Pal, C. P. Burgess, X. Glorot, M. M. Botvinick, S. Mohamed, and A. Lerchner, "Beta-vae: Learning basic visual concepts with a constrained variational framework", in *International Conference on Learning Representations*, 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:46798026.

[18] T. Ji, A. N. Sivakumar, G. Chowdhary, and K. Driggs-Campbell, "Proactive anomaly detection for robot navigation with multi-sensor fusion", *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4975–4982, 2022.

[19] D. P. Kingma and M. Welling, "Auto-encoding variational bayes", 2022. arXiv: 1312.6114 [stat.ML].

[20] A. Loquercio, A. Kumar, and J. Malik, "Learning visual locomotion with cross-modal supervision", in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2023, pp. 7295–7302.

[21] R. McAllister, G. Kahn, J. Clune, and S. Levine, "Robustness to out-of-distribution inputs via task-aware generative uncertainty", in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 2083–2089.

[22] A. V. Nair, V. Pong, M. Dalal, S. Bahl, S. Lin, and S. Levine, "Visual reinforcement learning with imagined goals", in *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[23] N. Otsu, "A threshold selection method from gray-level histograms", *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[24] C. Richter and N. Roy, "Safe visual navigation via deep learning and novelty detection", in *Robotics: Science and Systems*, 2017.

[25] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization", *International Journal of Computer Vision*, vol. 128, no. 2, pp. 336–359, Oct. 2019, ISSN: 1573-1405.

[26] D. Shah, A. Sridhar, A. Bhorkar, N. Hirose, and S. Levine, "Gnm: A general navigation model to drive any robot", in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[27] A. Van Den Oord, O. Vinyals, *et al.*, "Neural discrete representation learning", *Advances in neural information processing systems*, vol. 30, 2017.

[28] L. Wellhausen, R. Ranftl, and M. Hutter, "Safe robot navigation via multimodal anomaly detection", *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1326–1333, 2020.

[29] J. Wong, A. Tung, A. Kurenkov, A. Mandlekar, L. Fei-Fei, S. Savarese, and R. Martín-Martín, "Error-aware imitation learning from teleoperation data for mobile manipulation", in *Conference on Robot Learning*, PMLR, 2022, pp. 1367–1378.

[30] T. Zhao, K. Lee, and M. Eskenazi, "Unsupervised discrete sentence representation learning for interpretable neural dialog generation", *arXiv preprint arXiv:1804.08069*, 2018.