Hazard Detection for Robotic Applications as Visual Anomaly Detection

Doctoral Dissertation submitted to the

Faculty of Informatics of the *Università della Svizzera italiana*

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

presented by

Dario Mantegazza

under the supervision of

Prof. Luca Maria Gambardella

co-supervised by

Prof. Alessandro Giusti

May 2024

Dissertation Committee

Prof. Kai Hormann	Università della Svizzera italiana, Switzerland
Prof. Paolo Tonella	Università della Svizzera italiana, Switzerland
Prof. Giacomo Boracchi	Politecnico di Milano, Italy
Dr. Simone Gasparini	INP Toulouse, France

Dissertation accepted on 3 May 2024

Prof. Luca Maria Gambardella
Research Advisor
Università della Svizzera italiana, Switzerland

Prof. Alessandro Giusti
Research Co-Advisor
Università della Svizzera italiana, Switzerland

Prof. Walter Binder
Prof. Walter Binder / Prof. Stefan Wolf

I certify that except where due acknowledgement has been given, the work presented in this thesis is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; and the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program.

Dario Mantegazza
Lugano, 3 May 2024

Alla mia nuova famiglia, Tatiana, Massimo e Caramella, e alla mia famiglia di origine Patrizia, Giorgio e Clara.

Acknowledgements

When I started my academic journey I would not ever guess that I would pursue a Ph.D. At that time, it was unfathomable for me to think that one would put himself under so many more years of study, nonetheless, here I am writing my doctoral thesis.

What I didn't know then was that a Ph.D. is not like a Bachelor's or a Master's; with a Ph.D. you have freedom, and you have the opportunity to explore and learn what you love and are passionate about. A Ph.D. is a lot of hard work, with highs and lows, long nights and early mornings, running toward deadlines but all of this effort leads and is leading to great opportunities. I built and strengthened my knowledge of AI and Robotics, met with impressive people from all over the world, and traveled to two other continents while participating in the communal effort that is scientific research. Now I'm starting a new business thanks to what I learned in these years.

This Ph.D. means more to me than only its academic value; in my Ph.D. I grew as a person, I learned to be more patient and focused, and I learned how to teach and speak in public. Most importantly, my Ph.D. coincided with exciting moments in my life during which I had the opportunity to form a new family.

For all of this, I have to thank those who helped me along this journey. Starting from my supervisors Alessandro Giusti and Luca Gambardella who first believed in me and gave me the opportunity to IDSIA, and guided me in my work. Thank you also to Jerome Guzzi for his valuable insights and precious support in many projects. Thanks to Omar, Daniele, Antonio, David, and others from IDSIA who listened, guided, and counseled me in these years. Thanks to Gulcan from Nomoko AG and Carlos from Hovering Solutions Ltd. for their collaboration. A heartfelt thanks to the friends I made in this journey: Arianna, Alberto, Elia, Luca, Stefano, and to those who are friends from even before: Andy, Samuele, Mirko, Gabri, Vale, and Francesco. Special thanks to my family, Clara, Patrizia, Giorgio, Jury, Ornella, Paolo, and my lovely wife Tatiana who supported me over these years and always believed in me.

I would like to express my gratitude to the initiatives that supported this Ph.D.: the Swiss National Science Foundation (SNSF) through the National Centre of Competence in Research (NCCR) Robotics, the Innosuisse agency, and the European Commission through the Horizon 2020 initiative. A final thanks to Giacomo Boracchi, Simone Gasparini, Kai Hormann, and Paolo Tonella for dedicating their time and effort as members of my evaluation committee.

Abstract

For a robot, a hazard is an event or object that poses risks to its mission or to itself. Some hazards such as obstacles are known, and can be accounted for; others, such as piercing debris or dense fog might be unexpected and may be seen only on some rare occasions. For these, collecting samples to train a perception model is often impossible. Thus, a hazard detection system should not require hazard samples to function correctly.

In this thesis, we propose to use deep learning-based visual anomaly detection models to solve hazard detection for mobile robots employed in industry. Our proposal of relying on visual anomaly detection is particularly suited for these robots since most of those have cameras.

Anomaly detection is a machine learning topic focused on finding rare, unexpected, patterns in data that deviate from an expected behavior. It can be applied to various fields and data types, but the application of anomaly detection in robotics is rather new and limited to specific use cases. Nonetheless, anomaly detection fits well with hazard detection as it requires datasets composed only of non-anomalous (i.e., expected, normal) samples.

No public datasets are available for the task of hazard detection for robotics. We start by closing this gap with our general-purpose visual hazard detection dataset for mobile robots. Then, we introduce a hazard detection system based on convolutional undercomplete autoencoders. Our approach detects multiple types of hazards using only images coming from the robot's front-facing camera. We test this solution using two real-world qualitative demonstrations with a wheeled robot in a lab, and an industrial drone in a factory, and detect all anomalies.

Based on the expectation that few anomalous samples will be collected during deployment, we experiment with an outlier exposure approach, to learn from these key anomalous samples. We employ a Real-NVP model, combined with a features extractor and a novel loss, to train using a few detected anomalies in addition to normal samples. Our experiments show that our solution effectively increases the detection performance for all anomalies, measured by the AUC, by 9.6%.

Similarly, we can expect that the data collected by the deployed robots becomes too much to be all manually inspected and labeled. We propose two novel active learning methods designed for anomaly detection using Real-NVP. We test our solutions against six other queries strategies from the literature, across more than 6500 experiments. We show that when multiple samples are collected, our approaches are best for choosing informative samples collected.

Lastly, we study how pre-trained feature extraction models perform on 3D anomaly detection tasks. Our results show that while our approaches are better than older models and baselines when data is scarce, ad hoc models outperform our proposed solution when enough data is available.

Contents

Contents **vii**

List of Figures **xiii**

List of Tables **xv**

1 Introduction **1**
1.1 Problem Definition 2
1.2 Contributions 3

2 Related Work **7**
2.1 Anomaly Detection 7
2.2 Visual Anomaly Detection 9
2.3 Models, Methods, and Applications 9
2.3.1 Reconstruction-based Methods 9
2.3.2 Generative Methods 10
2.3.3 Normalizing Flow 11
2.3.4 Other Methods 11
2.4 Outlier Exposure 12
2.5 Anomaly Detection in Robotics 12
2.5.1 On low-dimensional data 12
2.5.2 On high-dimensional data 13
2.6 Datasets for Anomaly Detection in Robotics 14
2.7 Conclusions 15

3 Hazard and Robots: A Novel Dataset for Anomaly Detection for Mobile Robotics **17**
3.1 Motivation and Objective 17
3.2 The Dataset 18
3.3 Latest Release (v3.1) 18
3.4 Data Description 19
3.5 Data Acquisition 22
3.6 Additional Releases of The Dataset 23
3.6.1 Release v1 24
3.6.2 Corridors 24
3.6.3 Releases v2 and v3 25
3.7 Anomaly Categorization 25

3.7.1 Background 25
3.7.2 Categorization of Anomalies 25
3.8 Conclusions 26

4 Using Visual Anomaly Detection as a Proxy for Mobile Robot Hazard Detection 27
4.1 Background 27
4.2 Experimental Setup 28
4.2.1 Anomaly Detection on Frames 28
4.2.2 Patch-Level Anomaly Detector 29
4.2.3 Training 30
4.2.4 Metrics 30
4.3 Experimental Results 31
4.3.1 S_8 Model Hyperparameters 31
4.3.2 Patch Aggregation 31
4.3.3 Scales and Anomalies 31
4.3.4 Run-time Evaluation 32
4.4 Demonstrators 33
4.4.1 Wheeled Robot Demonstrator 33
4.4.2 Drone in Factory Demonstrator 34
4.5 Conclusions 36

5 Hazard Detection with Outlier Exposure 37
5.1 Background 37
5.1.1 Models and Approaches 38
5.2 Method 39
5.2.1 Problem Statement and Definitions 39
5.2.2 Real-NVP 40
5.2.3 Real-NVP With Outlier exposure 40
5.3 Experimental Setup 41
5.3.1 Autoencoder: Reconstruction and Dimensionality Reduction 41
5.3.2 Density Estimation (RNVP, RNVP+OE) 41
5.3.3 Binary Classifier Baseline (BCLASS) 42
5.3.4 Wide Residual Networks with Outlier Exposure (WRN+OE) 42
5.3.5 Metrics 43
5.3.6 Computational Costs 43
5.4 Experimental Results 43
5.4.1 Hyperparameter Exploration 43
5.4.2 Comparison with Baselines 43
5.4.3 Effect on the RNVP Target Space 44
5.4.4 Impact of the Number of Exposed Anomaly Frames 44
5.4.5 Impact of the Number of Exposed Anomaly Types 45
5.5 Conclusion 47

6 Active Learning Approaches for Anomaly Detection 49
6.1 Background 49
6.1.1 Active Learning 50
6.1.2 Active Anomaly Detection 50

6.2 Method . . . 50
6.2.1 Anomaly Detection . . . 51
6.2.2 Active Learning . . . 51
6.3 Experimental Setup . . . 52
6.3.1 Implementation . . . 52
6.3.2 Metrics and Tracking . . . 53
6.3.3 Experiment Run . . . 53
6.4 Experiment Results . . . 53
6.4.1 Overall Performance . . . 53
6.4.2 Initial Training Size and Unlabeled Anomaly Presence . . . 53
6.4.3 Anomaly Selection . . . 54
6.4.4 Drone in Train Tunnel Demonstrator . . . 55
6.5 Conclusion . . . 56

7 Applications to 3D Point Clouds **57**
7.1 Background . . . 57
7.2 Related Work . . . 58
7.2.1 Dataset . . . 58
7.2.2 Models and Approaches . . . 58
7.3 Experimental Setup . . . 60
7.4 Experimental Results . . . 62
7.5 Discussion . . . 65

8 Conclusions **67**
8.1 Future work . . . 68

A Appendix **71**

Bibliography **75**

Figures

1.1 Not all hazards can be predicted at design time. Some can be extremely dangerous for the robot or its surroundings 2
1.2 Thesis Overview 4

3.1 Hazard&Robots samples example 22
3.2 Dataset used for image level anomaly detection study 23
3.3 Example of categorization of anomalies 25

4.1 Robot encounter Hazard 28
4.2 Comparing scale approaches 29
4.3 Autoencoder hyper-param experiment results 32
4.4 Autoencoder output comparison at different scales 33
4.5 Example of runtime evaluation 34
4.6 Drone Demo in Factory 35

5.1 Schema explaining the Real-NVP+OE model 39
5.2 Autoencoder Architecture 42
5.3 The impact of parameters λ and γ (x-axes) on the performance of the model measured with the AUC (y-axis) 44
5.4 The effect on loss function and L_2 norm of exposing the model to anomalies during training. 45
5.5 In depth analysis of Outlier Exposure 46
5.6 In-depth analysis of Anomaly type exposure 46

6.1 Progress curves 54
6.2 Progress curves with different anomalies availability 55
6.3 Progress curves with different anomalies variety 55

7.1 Examples of samples from the Mvtec3D dataset 59

Tables

3.1 Dataset split details 20
3.2 Classes of Anomalies in the dataset 21

4.1 Scale model comparison results 30

5.1 Baseline comparison 44

6.1 Active AD initial size analysis 54
6.2 Detailed analysis of average AUC performance 54
6.3 Detailed analysis of average AP performance 54

7.1 Comparing our approaches to the MVTec-3D benchmark 63
7.2 Baselines based Model AUC 63
7.3 XYZ Model AUC 63
7.4 RGB Model AUC 63
7.5 XYZ+RGB Model AUC 64
7.6 Handcrafted based Model AUC 64

Chapter 1

Introduction

In the last decade, the field of robotics has been evolving at an incredible pace. Driven by new manufacturing processes, better electronics, and advancements in core technologies, new mobile robots are becoming available for purposes outside of academic research labs. Wheeled, legged, and track robots are used in various real-world applications, such as site inspection, search and rescue missions, and even entertainment; drones are now a commodity and are used in a wide range of situations, from vacation filming to construction 3D scanning; fixed-wing drones are used to ship medicine and transfusion material across large African countrysides; new small water drones are now used to explore sea and lake beds, extending our reach to new and unexplored environments. With the advent of advanced driver-assistance systems (ADAS) and automated driving systems (ADS), even road vehicles now fall under the robotic hat.

Similarly, the field of artificial intelligence showed an explosion in interest, thanks to the democratization of training methods for deep learning models with commercial GPUs, reaching the current all-time high that we are witnessing today. The diffusion of deep learning models, even without considering LLMs such as ChatGPT, led to huge improvements in performance, efficiency, or efficacy, across many industries and research fields. So many in fact, that it is now difficult to find a research field or an industrial sector that has not seen the use of deep learning approaches.

Thanks to these advancements, the field of intelligent robotics, which sits at the intersection of the two aforementioned ones, has also grown. Now, intelligent robots with varying degrees of autonomy are available for many applications, from pallets and packages automatic unloading out of truck trailers to the destruction of unwanted weeds in agricultural fields. In addition to new applications, robots are being used in some use cases for which they were not designed for; e.g., a legged robot was used to tow a rickshaw carriage [1] or a drone was flown over an erupting volcano [2]. The wider adoption and addition of unique and unexpected use cases are increasing the exposition of intelligent robots to novel environments, scenarios, objectives, and entities that might be unforeseen or different from what can be anticipated. Some of these unexpected events might even lead to problems with the robot's missions, damage to the surroundings, damage to the robots themselves, or even harm to humans. Nonetheless, intelligent robots are expected to perform correctly in all situations and this performance expectations often fall back on the robot's environmental perception system.

[1]https://www.youtube.com/watch?v=zyaocKS3sfg
[2]https://www.youtube.com/watch?v=5cFcOVIYxAQ

It is becoming expected to have robots to work even outside of the design scenarios. Environment perception is more critical than ever, extending beyond applications for autonomous robots such as autonomous road vehicles, robots for agriculture, robotic delivery, and industrial robotic inspection, to applications of semiautonomous ones such as drone inspection and scanning, and even non-robot related ones such as navigation assistance for the visually impaired or mixed and augmented reality.

1.1 Problem Definition

When robots are being developed, a lot of effort is put into preemptively avoiding unwanted scenarios and finding ways to deal with them. Nonetheless, most of the perception systems are limited to a set of specific scenarios and do not generalize out of these. Often, for tasks such as object detection, obstacle detection, and avoidance, a solution is built using supervised machine learning methods; for example, if a new event or problem appears, it is studied, and if deemed relevant, data is collected and labeled and a solution is implemented in the perception system. This approach produces detection models specifically for those known events that can modeled and learned, resulting in well-performing, albeit specific, perception systems.

The key limitation of this approach is that even considering future software updates, the detection capabilities of the environment perception system are limited to what the underlying model is trained on; moreover, the model is limited to the data that can be collected. Imagine a system that is tasked with the detection of obstacles in the path of a robot (see Figure 1.1). These systems are perfectly capable of detecting walls or objects of different dimensions that the robot might face. But, if the robot encounters a pile of small-sized sharp objects such as screws and nails, the obstacle detection system might fail to recognize the hazards resulting in a punctured wheel. Similarly, the navigation system of a drone could wrongly interpret a dense cloud of smoke, resulting in an unknown flight behavior of the drone. To solve these situations, one needs a system that, paired with existing environmental perception modules, adds a way to perceive unexpected hazardous situations; this system would solve the task of hazard detection.

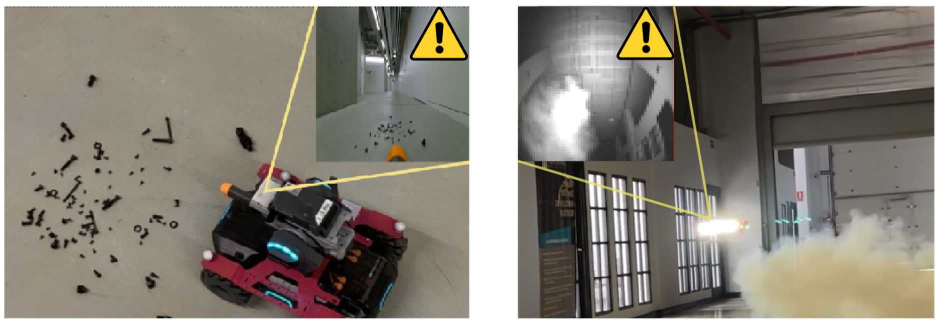


Figure 1.1. Not all hazards can be predicted at design time. Some can be extremely dangerous for the robot or its surroundings

In general, the task of hazard detection for robots consists of finding any situation, object, actors, or events that might be or become hazardous for the robot or its mission. While sharing some aspects with the task of obstacle detection or path planning, hazard detection goes beyond the recognition of objects that might block or reduce the mobility of the robot, and tries to find any type of situation or object that is or might become hazardous. Hazard detection can be differentiated from other perception tasks by considering two key characteristics. The first is that *hazards* need to be detected both when they are exogenous, such as a box or a puddle of water in front of a robot, and when they are endogenous, such as an improperly fitted component that impairs the robot in some ways; for example, a working but badly mounted sensor that points in the wrong direction. The second is that the detection of hazards must happen with or without having prior knowledge of the hazard encountered. Supervised approaches cannot fit these requirements. Supervised solutions in addition to requiring large labeled datasets are strongly limited by the composition of the dataset itself; i.e., by what cases are represented in the data.

To handle difficult situations such as the one now faced by robots, we humans and animals have evolved a behavior counterposed to curiosity known as *neophobia* [85, 66]. Described as “the avoidance of an object or other aspect of the environment solely because it has never been experienced and is dissimilar from what has been experienced in the individual’s past” [87], neophobia allows us and animals alike to avoid perils and risks. In this case, the anomalous situations are identified as those unusual events that differ strongly with respect to past experiences.

One task, in machine learning and deep learning, that fits correctly the *characteristics* described before of hazard detection and is similar to the *neophobia* behavior is anomaly detection, in particular, unsupervised anomaly detection. If we consider all hazards as anomalous patterns in the data collected by the robot from an otherwise nominal environment, we can map hazard detection to unsupervised anomaly detection. Instead of trying to learn all possible classes of anomalies, in the task of anomaly detection, we focus on learning only the concept of normality. This concept is learned through a proxy task, such as input reproduction, and is then used in to determine if a new sample is similar (alas normal) or different (anomalous) from what was seen previously.

1.2 Contributions

This thesis presents our work on the use of unsupervised anomaly detection to solve the task of visual hazard detection for robots. Specifically, we focus on exploring all the aspects that concern the deployment, use, and continual improvement of a hazard detection system used on a robot employed in unstructured environments industrial facilities, underground tunnels, or large campuses’ indoors. Our objective is to study and introduce all the methodologies needed to develop a proper hazard detection system with a feedback loop for iterative detection improvement. Across our work our only source of information is the images (RGB or greyscale) coming from the robot’s front-facing camera; briefly, we also explore 3D point clouds as a possible source of information. In Figure 1.2 we represent an overview of our work as it was just described. One key aspect of this thesis to consider is that, while the neophobia behavior described before implies a reaction to the detection of a problem, in our work we only focus on the detection of hazards and not on the reaction after detection.

In Chapter 2 we dive into the literature of anomaly detection, visual anomaly detection, and their applications in robotics to describe the state of the art before our contributions. We explain how anomaly detection differs from other supervised approaches, we detail the multiple application scenarios and uses of anomaly detection, we explore existing solutions for visual

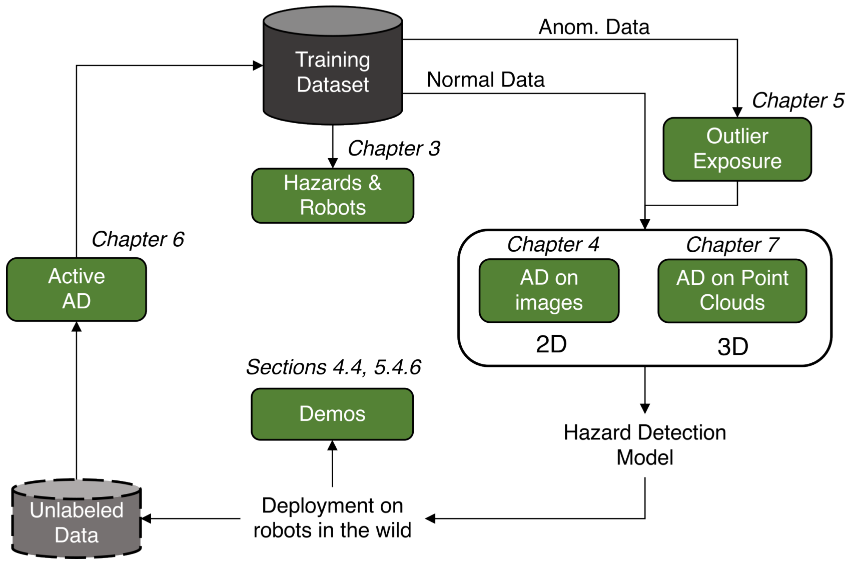


Figure 1.2. Overview of our thesis objective. Contributions are highlighted in green with an indication of relative Chapters. AD stands for anomaly detection.

anomaly detection and finally, we paint the dataset landscape for training deep learning models, highlighting the lack of a proper dataset for the task of hazard detection for robots.

In Chapter 3 we introduce our dataset Hazards&Robots, purposely made to explore the task of hazard detection without assumptions on the robot mission or robot type. To accomplish this, we define three application scenarios using two different robots. The first two scenarios consider a drone and were developed in collaboration with our industrial partner Hovering Solutions Ltd. under the EU Horizon 2020 1-SWARM project. The third scenario focuses on a wheeled ground robot and the data has been collected and labeled by us with the robot patrolling our university corridors. The dataset is composed of high-resolution 512 × 512 pixels images captured using the front-facing cameras of the robots, while they traverse the environments. Both normal samples representing empty, tidy, and in-order environments without obstacles or human presence, and anomalous samples, containing anomalies ranging from dust in the air to glass shards on the floor, have been methodically collected in all three scenarios. This dataset is used in all the following Chapters to train and explore deep learning models for visual anomaly detection (except in Chapter 7).

With our dataset, in Chapter 4 we explore a simple but effective deep learning approach for detecting visual anomalies using undercomplete convolution-based autoencoders. We explore the use of autoencoders as anomaly detectors on image patches at different sizes, analyzing the detection performance for anomalies with different scales. We observe that a low-resolution patch of 64 × 64 pixels representing the whole field of view of the robot (ie. the whole frame) is sufficient to capture most of the anomalies presented. We conclude the Chapter by deploying the model on two robots in two different scenarios, demonstrating that autoencoders can solve the task of hazard detection.

In Chapter 5 we explore Outlier Exposure, a methodology to use anomalous samples during the training of anomaly detection models without changing the core unsupervised principle that characterizes these solutions. While outlier exposure is tested in multiple toy-problems in the original paper, we are the first to test it in an real-world applied scenario; we do so by introducing a new component to our anomaly detector, a Real-NVP model. In the Chapter we explore all the nuances of this approach in a series of experiments, showing that our integration of outlier exposure to the Real-NVP loss indeed works.

With the ability to detect visual anomalies and train models not only with normal data collected in the wild, we now need to explore the best approach to efficiently label the thousands of samples that a deployed robot would collect along its missions. In our vision of a system with a feedback loop for continuous detection improvement, in Chapter 6 we use our dataset as mission proxy, and a variation of our Real-NVP with outlier exposure, to explore 8 different Active Learning queries, 6 from literature and 2 introduced by us, across more than 6500 experiments. The results indicate that the active learning methods to be used, depend on the initial training size and the total amount of samples collected during deployment. With this Chapter, we close the feedback loop and we demonstrate our approach in a drone-based tunnel exploration scenario.

In our last contributions, detailed in Chapter 7, we move from studying the environment using 2D vision to studying it with 3D vision. 3D sensors and scene understanding are gaining popularity and are often found in robots side by side with 2D RGB cameras. In our most recent work, we try to use a foundation model for 3D understanding, namely Point-M2AE, to solve the task of anomaly detection on a recently released 3D point cloud anomaly detection dataset called MVTec-3D. Our results show that while our approach is enough to beat naive and baselines' performance on the dataset benchmark, considerably more work is needed to achieve a generalized performance comparable to what we can obtain using image-based models as bespoke models outperform our solution.

We conclude this thesis with Chapter 8, where we also propose future work we believe is needed to improve the state of the art for the task of hazard detection for robots; we also introduce a note on our future effort on bringing this work to the market.

Chapter 2

Related Work

In this Chapter, we introduce the different topics upon which we base our work. Specifically, we discuss about anomaly detection, with a focus on visual anomaly detection with the objective of building a visual hazard detection system. We continue with a discussion of the applications of anomaly detection in robotics and we finish with a focus on the datasets available for training visual anomaly detection models for robotics.

2.1 Anomaly Detection

Anomaly detection, also known as out-of-distribution detection or novelty detection, is a widely researched topic in machine learning. It focuses on finding patterns or samples in data sources that differ from a known behavior. Consider the task of finding anomalies in food packages in an industrial production line. An anomaly could be a misproduced package or a package with a foreign object like a piece of metal or even something completely different such as a tennis ball. The definitions of what is and what is not an anomaly – also called outlier or novelty depending on the context – are necessarily vague; Chandola states that an anomaly is a "pattern in data that do not conform to expected behavior" [16], while Ruff defines it as "an observation that deviates considerably from some concept of normality" [76]. In our example the context could be a working packaging line, producing correctly built food packages. One thing of notice is that all definitions indicate an incompatibility between finding loosely defined anomalies and typical machine learning tasks such as multi-class classification or regression. In object classification and detection, or semantic and instance segmentation, the subjects of the analysis need to be well-known, well-sampled, and modeled. Instead, in anomaly detection most of the subjects of interest are unknown, as anomalies are rare. In anomaly detection, models have to rely only on known samples that represent normality. This is why sometimes anomaly detection is also referred to as one-class classification, as the only class we can learn from is the normal one. Clearly, anomaly detection fits a different paradigm of data analysis.

Another key aspect of anomaly detection is the data setting. Anomaly detection can be solved in a supervised, semi-supervised, or unsupervised setting; the first two are more difficult to encounter as they require complete or partial labeling of the data, thus a large number of anomalous samples at hand, often difficult to acquire. The supervised setting implies complete labeling of the data and it is less realistic to be applied to a real-world problem; one could say that anomaly detection in a supervised setting is comparable to an open-set classification problem. The semisupervised setting implies that in addition to the unlabelled (assumed normal) data, we have a subset of labeled (normal or anomalous) samples that can be used for training. Instead, the latter setting (unsupervised) is the most diffused one; it assumes that only a known data source of normality is available and it fits almost all kinds of real-world anomaly detection problems. The reason why it is called unsupervised is that the data source used to gather training data is expected to produce only normal samples, except for minor contaminations of undetected anomalies, thus not requiring a labeling process. Going back to the food packages example, if we collect data from a working production line with an already present quality process, we can assume that the data we have collected is representative of anomaly-free packages; that fits the unsupervised setting.

The unsupervised setting fits also our scenario of hazard detection for mobile robots. For example, if we consider a robot that moves along a specific route in a building without encountering any problems or peculiarities, we can assume all the samples collected by the robot as hazard-free, as per the unsupervised setting. Now that we have a general idea of what anomaly detection is and some of its characteristics, we will briefly expand the concept of anomaly.

Categorization of Anomalies

Ruff et al. [76], in their survey, identify common characteristics of anomalies and categorize them as follows: point anomalies, contextual, collective, low-level sensory, and high-level semantic. Point anomalies are the most commonly studied type of anomalies, they represent a single anomalous instance like a photo of a cat in a dataset of dog photos. Contextual anomalies are anomalies that are so only when inserted in a specific context, for example, a screw in a food package line is anomalous but would be ok in a screw manufacturing line. Collective anomalies are sets of anomalies that have some sort of relation with one another but taken individually might look normal. For example, a fraudulent IBAN in a series of money transfers, while composed of characters and numbers that are normal per se, collectively they indicate a malicious act or an error.

The last two categorizations are introduced by Ruff et al. following the diffusion of deep learning models and their wider set of applications. They define low-level sensory anomalies as anomalies related to raw data characteristics; such anomalies would be relative only to the appearance of images, such as brightness, noise, blur, or typos in textual data. Instead, high-level semantic anomalies are those that involve data semantics, such as an open pothole in an environment where potholes are always closed or a different recipient for a recurring money transfer. Detecting the latter is harder, and requires approaches capable of extracting and representing semantic information from data. Note that this categorization regards anomalies and not the data sources dimensionality, low-level anomalies like noise can be found in images (high-dimensionality) and high-level anomalies can be found in low-dimensional data, like detecting a fault in a machine before it happens using low-dimensional sensory data. Shallow machine learning approaches such as SMV, PCA, mixture models, the one proposed by Chakravarty et al. [15] others well reviewed by Chandola et al. in their survey [16] cannot deal with high-level semantic anomalies such as those that a robot might encounter due to their lack of complex information internal representations, failing at the detection task. If we want to be able to find complex, high-level semantic anomalies in the surroundings of a robot, one simple way of achieving this is through vision. For these reasons, we will focus only on deep learning approaches for vision to solve the task of anomaly detection using images coming from the robot's camera in an unsupervised setting.

2.2 Visual Anomaly Detection

We formally define visual anomaly detection as the task whose objective is to learn the detection function $ad : k \rightarrow R$ that assigns a real value (i.e. an anomaly score) to an image k. This often is achieved by learning another proxy function, like input reconstruction, to obtain the anomaly score. Given the unsupervised setting, we state that the function ad is learned over a training set T_{train} composed only of images containing normal situations. The function is tested on a testing set T_{test} that contains images that can be either normal K_n or anomalous K_a. A threshold is needed at inference time to produce a final decision on a sample, based on its anomaly score. The threshold can be chosen using the validation set T_{val} composed of only nominal samples. Later we will allow a relaxation of this definition to the semi-supervised setting by adding to T_{train} of a small set T_{out} containing only known anomalies.

2.3 Models, Methods, and Applications

When operating on images, the task of anomaly detection often consists of finding high-level semantic signals that identify an anomaly, therefore implying some level of semantic understanding of the input. Deep learning approaches, thanks to their ability to extract both low-level and high-level information, have been successfully used for anomaly detection in various fields. As anticipated, often the model responsible for the detection is not trained directly on the task of anomaly detection but on a proxy task, e.g. the reconstruction of a portion of the input. This is a widespread approach to anomaly detection. For example, Haselmann et al. [34] focuses on the surface inspection task; they set up an unsupervised dataset composed of fault-free sample images with central regions of images cut out. To train the anomaly detector they fit an autoencoder-like convolutional model on a proxy reconstruction task. The model fed with images missing the centerpiece, learns to reproduce that part, in case of anomalies it will fill the cutout with a normal texture, thus missing the anomaly and making it segmentable. In industrial manufacturing, anomaly detection is used for a diverse set of scenarios. Scime et al. [82] train a multi-scale CNN (MsCNN) to detect anomalies in power-based additive metal manufacturing processes in a supervised setting.

Anomaly detection applications are not limited to industrial manufacturing, Schleg et al. [80] propose AnoGAN, a generative adversarial network-based model, to find anomalies in medical imaging data. In their work, they assume an unsupervised setting to find and segment anomalies in tomography images of human retinas. Another example of an application of visual anomaly detection is the work of Castellani et al. [13]. In their work, the authors propose to use a siamese autoencoder in combination with a novel clustering method to find anomalies in a digital twin of complex machinery. These application examples show the variety of methods that can be used, in the following sections we provide an in-depth description of some of the principal deep learning methods used for visual anomaly detection.

2.3.1 Reconstruction-based Methods

Most of the recent research [76] on anomaly detection, be it visual or based on signal data, is centered on undercomplete autoencoders with a reconstruction-based proxy task. Undercomplete Autoencoders (AE) [49, 18] are neural reconstruction models that take an image as input and are trained to reproduce it as similar as possible in their output. The typical architecture of an autoencoder consists of two modules, the encoder, and decoder, separated by a set of multilayer perceptron (MLP) called bottleneck. In the autoencoders used for visual anomaly detection, the encoder and decoder modules are convolutional networks that learn to encode the input image into a low-dimensional embedding and decode the semantically high-level informative embedding back into an image. The whole model is trained to minimize losses that measure the reproduction error in the output (e.g. Mean Absolute Error loss) while being constrained in the number of nodes in one of the central hidden layers (i.e. the data bottleneck). This limitation to the amount of information that can flow through the network introduces an error in the autoencoder's output, this prevents the autoencoder from simply copying the input to the output. To minimize the loss on a large dataset of normal (i.e., non-anomalous) samples, the model has to learn to compress the inputs to a low-dimensional representation that captures high-level semantic information. The result is that during training the autoencoder learns to encode high-level semantics of normal samples in the limited latent space to minimize the reproduction error.

At inference time, when tasked to encode and decode a never-seen sample, i.e., a sample from a different distribution than the training set, the autoencoder will commit mistakes in the reconstruction. A normal sample would have some small errors but the reproduction will still be similar to the input. Instead, anomalous samples will be wrongly reproduced with lots of errors; for example, if the input sample is a food package with a visible foreign object like a smartphone, that object could be completely absent from the reproduction as the model has never seen a smartphone and doesn't know how to reproduce it. This means that the reproduction error can be exploited to detect anomalies when an autoencoder is trained only on normal images and potentially, albeit not of interest to us, be used to segment the anomaly. Measuring the reconstruction error for a sample, therefore, yields an indication of the sample's anomaly, normal samples will produce low errors, and anomalous ones high ones.

2.3.2 Generative Methods

A variation to the reconstruction task proposed with undercomplete autoencoders comes from generative models such as Variational Autoencoders [46] and Generative Adversarial Networks (GAN) [32]. These models are also used for anomaly detection tasks. The training process of generative models focuses on mapping an input, sampled from a predefined distribution (i.e., Gaussian or uniform), to the distribution of normal training samples, i.e., generating an image. Variational Autoencoders (VAE) [46] have an architecture based upon undercomplete autoencoders with the addition of variables at the bottleneck level representing the parameters of the chosen probability distribution. For example, mean, standard deviation, and one additional variable acting as random noise input, can be added at the bottleneck level; the last one is needed for the reparameterization trick that allows the backward propagation through the model. During training the encoder learns how to map input samples to the chosen distribution (e.g. a Normal distribution) by changing the parameters, while the decoder receives as input a vector sampled from the chosen distribution. The loss for a VAE is composed of a reconstruction loss identical to those used in undercomplete autoencoders, with the crucial addition of the KL divergence component; this is used as a regularizer during the training of the model. Once the model is trained on normal samples, it is possible to sample the latent space distribution, to compute the likelihood of an input over the latent distribution. Samples that have a low likelihood are identified as anomalies; conversely, a high likelihood indicates normal-looking samples.

GANs are another kind of generative neural network architecture that can be trained in an unsupervised setting and be used to solve anomaly detection. These networks strongly differ from autoencoders and variational autoencoders. Instead of having a single “end-to-end” network, GANs are composed of two separate neural network models (the generator and the discriminator networks) that are pitted against one another. The generator network, fed with normal samples, is trained by minimizing the objective loss, to generate images to mislead the discriminator network, which at the same time aims at maximizing the GANs’ objective function by learning to discern between images from the reference distribution and those outside of it. The result of this zero-sum game is the generator learning to produce images as similar as possible to the reference distribution, and the discriminator learning to find those samples that are different from the original distribution. Differently from VAEs and flow models (which we will explain in the next section), GANs’ latent distributions are not explorable as are learned implicitly by the models; given a trained GAN it is not possible to estimate directly the likelihood of a sample. When applied to anomaly detection tasks, GANs often are modified to better fit the task as an anomaly score cannot be directly computed; for example, the method proposed by Schlegl et al. [81] utilizes a combination of reconstruction and discrimination losses as anomaly score.

2.3.3 Normalizing Flow

Normalizing flow models are often used as, an alternative to VAEs and GANs, for generative purposes [2, 47, 37, 48] but can be also used in the context of anomaly detection [97, 10]. In our work we do not exploit the generative functionality of flow-based models, thus we prefer to describe them as a separate approach to the previously described generative ones. Normalizing flows are used to explicitly learn a probability distribution from the input training set in an unsupervised setting. Through clever use of change-of-variable laws, normalizing-flow models use coupling layers [24] to learn a combination of invertible transformations that map the inputs to a chosen latent distribution, such as a normal distribution. The result is a model that given a sample, maps it to the chosen distribution, or given a point from the distribution, produces a new sample. For the task of anomaly detection, one can use the flow-based model learned mapping to directly estimate the likelihood of a new sample with respect to the probability distribution learned during training. As we will better describe in Chapter 5 and 6, instead of using the model directly on the input image, we first use dimensionality reduction/feature extraction models to lower the data dimensions upon which the flow models are trained. In our research, we use Real-NVP models [24] to learn a mapping from some latent space embeddings of images to a similarly sized multivariate Gaussian distribution.

2.3.4 Other Methods

One-class classifiers, such as Deep SVDD [75] and deep OC-SVM [27], can also be used as anomaly detectors; these methods are used to learn a latent space upon which define a decision boundary around the training instances. Alternatively, in a recent work, Sabokrou et al. [77] propose a new adversarial approach using an Autoencoder as a reconstructor, feeding a standard CNN classifier as a discriminator, trained adversarially. During inference, the reconstructor is expected to enhance the inlier samples while distorting the outliers; the discriminator’s output is used to indicate anomalies. Sarafijanovic introduces [79] an Inception-like Autoencoder for the task of anomaly detection on images. The proposed method uses different convolution layers with different filter sizes all at the same level, mimicking the Inception approach [88]. The model works in two phases; first, it trains the Autoencoder only on normal images, then, instead of the Autoencoder reproduction error, it measures the distance over the pooled bottleneck's output, which keeps the memory and computation needs at a minimum.

2.4 Outlier Exposure

Until now, we assumed that the anomaly detection task we are solving is set in an unsupervised setting, meaning that the training data is only nominal. As explained in the introduction, this assumption can be limiting when dealing with real-world applications of anomaly detection; in any application, after deployment, one can expect to collect some samples of anomalies. Thus, the standard approach to anomaly detection of training a model only on normal samples can be a limitation. In a recent work, Hendrycks et al. [36] introduced Outlier Exposure. This technique increases the model's prediction performance by allowing during training the *exposure* of the model to known samples of outliers. In their work, these known outliers are sampled from large benchmark datasets (e.g., 80 Million Tiny Images [91] or ImageNet-22K [23]) that do not intersect with the training set used for the origin anomaly detection task. The authors utilize these *external* sets, which contain a huge amount of data because they are semantically very different than in-distribution data; in their experiment, the approach helps the model to generalize and detect unseen anomalies. Anomaly detection tasks on real-robot datasets differ from the authors' experiments; much smaller datasets are available, and anomalies can be semantically very similar to normal images. Thus, in Chapter 5 instead of using a large non-relevant dataset, we use a portion of our Hazards&Robots dataset to add outliers during training; these outliers are similar to the normal samples but contain hazards for the robot.

2.5 Anomaly Detection in Robotics

In the field of robotics, anomaly detection has been studied for specific purposes, albeit never for generalized settings such as hazard detection. As for other robotic perception tasks, even anomaly detection can be executed on readings from exteroceptive [97, 19], or proprioceptive [43, 9] sensors. Usually but not exclusively, the first case corresponds to detecting anomalies on high-dimensional sensing data, as we require a lot of information to understand the surroundings, while the second often refers to low-dimensional data, e.g., we are checking the robot *heart beat* and thus we require fewer dimensions. In the next two subsections, we explore the use of anomaly detection, with both shallow and deep approaches, in robotics, on different data dimensionalities.

2.5.1 On low-dimensional data

Historically, anomaly detection in robotics has focused on using low-dimensional data streams from sensors. A large amount of literature deals with detecting anomalies using low-dimensional (but potentially high-frequency) data streams. On this type of data, a combination of handcrafted feature selection, Machine Learning, and, recently, deep learning models have been used to find anomalies in and around robots.

Khalastchi et al. in two works [44, 43], focus on building an anomaly detection system, for finding problems such as physical faults or software bugs that might compromise the autonomy of unmanned vehicles. They propose ODDAD an online data-driven anomaly detector that takes as input a series of low-dimensional sensory information, such as odometry, speed, heading, or actuators' internal sensory data. The data is then compared with a Mahalanobis distance and a custom correlation detector to find anomalies. They use commercial Unmanned Aerial Vehicles(UAV), a vacuum robot, and a flight simulator to test their method obtaining good results in multiple scenarios.

Sakurada et al. [78] task themselves to build an anomaly detector for internal failures of a spacecraft system. The authors benchmark different approaches on both real and artificial telemetry data. They compare the performance of an autoencoder, a denoising autoencoder, a linear PCA, and a kernel PCA. They find that autoencoders and denoising autoencoders are best at detecting, even subtle, anomalies; they continue by stating the advantages of autoencoders of learning non-linear correlations in the data without the need of manually defining a non-linear kernel, as in the k-PCA. Finally, they explore the autoencoder's latent space confirming that the autoencoders have properly learned a concept of normality.

Birnbaum et al. [9] propose an approach for detecting cyber-attacks, sensor faults, or structural failures on Unmanned Aerial Vehicle (UAV) flight data. Their approach is based on comparing hand-crafted descriptors of UAV behavior (including state, flight plan, and sensor readings) to predefined behavioral profiles. To do so they use heuristic-based event matrices that are then used as references for a bespoke detection algorithm. To test their approach they developed a simulation UAV behavior testing platform consisting of a flight sim, an autopilot, and a set of software needed for flight and mission planning. With this system, they successfully demonstrate that they can internally detect sensor failures and general behavioral anomalies indicating an external attack.

Park et al. tackle anomaly detection in robot-assisted feeding, using Hidden Markov Models with hand-crafted features [70], or a combination of Variational Autoencoders and LSTM networks [71]. The latter approach encodes a sequence of multi-modal sensory signals into a latent space and then estimates the expected distribution of the received inputs: an anomaly is detected when the negative log-likelihood of the current input, given an expected distribution, exceeds a certain threshold. In their test, they use a real robot arm and in addition, check if their detectors can find anomalies in real-time. Even in this research, the data comes from low-dimensional sources, namely 17 different internal sensors and 4 hand-crafted features.

2.5.2 On high-dimensional data

Anomaly detection on high-dimensional sensing data, like images, videos, or point clouds, is used in many different robotics application domains. Due to the complexity of the data, the anomalies detected by the following approaches often match the high-level semantic anomalies described in a section before. An early approach [15] to anomaly detection on high-dimensional data concerns the task of autonomous robot patrolling used to identify unexpected situations. The authors apply image-matching algorithms to panoramic images of large databases of normal ones to detect anomalies. The detection algorithms use stereo vision techniques to find anomalies and a particle filter to track them. Unfortunately, this approach is also sensible to minor changes in the environment, such as a door open that was closed, and thus is suggested only in indoor environments with few or no changes.

In a more recent approach, Christiansen et al. [19] propose a method to detect obstacles and anomalies in the surroundings of autonomous agricultural robots. Their solution, called DeepAnomaly, is based on a custom Convolutional Neural Network(CNN) architecture derived from AlexNet [51]. To find anomalies, the authors propose to use the ability of a CNN to extract features of higher levels in the network's deeper layers. They benchmark different approaches to compare the features, at a specific network layer, of the input with the features collected during training from the same layer. They compare more than 400 configurations to determine the best combination of feature level and detector, proposing different solutions depending on some high-level assumptions of the task, for example, the need to find bounding boxes for anomalies or the need to run the model in real time. While this scenario is similar to ours, they focus on finding only specific anomaly cases that are relevant to agricultural robots.

The most similar work to ours is from Wellhausen et al. [97]. Their work is a cornerstone for our early research as they extensively explore, across three approaches, solutions for building a system for safe navigation for a legged ANYmal [40] robot. Similar to us, they assume that the robot has to travel in unknown environments, and the system they design is precisely tasked with the objective of avoiding foothold locations on terrain whose appearance is anomalous. As most of the research is in anomaly detection, even in this paper, the setting is unsupervised. The models proposed are trained to detect anomalies of future footholds. They do so by using foothold projections over camera images. In this work, the authors compare an undercomplete autoencoder, a Deep SVDD model [75] and Real-NVP [24] one; the last two share with the autoencoder's encoder to extract features from the input data. The sensory data used by the models is multi-modal; using multiple sorties with no problems visible, the authors collect information not only from the front-facing RGB cameras but also from the onboard depth sensors thus collecting RGB-D images. Then the information about the footholds is retroactively projected onto the RGB-D images producing a set of image patches where the robot legs walked successfully. In the final comparison, the Real-NVP model performed best, closely followed by the autoencoder. Finally, they demonstrate that the model can run in real-time on a Jetson board. Compared to our work, Wellhausen et al. focus on finding anomalies relevant only to robot leg placement.

2.6 Datasets for Anomaly Detection in Robotics

A large portion of the anomaly detection literature [36, 75, 94] relies for the experiments, on datasets built for image classification, such as MNIST [54], ImageNet [23, 53], CIFAR [50], and SVHN [67]. What is often done, is to define an anomaly as an instance sampled from a different dataset than the one used for training, or an instance of a given class that is not seen in the training set. This is not a good model for realistic anomalies in robotics applications like the scenario we focus on. Anomalous frames might differ from normal ones in subtle ways, like a crack in the camera lens. To properly develop and evaluate anomaly detection methods for robotic applications, researchers need datasets that represent *realistic* anomalies, as well as large amounts of normal images with their expected variability. Unfortunately, most of the recently-released datasets, while tackling the data collection issue, focus explicitly on specific sub-tasks of anomaly detection [10, 97, 8].

MVTec [8] is specific to defects of industrial goods; it is composed of 5354 RGB images of 15 different objects captured in controlled scenarios in an industrial production environment. Fishyscapes [10], built upon Cityscapes [22], focuses on segmenting anomalous objects in images acquired by self-driving vehicles. It includes both anomalous images acquired in real-world settings as well as synthetically generated data; dense segmentation masks of anomalous objects are provided. Wellhausen et al. [97] propose ANNA, a dataset for detecting visual anomalies in ground patches, to predict terrain traversability for legged robots; the dataset is built from observations acquired by robots traveling on different terrains. None of these provide a general dataset that can be used for multiple robotic applications and is thus compatible with our hazard detection task. Such a dataset would allow research towards general approaches instead of ad-hoc ones, that is why in the next Chapter we introduce our dataset for the task of hazard detection.

2.7 Conclusions

In this Chapter, we saw that anomaly detection concerns finding anomalous patterns in data given a model of normality [16, 76]. Its applications cover diverse fields: surveillance [15], intrusion detection [9], medical imaging [80], fault detection [43], agriculture [19] and autonomous robot [97]. In literature, these are achieved both using low-dimensional data [44, 43] or high-dimensional one [15, 19]. In robotics, anomaly detection is used to detect anomalies in the environment [97, 19] or in the robot itself [43, 9] with diverse derived objectives such as patrolling or safe robot navigation. Even if the task of visual anomaly detection in robotics is gaining track, there is still a lack of a general-purpose dataset to study new approaches. In the following Chapters, we will introduce our dataset for hazard detection and then focus our analysis on finding anomalies that might become hazards for a robot using deep learning methods based on information from exteroceptive visual sensors such as cameras.

Chapter 3

Hazard and Robots: A Novel Dataset for Anomaly Detection for Mobile Robotics

In the previous Chapters we introduce the key concepts and the state-of-the-art methods regarding anomaly detection and visual anomaly detection, with a focus on robotic hazard detection; in Section 2.6 we specifically discussed the limitations of current datasets available for the task of visual anomaly detection for mobile robots. While some exist, none are compatible with our main task of hazard detection. Motivated by the lack of datasets to train a deep learning model for visual anomaly detection, in this Chapter, we introduce our dataset, purposely made to explore both visual anomaly detection for robots and the task of hazard detection without assumptions on the robot mission or robot type. The dataset we propose has been released multiple times, for the main part of this Chapter we will focus on the last one (release v3.1), with more information on the other releases in Section 3.6.

3.1 Motivation and Objective

In Chapter 2 we discussed the limitations of existing datasets regarding anomaly detection in robotics. Even outside of robotic anomaly detection datasets, state-of-the-art models and methods [49, 46, 76, 10] for anomaly detection in images are often trained and tested on datasets that strongly differ from what is encountered by autonomous robots [8, 80, 76] in a real scenario. These are often limited to the paramount task the researchers wanted to solve. We propose to do the inverse, by defining a dataset that fits no particular robotic task and instead can be adapted to different ones. We decide to start by focusing on indoor scenarios. Indoor scenarios can be relevant for a diverse selection of robotic tasks, and anomaly (i.e. hazard) detection can be adapted to different cases. For example, in robotic safe navigation anomalies might represent potential hazards to avoid, or for robot patrolling, anomalies are interesting events to report, or even in robotic industrial inspection where anomalies might become costly problems such as a pipeline leaking a liquid. As we want to study anomaly detection independently of the specific task that the robot may be doing, we set ourselves to collect a dataset to act as a proxy of realistic robotic settings. The dataset we collected makes it possible to develop and benchmark anomaly detection systems for robots that use cameras indoors.

3.2 The Dataset

The dataset we propose is called Hazard&Robots, and across different releases, it comprises two robots, a drone and a wheeled robot, and three scenarios. The dataset differs from well-known anomaly detection datasets [8, 80] as those are usually collected in fixed, controlled settings often regarding industrial processes and thus not relevant for robotics. Ours is collected by mobile robots with environmental variables such as reflections, changes in illumination, or variations in the surroundings. The first two scenarios have been developed in collaboration with our industrial partner Hovering Solutions, under the EU Horizon 2020 1-SWARM project. Hovering Solutions is a company specializing in visual inspection and 3D mapping of inaccessible tunnels and pipes. In these scenarios, we utilize a drone to fly inside underground tunnels (in a simulation) and in a factory space (with a real drone). With these, we want to replicate the inspection process of a custom-made industrial drone that, due to the environmental conditions has no connection to the outside. The drone, while accomplishing its missions with complete autonomy, might encounter hazards such as raindrops due to condensation on the tunnel roof, small roots coming from the ceiling, or dust that might reduce the navigation system functionalities. The drone has an obstacle and navigation system composed of a planar lidar and a camera, but due to the unpredictability of underground tunnels or the presence of foreign objects in the factory space, this system cannot be solely relied upon thus a hazard detection system has to be run onboard. These scenarios are representative of the real use cases of our partner. The third, and main, scenario is composed of video frames collected by a DJI Robomaster S1 that traverses the USI-SUPSI Est campus' corridors. In this scenario, we imply that the Robomaster S1, a ground-wheeled robot, is tasked to patrol our university's corridors. Note that this assumption was made only to simplify the collection process but does not influence the dataset content per se. The robot used is fitted with a monocular RGB camera on top of a controllable gimbal. The robot might face multiple situations that either can cause damage to the robot or might be of interest for the patrolling task, thus a hazard detection system is needed. Our dataset is available on Zenodo [62]; at the time of the writing, has been downloaded more than 470 times.

The dataset is composed of 324,408 RGB labeled frames, and corresponding 512-sized feature vectors; it contains 145,470 normal frames and 178,938 anomalous ones categorized in 20 different anomaly classes. Thanks to its large number of frames, the dataset can be used to train and test current and novel visual anomaly detection methods such as those based on deep learning vision models, it can be used by applied researchers to replicate real robot issues and test their solutions or by theoretical researchers to benchmark their models on a realistic anomaly detection task.

3.3 Latest Release (v3.1)

As anticipated, the dataset has different releases. In the latest one, the data is extracted from videos recorded with a DJI Robomaster S1 front-facing camera. The ground robot, controlled by a human operator, traverses university corridors. With the robot we explore 6 different university corridors; for these environments, a normal situation corresponds to a tidy corridor with no standing humans and all windows and doors closed. To vary the data recorded and avoid biases, the corridors are passed multiple times, at different times of day and night, some with varying furniture distribution. Regarding anomalies, we considered anomalies including the presence of humans (walking or standing), unexpected objects on the floor, defects to the robot, and simulated accidents (e.g. spilled water). The dataset is split into training (only normal), training (mixed), validation, and test sets. The dataset provides both images and pre-extracted feature embeddings to allow a larger variety of approaches to the task of anomaly detection.

3.4 Data Description

The dataset in the final version (V3.1) is composed of 8 files:

- *README.md* the readme files containing some basic information about the dataset,
- *embeddings.zip* a zip file containing 324,408 512-sized feature vectors extracted using a CLIP ViT-B/32 from the images. The vectors are divided into 4 PyTorch `.pt` files for fast loading,
- *metadata.zip* a zip file containing the annotation files for the dataset,
- *training_set.zip* a compressed folder containing $47,157$ video frames,
- *training_mixed_set.zip* a compressed folder containing $144,603$ video frames,
- *validation_set.zip* a compressed folder containing $2,801$ video frames,
- *test_set.zip* a compressed folder containing $129,846$ video frames,
- *code.zip* a compressed folder containing the code used in the creation of the frames, for reading the dataset and additional custom PyTorch dataset Classes for both the embeddings and images.

Regardless of the representation in frames or embeddings, the data is split into four disjoint sets:

- *Training*, containing only normal samples,
- *Training_mixed*, containing both normal and anomalous samples,
- *Validation*, containing only normal samples,
- *Testing*, containing normal and anomalous samples.

The training sets are used for training anomaly detection models and the validation set for performing model selection, for example determining when to stop training a model. The testing set can be used to compute quantitative performance metrics for the anomaly detection problem. In version 1 of the dataset, we include a qualitative testing set that can be used to analyze how the model outputs react to a video stream as the robot traverses normal and anomalous environments, more info is in Section 3.6.1.

A summary of the four sets is in Table 3.1. Note that each split has a set of samples from each corridor and the samples are not repeated across training, validation, and testing sets.

Set	Total samples	Anomalous samples
Training (normal)	47,157	0
Training (mixed)	144,603	94,362
Validation	2,802	0
Test	129,846	84,576

Table 3.1. Dataset split details

The compressed folders (`<set_name>_set.zip`) contain JPEG images with a separate sequential numeration for each folder. The file naming follows this style: `000000_512_512.jpg` is the 1st sample of the set (the file numeration starts at 0) and `001234_512_512.jpg` is the 1235th sample in the set.

In `embeddings.zip`, we provide four PyTorch files and four Numpy files; each file represents one of the sets. The naming convention is `<set_name>_embs.pt` for PyTorch files and `<set_name>_embs.npy` for Numpy ones. When loaded with the dataset classes provided in the code.zip, the result is a PyTorch tensor of shape `(set_size, 512)`; for example, for the validation set the shape of the tensor in `validation_embs.pt` is `(2802, 512)`. Be sure to use the correct code provided for Numpy or PyTorch embeddings. These vectors correspond to embeddings of the image encoder component of the CLIP model ViT-B/32 publicly provided by Open AI [73].

In the `metadata.zip` compressed folder we provide 6 CSV files and 2 TXT files. Two CSV files contain the frame labeling; each CSV file provides two columns; `frame_id` with the sample ids and label with the label id of the sample.

- `test_frames_labels.csv`,
- `training_mixed_frames_labels.csv`,

The other four CSV files contain the corridor labeling for each sample of a set; each CSV provides a `frame_id` column and an env column; the latter indicates the corridor ID of a sample.

- `training_set_frames_envs.csv`,
- `training_mixed_set_frames_envs.csv`,
- `validation_set_frames_envs.csv`,
- `test_set_frames_envs.csv`,

The two TXT files contain the mapping between IDs and names of labels and environments:

- `labels_mapping.txt`,
- `envs_mapping.txt`.

The six corridors used for the data recording can be grouped into three classes called *environments*; the environments are: long, short, and underground. Long corridors have a wooden wall, short corridors are larger and have chairs and tables, and underground corridors are concrete corridors with maintenance tubing and cabling on the ceiling. In the dataset we provide 21 different classes, in Table 3.2 we list all labels, their IDs, and the number of samples across all splits. In Figure 3.1 we show an example for each class of anomaly and some examples of normal data in the different corridors.

Table 3.2. Classes of Anomalies in the dataset

Id	**Name**	**Samples in Training**	**Samples in Testing**
0	Normal	47,157 (training) +50,241 (training mixed) +2,802 (validation) =100,200	45,270
1	Box	5,868	4,976
2	Cable	5,460	5,201
3	Cones	5,621	4,619
4	Debris	4,474	4,588
5	Defects	13,295	12,056
6	Door	707	561
7	Floor	4,751	4,305
8	Human	14,318	13,358
9	Misc	597	236
10	Tape	6,014	5,604
11	Trolley	4,811	4,236
12	Clutter	659	538
13	Foam	174	51
14	Sawdust	253	136
15	Shard	39	109
16	Cellophane	372	395
17	Screws	169	157
18	Water	211	93
19	Obj. on Robot	12,551	11,716
20	Obj. on Robot 2	14,018	11,641

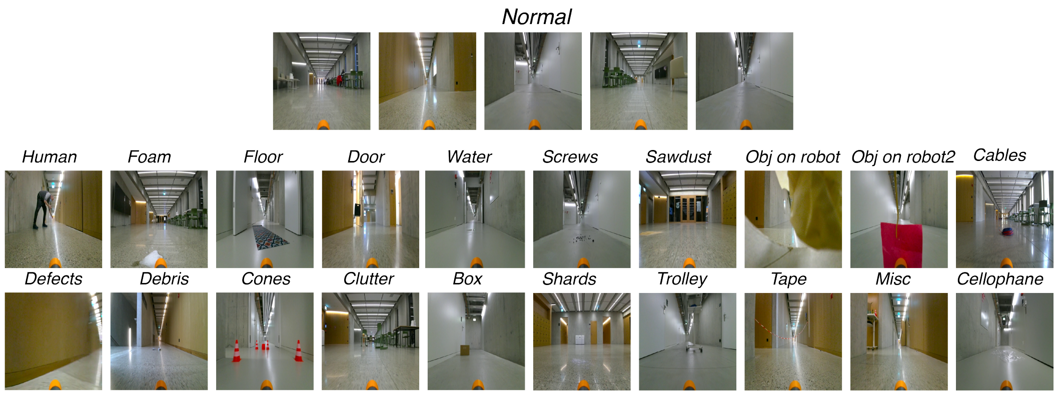


Figure 3.1. Samples of all anomalies and 5 samples of normal frames from the Hazards&Robots dataset. (Figure in color)

3.5 Data Acquisition

We record the feed of a DJI Robomaster front camera on a microSD or via the Robomaster app. The robot is teleoperated in more than 550 sorties, in 6 different corridors located on two different floors of a university campus. In all recordings, the robot moves at the center, along the length of the corridor.

The DJI Robomaster S1 provides a 120 deg field of view camera with a maximum resolution of 2560 × 1440. The camera, capable of recording full-HD videos at 30fps, is mounted on a gimbal. In addition, the robot is fitted with omnidirectional wheels, and combined with the gimbal, it allows for a decoupling of the direction of movement from the camera's pointing direction. This was exploited, for example, to simulate problems with the robot; frames acquired in this condition are labeled as defects.

The data collection was structured in multiple phases, all with the same setup, to capture the environments in different conditions. The resulting recordings vary in duration, location, presence of anomalies, illumination, and corridor traversing direction. In normal videos, no anomalies are present, while in anomalous ones we manually place anomalies making sure that anomalies are always visible and identifiable. In longer sorties the robot fully traverses random sections of the corridor, instead, in short ones, the anomalies are placed at 2m from a start position. Then, the robot moves towards the anomaly while keeping it in frame, stopping at 5cm and moving back to the starting position. After a set of sorties is recorded, the recordings are divided into frames; these are then resized and from the resized frames the feature vectors are extracted.

Note that the very concept of an anomaly in robotic perception is highly subjective and application-dependent [9, 19, 97]. Whether a given situation should be considered an anomaly depends on the features of the robot and its task; for example, consider a robot patrolling corridors with floors normally clear of objects; the presence of screws and bolts littering the ground could be hazardous for a robot with inflated tires that could get punctured, but completely irrelevant for a drone or legged robot. On an orthogonal dimension, some applications might be interested in determining anomalies regardless of whether they pose a hazard to the robot: in a scenario in which a robot is patrolling normally empty tunnels, finding anything different in the environment could be a sign of an intrusion and should be detected. The appearance

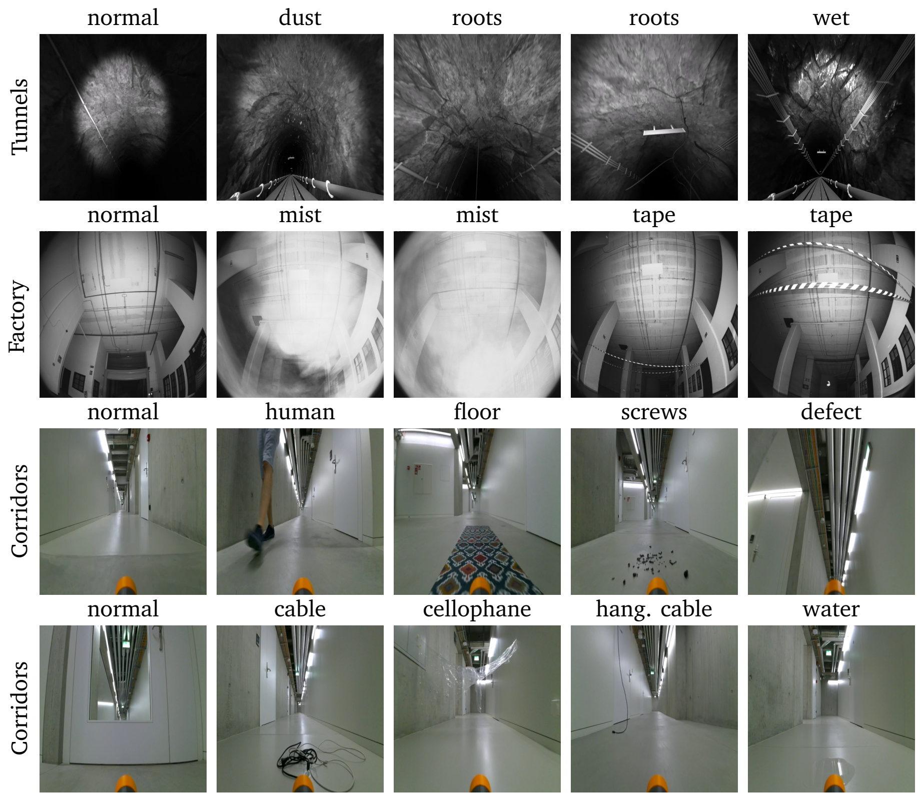


Figure 3.2. The testing datasets are composed of normal images and images of different anomaly classes.

of anomalies in forward-looking camera streams is also dependent on the distance from the robot; wires or other thin objects that might pose a danger to a robot could be simply invisible if they are not very close to the camera. Our labeled sets are manually curated, we defined the recording process to be as objective as possible but still, on certain occasions, and in the previous releases of the dataset, we used our best judgment to determine whether to consider a frame anomalous or not. For example, frames with anomalies that are not clearly visible in the 512 × 512 full-resolution images are excluded from the testing set.

3.6 Additional Releases of The Dataset

On the Zenodo dataset page it is possible to select previous releases of the dataset; release version 1, for example, contains data coming from a drone and a simulated drone. From release version 2 onwards we focused only on collecting data, methodically, using only the DJI Robomaster S1. All the versions of the dataset are complementary.

3.6.1 Release v1

The first release includes a large number of grayscale or RGB frames with a 512 × 512 px resolution. For each dataset, we define four subsets: a training set, a validation set, composed of only normal frames a labeled testing set, and an unlabeled qualitative testing set, consisting of one or more continuous data sequences acquired at approximately 30 Hz, depicting the traversal of environments with alternating normal and anomalous situations. Only in the first version of the dataset did we also split it into three scenarios: Tunnels, Factory, and Corridor. The latter is the only one present in the other versions.

Tunnels The scenario, provided by Hovering Solutions Ltd, is composed of grayscale frames from simulated drone flights along procedurally generated underground tunnels presenting features typically found in aqueduct systems, namely: random dimensions; random curvature radius; different structures on the floor; tubing, wiring, and other facilities attached to the tunnel walls at random positions; uneven textured walls; various ceiling-mounted features at regular intervals (lighting fixtures, signage). The drone flies approximately along the centerline of the tunnel and illuminates the tunnel walls with a spotlight approximately coaxial with the camera. Both the camera and the spotlight are slightly tilted upwards.

This scenario is composed of 143070 frames: 72854 in the training set; 8934 in the validation set; 57081 in the quantitative labeled testing set (40% anomalous); 4201 in the qualitative testing sequences. Three anomalies are represented: *dust*, *wet* ceilings, and thin plant *roots* hanging from the ceilings (see Figure 3.2). These all correspond to hazards for quadrotors flying real-world missions in aqueduct systems: excessive amounts of dust raised by rotor downwash hinder visual state estimation; wet ceilings, caused by condensation on cold walls in humid environments, indicate the risk of drops of water falling on the robot; thin hanging roots, which find their way through small cracks in the ceiling, directly cause crashes.

Factory This scenario contains grayscale frames recorded by a real drone, with a similar setup to the one simulated in the Tunnels scenario, flown in a testing facility (a factory environment) at Hovering Solutions Ltd. During acquisition, the environment is almost exclusively lit by the onboard spotlight.

This scenario is composed of 12040 frames: 4816 in the training set; 670 in the validation set; 6001 in the quantitative testing set (53% anomalous); 553 in the qualitative testing sequences. Two anomalies are represented: *mist* in the environment, generated with a fog machine; and signaling *tape* stretched between two opposing walls (Figure 3.2). These anomalies represent large-scale and small-scale anomalies, respectively.

3.6.2 Corridors

This scenario contains RGB frames recorded by the same DJI robot of the final version of the dataset. The corridors have a mostly uniform, partially reflective floor with few features; various side openings of different sizes (doors, lifts, other connecting corridors); variable features on the ceiling, including service ducts, wiring, and various configurations of lighting. The robot is remotely teleoperated during data collection, traveling approximately along the center of the corridor.

This scenario is composed of 52607 frames: 25844 in the training set; 2040 in the validation set; 17971 in the testing set (45% anomalous); 6752 in qualitative testing sequences.

8 anomalies are represented, ranging from subtle characteristics of the environment affecting a minimal part of the input to large-scale changes in the whole image acquired by the robot: *water* puddles, *cables* on the floor; *hanging cables* from the ceiling; different mats on the *floor*, *human* presence, *screws* and bolts on the ground; camera *defects* (extreme tilting, dirty lens) and *cellophane* foil stretched between the walls.

3.6.3 Releases v2 and v3

The releases version 2 and 3 of the dataset are composed of frames acquired by the Robomaster S1 along the university corridor; these versions can be seen as a subset of the final version

3.7 Anomaly Categorization

In our work on the dataset, we also expand on the anomaly categorization that is proposed in the literature. Using our Hazards&Robots dataset v1, we explore new ways of classifying an anomaly based on its characteristics.

3.7.1 Background

In Chapter 2 we already explored an anomaly classification proposed by Ruff [76].

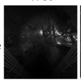
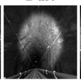
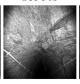
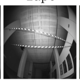
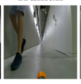
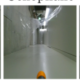
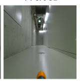
Scenario	**Tunnel** (AUC=0.86)			**Factory** (AUC=0.93)		**Corridor** (AUC=0.72)							
Anomaly	Wet	Dust	Roots	Mist	Tape	Human	Floor	Screws	Defect	Cable	Cellophane	Hang.Cable	Water
Example													
Level	Low	Low	Low	Low	High	High	Low	Low	Both	High	Low	High	High
Hazard	Yes	Yes	Yes	No	Yes	Both	No	Both	Yes	Both	Both	No	Both
Geometric	No	No	Yes	No	Yes	Yes	No	No	Both	Both	No	Both	No
Frames	17058	1578	4079	2812	365	1196	667	559	1727	1820	1658	284	469

Figure 3.3. Summary of the anomalies represented in the dataset, with an example of each. The top row reports the score of our model for each scenario. Below: the categorization of the dataset's anomalies along three of the four proposed axes.

3.7.2 Categorization of Anomalies

We propose to categorize visual anomalies encountered by mobile robots during their operation along four independent axes. The first, following Ruff et al. [76], differentiates low-level sensory and high-level semantic anomalies. Low-level anomalies are described by features close to the image space, such as image brightness, smoothness, noise, and texture; one example of such anomaly occurs when the robot suddenly finds itself in the dark or when it is blinded by direct light. High-level anomalies refer to the semantic contents of the image: examples include the observation of a pressure gauge reporting a different value than usual, or of a puddle of liquid on the ground.

The second axis represents whether an anomaly is a hazard to the robot. This is specific to the robot's characteristics (an oil puddle on the floor might represent a hazard for a ground robot but not for a drone) and its task (the puddle is not dangerous unless it lies on the robot's path).

The third axis differentiates anomalies that are relevant to the robot mission and anomalies that are not. For example, a patrolling robot might want to detect and report the fact that a door that is usually closed is observed to be open, whereas a delivery robot should not be affected by this observation unless it impacts its path planning.

The fourth axis discriminates visual anomalies that are geometric in nature from those that are not. Geometric anomalies have a well-defined 3D shape in the robot environment (e.g. a never-before-seen object in a normally-free corridor) or consist of changes in the position or shape of a part of the environment (a wall that collapsed). Non-geometric anomalies are anomalies that, while perceivable using an RGB camera, would be undetectable with an ideal depth sensor: for example, a puddle on the ground; plaster rubble scattered across a building floor; dust, fog, or smoke; a wet ceiling in a tunnel.In Figure 3.3 we show examples of all the represented anomalies in version 1 of the dataset. We also classify anomaly types along the axes introduced before. The third axis is not represented in the Figure since it is mission-dependent and our dataset is gathered with no specific task in mind.

3.8 Conclusions

In this Chapter, we described our novel open-source dataset, Hazards&Robots. The dataset on its latest release is composed of more than 320000 labeled RGB frames with a resolution of 512 × 512 pixels and their relative 512 dimensional feature embedding. The latest release focuses on data collected by a wheeled robot that explores the indoor corridors of a university building. Other releases also feature a drone real industrial facility and a simulation of a drone that traverses underground tunnels. We collected and publicly released the dataset as, to the best of our knowledge, this is the first one made for the task of hazard detection in the context of robotics. Finally, we expanded the anomaly categorization first explained in Chapter 2, by introducing robotic relevant categories; we think that these might be helpful to better analyze the hazard detection requirements of a robot application.

In the following Chapters, we will use this dataset in its different releases to train different deep-learning models for achieving our objective of building a hazard detection system.

Chapter 4

Using Visual Anomaly Detection as a Proxy for Mobile Robot Hazard Detection

In the previous Chapter, we introduced our dataset Hazards&Robots; we described the motivation and acquisition of a hazard detection visual dataset purposely made for studying the task for robots. In this Chapter, we propose our solution for employing visual anomaly detection techniques to detect hazards. We will use our dataset for training and testing our models. Various anomalies are considered, spanning from camera malfunctions to environmental hazards: some affect the acquired image globally; others only impact a small portion of it. We benchmark an anomaly detection method based on autoencoders operating at different scales on the input frames. Finally, we deploy our best model on two robots in two separate demonstrators; with the demonstrators, we show that our solution is reliable and can be used in real-time hazard detection.

4.1 Background

Many emerging applications involve a robot operating autonomously in an unknown environment; the environment may include hazards, i.e., locations that might disrupt the robot's operation, possibly causing it to crash, get stuck, and more generally fail its mission. Robots are usually capable of perceiving hazards that are expected during system development and therefore can be explicitly accounted for when designing the perception subsystem. For example, ground robots can typically perceive and avoid obstacles or uneven ground.

In this Chapter, we study how to provide robots with a different capability: detecting *unexpected* hazards, potentially very rare, that were not explicitly considered during system design. Because we don't have any model of how these hazards appear, we consider anything that is novel or unusual as a potential hazard to be avoided.

Animals and humans exhibit this exact behavior [85], known as *neophobia* [66]: "the avoidance of an object or other aspect of the environment solely because it has never been experienced and is dissimilar from what has been experienced in the individual's past" [87]. We argue that autonomous robots could benefit from implementing neophobia, in particular whenever

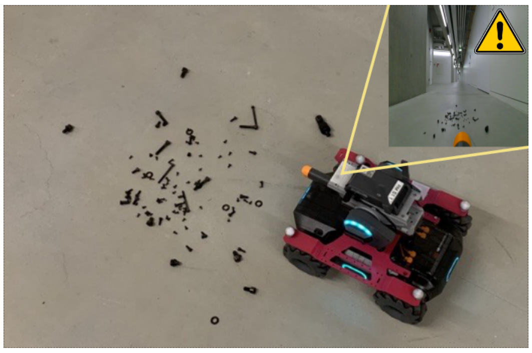


Figure 4.1. A Robomaster detects an anomaly in the camera frame: cautiousness is required.

the potential failure bears a much higher cost than the avoidance behavior. Thus, for example, for a ground robot, it makes sense to avoid unusual-looking ground [97] when a slightly longer path on familiar ground is available; or a planetary rover might immediately stop a planned trajectory if something looks odd, waiting for further instructions from the ground control.

Our experiments are motivated by a similar real-world use case in which a quadrotor equipped with sophisticated sensing and control traverses underground tunnels for inspection of aqueduct systems. During the flights, that might span several kilometers, the robot is fully autonomous since it has no connectivity to the operators; they wait for the robot to either reach the predetermined exit point or — in case the robot decides to abort the mission — backtrack to the entry. In this context, a crash bears the cost of the lost hardware and human effort, but most importantly the lost information concerning the hazard that determined the failure, remains unknown. It then makes sense to react to unexpected sensing data by aborting the mission early and returning to the entry point;[1] operators can then analyze the reported anomaly: in case it is not a genuine hazard, the system can be instructed to ignore it in the current and future missions and restart the exploration.

4.2 Experimental Setup

In this Chapter, we use the release v1 of our dataset described in Section 3.6.1 to create disjoint sets for training validation and testing.

4.2.1 Anomaly Detection on Frames

We follow our definition of visual anomaly detection task from Section 2.2 and train an anomaly detector to map a frame (512×512) to an anomaly score. The frame-level anomaly detector relies on a patch-level anomaly detector (see Figure 4.2), which instead operates on low-resolution inputs (64×64), which is a typical input size for anomaly detection methods operating on images [97, 42].

First, the frame is downsampled (using local averaging) by a factor $s \in \{1, 2, 4, 8\}$; we will refer to the respective models as S_1, S_2, S_4 and S_8. The resulting downsampled image, with

[1]Similarly, retention of information following encounters with novel predators is one of the recognized evolutionary advantages of neophobic animals [65].

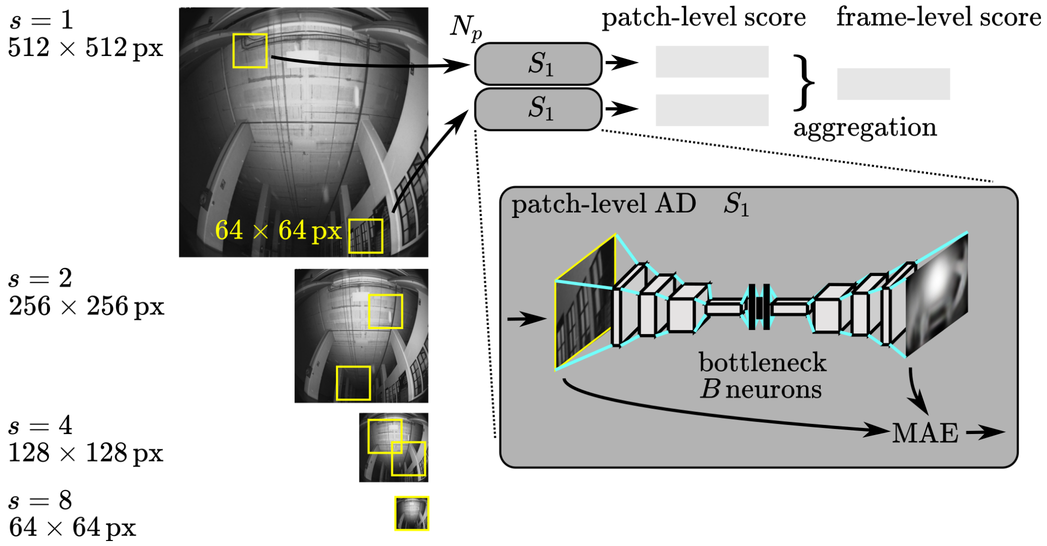


Figure 4.2. Anomaly detection model: using an autoencoder to compute the patch-level anomaly scores, which are aggregated in a frame-level score.

resolution $512/s \times 512/s$, is standardized to zero mean and unit variance, independently for each channel; we then extract N_p 64×64 patches, at random coordinates, such that they are fully contained in the downsampled image. The patch-level anomaly detector is applied to each patch, producing N_p anomaly scores; these are aggregated together (e.g., computing their average) to yield the frame-level anomaly score.

Note that in the case of S_8, $N_p \equiv 1$ since a unique patch can be defined on a 64×64 downsampled image. This corresponds to the special case in which the whole frame (after downsampling) is directly used as input to the patch-based detector. This approach is simple and attractive but is unsuitable for detecting small-scale anomalies since it can not leverage the full resolution of the frame.

4.2.2 Patch-Level Anomaly Detector

Patch-level anomalies are detected with a standard approach based on the reconstruction error of an autoencoder. The encoder part operates on a 64×64 input and is composed of four convolutional layers with a LeakyReLU activation function; each layer has a number of filters that is double the number of filters of the previous layer; we start with F 3×3 filters for the first layer. Each Convolution has stride 2 thus halving the resolution of the input. The neurons of the last layer of the encoder are flattened and used as input to a fully connected layer with B neurons (bottleneck); the decoder is built in a specular manner to the encoder, and its output has the same shape as the encoder's input; the output layer has a linear activation function, which enables the model to reconstruct the same range as the input. During inference, the patch-based anomaly detector accepts a patch as input and outputs the Mean Absolute Error between the input patch and its reconstruction, which we interpret as the patch anomaly score.

4.2.3 Training

For a given scale s, the autoencoder is trained as follows: first, we downsample each frame in the training set by a factor s; then, as an online data generation step, we sample random patches completely contained in the downsampled frames.

We use the Adam [45] optimizer to minimize the mean squared reconstruction error, with an initial learning rate of 0.001, which is reduced by a factor of 10 in case the validation loss plateaus for more than 8 epochs. Because the size of the training set of different datasets is widely variable, we set the total number of epochs in such a way that during the whole training, the model sees a total of 2 million samples; this allows us to better compare results on different datasets.

The approach is implemented in PyTorch and Python 3.8, using a deep learning workstation equipped with 4 NVIDIA 2080 Ti GPUs; training each model takes about 1 h on a single GPU.

Table 4.1. AUC values for models at all scales

Scale	**Avg**	Tunnels				Factory			Corridors								
	all	**all**	dust	root	wet	**all**	mist	tape	**all**	water	cellophane	cable	defect	hang. cable	floor	human	screws
S_8	**0.82**	**0.82**	0.54	0.76	0.87	**0.90**	0.95	0.48	**0.74**	0.63	0.66	0.70	1.00	0.44	0.85	0.67	0.48
S_4	**0.62**	**0.89**	0.62	0.79	0.94	**0.24**	0.25	0.17	**0.73**	0.81	0.70	0.81	1.00	0.41	0.38	0.73	0.30
S_2	**0.60**	**0.88**	0.63	0.80	0.93	**0.21**	0.20	0.30	**0.71**	0.78	0.70	0.75	0.99	0.50	0.51	0.56	0.40
S_1	**0.55**	**0.85**	0.61	0.80	0.88	**0.12**	0.10	0.25	**0.69**	0.72	0.73	0.68	0.90	0.60	0.59	0.51	0.55

4.2.4 Metrics

We evaluate the performance of the frame-level anomaly detector on the testing set of each dataset. In particular, we quantify the anomaly detection performance as if it was a binary classification problem (normal vs anomalous), where the probability assigned to the anomalous class corresponds to the anomaly score returned by the detector. This allows us to define the Area Under the ROC Curve metric (AUC); an ideal anomaly detector returns anomaly scores such that there exists a threshold t for which all anomalous frames have scores higher than t, whereas all normal frames have scores lower than t: this corresponds to an AUC of 1. An anomaly detector returning a random score for each instance, or the same score for all instances, yields an AUC of 0.5. The AUC value can be interpreted as the probability that a random anomalous frame is assigned an anomaly score larger than that of a random normal frame. The AUC value is a meaningful measure of a model's performance and does not depend on the choice of threshold.

For each model and dataset, we compute the AUC value conflating all anomalies, as well as the AUC individually for each anomaly (versus normal frames, ignoring all other anomalies).

4.3 Experimental Results

4.3.1 S$_8$ Model Hyperparameters

Figure 4.3a explores the choice of the bottleneck size B for model S_8. Increasing B reduces reconstruction error for both anomalous and normal data; the reconstruction error best discriminates the two classes (higher AUC, higher average gap between the two classes) for intermediate values of B (16 neurons): then, the Autoencoder can reconstruct well normal data while lacking the capacity to properly reconstruct anomalous samples. These findings apply to all three datasets. Figure 4.3b investigates a similar capacity trade-off: autoencoders with a small number of filters for the first convolution layer (*first layer size*) are not powerful enough to reproduce well even normal samples, therefore have lower discriminating performance. For the rest of the Section, we only report results for bottleneck size $B = 16$ and first layer size $F = 128$.

4.3.2 Patch Aggregation

Figure 4.3c:top explores the impact of N_p on the anomaly detection performance of model S_2; we observe that, for the Tunnels and Corridors scenarios, the performance increases as N_p increases. This is expected, as more patches are processed and aggregated to compute the frame-level score. Only for Tunnels, S_2 outperforms S_8 for 10 or more patches.

On the contrary, for the Factory scenario, the model S_2 performs worse than chance at detecting anomalies and assigns lower scores than normal data. this is due to the testing set being dominated by the mist anomaly, which is not detectable at low scales as discussed previously.

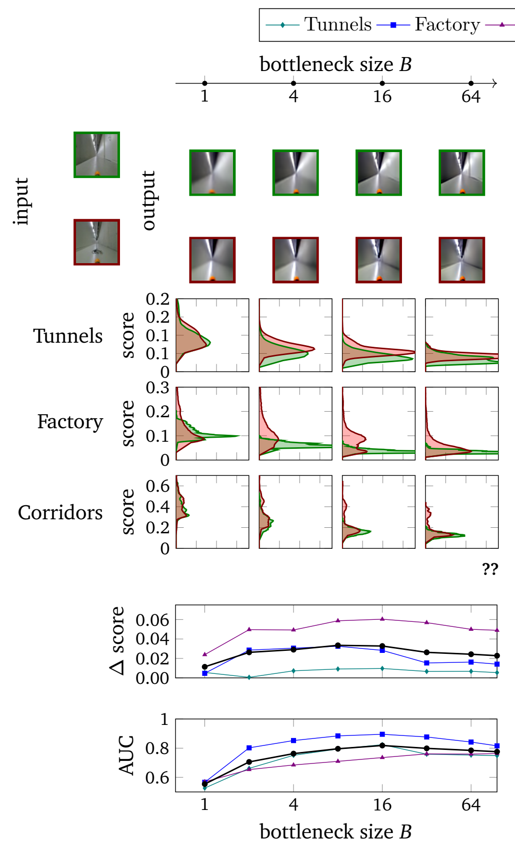
Figure 4.3c:bottom reports how computing the 0.7-0.8 quantile offers a slightly better aggregation than averaging.

4.3.3 Scales and Anomalies

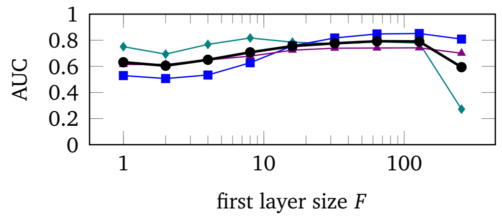
Table 4.1 summarizes the most important results on all model scales, scenarios, and anomalies. We note that most anomalies are best detected by the full-frame approach S_8; this is especially true for large-scale anomalies that cover a significant portion of the frame, such as mist for Factory, or human and floor for Corridors. In contrast, S_8 underperforms for small-scale anomalies, that cover few pixels of the downsampled image (e.g., dust and roots for Tunnels; cellophane, water, and hanging cable for Corridors); in this case, small-scale models sometimes have an advantage over S_8.

In contrast, we observe that small-scale models struggle with the large-scale mist anomaly, returning consistently lower anomaly scores than normal frames, which yields AUC values well below 0.5. Figure 4.4 compares how S_1 and S_8 reconstruct a mist frame: clearly, S_8 fails to capture the large-scale structure of mist, which yields high reconstruction error as expected in an anomalous frame; in contrast, since individual high-resolution patches of the mist frame are low-contrast and thus easy to reconstruct, the S_1 model yields very low reconstruction error and, thus, low AUC.

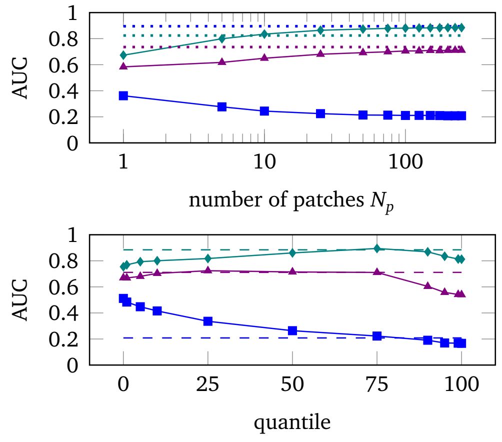
Some anomalies, such as defects for Corridors, are obvious enough that models at all scales can detect them almost perfectly.



(a) Results for different autoencoder's bottleneck sizes for model S_8. Top two rows: for the same two samples (normal in green, anomalous in red), autoencoder reconstructions. Center: score distributions over the testing set. Bottom: mean score difference between anomalous and normal samples and AUC of the anomaly detector.



(b) Results for model S_8 for Autoencoders with different first layer sizes.



(c) Results for model S_2 when aggregating the scores of multiple patches extracted from each frame. Top: AUC, when aggregating by averaging, for different numbers of patches, compared to S_8 (dotted). Bottom: AUC, when aggregating 250 patches by computing a quantile (solid) or by averaging (dashed).

Figure 4.3. Experimental results

4.3.4 Run-time Evaluation

A quadcopter runs the S_8 model to detect anomalies on board. In the experiment, the robot captures a camera frame, computes an anomaly score, and raises an alarm when the score passes a predefined threshold. The model successfully recognizes unforeseen hazards and allows the drone to avoid them on multiple runs. Figure 4.5 illustrates a representation of the execution of model S_8 on a sequence of frames that are part of the qualitative testing set for Factory. In the example shown in the Figure, the drone that collects the data is manually piloted, first, it traverses a long area of mist and later it stops in front of a small signaling tape. On all occasions, the model correctly identifies the anomalies. We manually annotate the ground truth presence of hazards such as mist (first red interval) and tape (second red interval) and the model predictions match these areas.

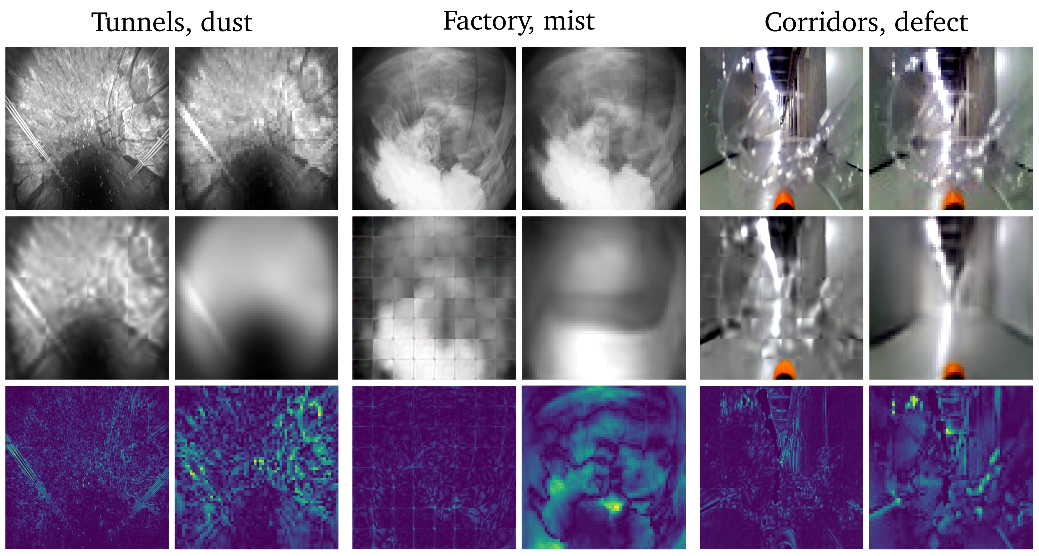


Figure 4.4. Comparison between S_1 and S_8: pairs of identical input images representing an anomaly (top row); autoencoder's outputs (central row) for S_1 (left) and S_8 (right); the absolute value of the difference between input and output (bottom row, using a colormap where yellow is high and blue is low). Only for this illustration, for S_1 we regularly sample 64 patches to cover the whole input image, and we use the output patches to compose a full-resolution image.

4.4 Demonstrators

Using our model we develop a hazard detection module to be deployed on a real robot. With this software, we built two separate demonstrators detailed below.

4.4.1 Wheeled Robot Demonstrator

The first demonstrators consist of the same DJI Robomaster S1 used for collecting the dataset in Chapter 3. We connect the robot to a nearby laptop to run the model while we are manually teleoperating the robot; if a hazard is detected, the control is taken away from us and the robot stops. For deploying the model on the laptop. we use a code that loads one of the trained models described before, and, at runtime, maps incoming images to a real-valued anomaly score. Additionally, we provide a graphical interface to inspect the inner workings of the system. We extensively use the GUI we developed to test the behavior of our ground robot equipped with the anomaly detector. In our tests when an anomaly (e.g., a pillow) is detected, the robot automatically stops moving and signals the anomaly using its LEDs while correctly removing the controls from the human operators. In the experiments, the system detects different anomalies and the robots always stop at a safe distance from the presumed hazards. This demonstrator was built during the Covid restriction and thus is only limited to houses indoors and household items as anomalies such as a bottle or a mug.

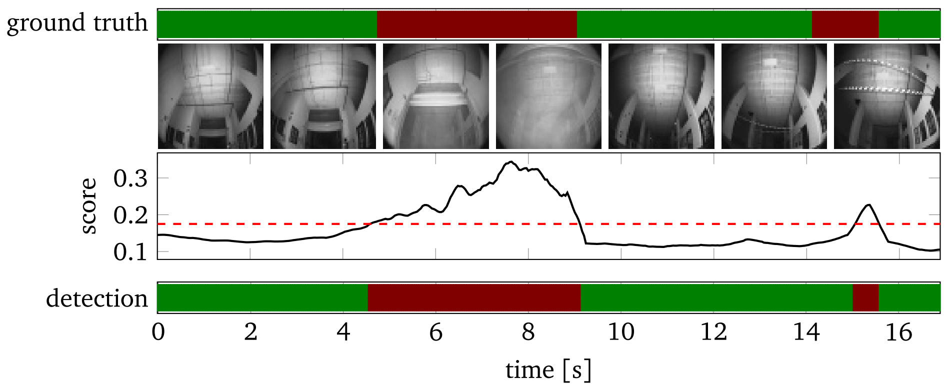


Figure 4.5. Part of the qualitative testing dataset in the Factory scenario where the drone first passes through the mist and then below a tape. Top: the manually added labels (green: normal, red: anomalous) and seven frames sampled from the timeline. Center: the score returned by the S_8 model (solid black) and anomaly threshold t (dashed red). Bottom: the anomaly detector output.

4.4.2 Drone in Factory Demonstrator

The second demonstrator is built in collaboration with our partner Hoovering Solutions which provided the drone, the facility, and the control system used for the reaction to hazard detection. For this demonstrator, we used the autoencoders discussed in this Chapter. With this demonstrator, we want to test the whole inference (and control) pipeline based on our partner's use case. We developed a Python package deployed using a Docker container, that exposes APIs that allow our partners to integrate the hazard detection model in the drone navigation system. The library was used by Hovering Solutions to perform the demo for the midterm review at M18 of the 1-SWARM EU Horizon project. In addition to the hazard detection models' API, we developed a GUI, visible in Figure 4.6. The GUI used provides feedback on what the model detects and can be used for debugging purposes; it is composed of the following elements:

- Three top images, from left to right, the input frame, the autoencoder's reconstruction, and the error map (built using the color-coded difference between the autoencoder's input and prediction)
- A history bar that shows how the past frames were identified in red (anomalous), and green (nominal); the most recent prediction lies on the right.
- A history plot of anomaly score (dots) per frame, filtered anomaly score (line), and anomaly threshold (red line); the most recent prediction lies on the right.

three top

In case an anomaly is detected the framing around the input images becomes red like in Figure 4.6. The demonstrator allowed a drone to fly unharmed across a testing environment with hazards. In this demonstrator, we deployed the models directly onboard the drone using an NVIDIA Jetson board as the the Hovering Solution drones are already equipped with such boards. NVIDIA Jetson is a family of single computer boards, particularly well suited for machine learning inference on embedded systems. We provide a setup guide for installing the library on these boards with or without docker. We tested the performances of the system on the NVIDIA Jetson board while running in tandem with Hovering Solution's custom navigation system; the autoencoder-based anomaly detector achieved a detection rate of 33FPS, reduced to 27 FPS when exposing a web-based GUI. In the demonstration, the drone flies towards some signaling tape stretched between a door and a pillar of the Factory scenario. As soon as the hazard is detected, the drone stops its mission and backtracks to safety. Figure 4.6 illustrates the detection of an unforeseen hazard during the demonstration.

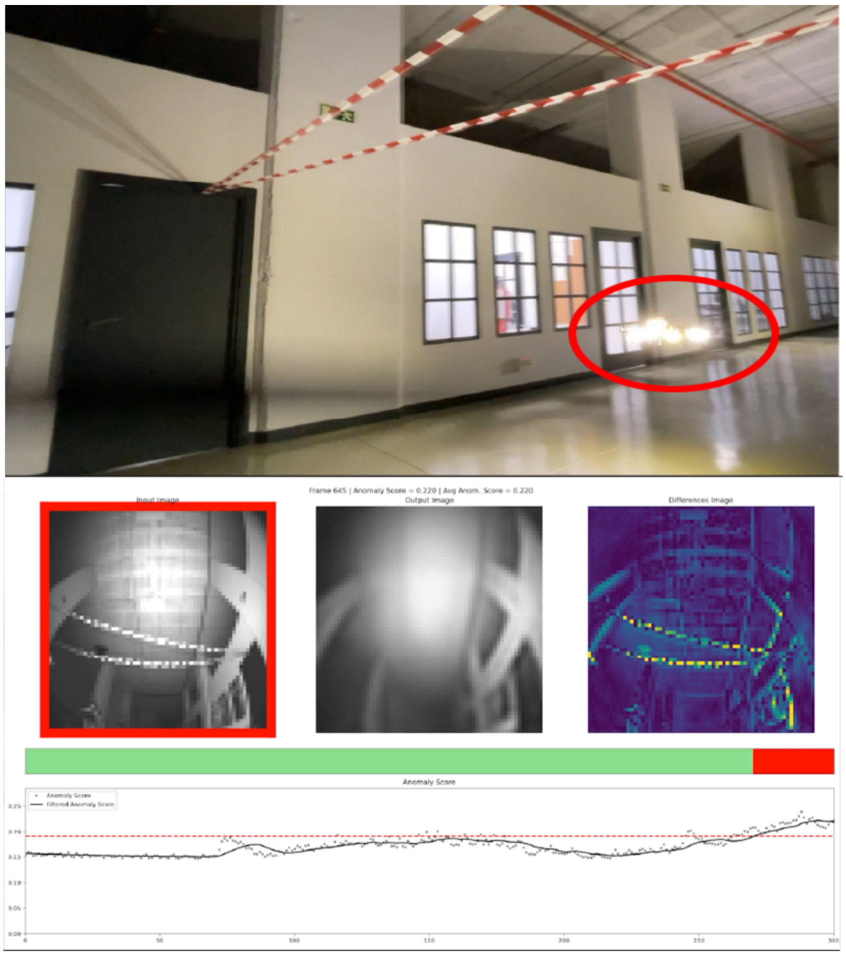


Figure 4.6. Deployment in the Factory scenario: the drone advances normally until it detects an anomaly (a tape crossing its path); a time series of anomaly scores in previous frames is reported at the bottom (see supplementary video https://youtu.be/SylhxUl20C0).

4.5 Conclusions

In this Chapter, we explored an autoencoder-based visual anomaly detection technique for detecting hazards in visual sensing data. The data is acquired by mobile robots freely exploring an environment. The autoencoder is trained only on normal samples from our dataset and is tested on a combination of normal and anomalous samples. Multiple anomalies with different sizes and characteristics are considered during the model testing. In this Chapter, we focus on how differently sized input affects the detection performance. Results show that the approach is successful at detecting most anomalies (detection performance with an average AUC metric of 0.82); nonetheless, detecting small anomalies is in general harder than detecting anomalies that affect the whole image. Given these results we successfully deployed our model on a ground robot and on a drone; then we tested its performance in real-time hazard detection in two demonstrations. In all cases, anomalies were correctly identified indicating the detection performance of our method.

In the next Chapter, we will modify this initial approach to be able to consider also anomalous samples during training

Chapter 5

Hazard Detection with Outlier Exposure

In the previous Chapter, we explored a visual anomaly detection technique based on autoencoders. We successfully trained and tested the model to detect hazards in visual sensing data acquired by mobile robots exploring an environment; the autoencoder was trained only on normal samples from our dataset. However, in robotics applications, it is often the case that (potentially very few) examples of anomalies are available. In this Chapter, we tackle the problem of exploiting anomalous samples to improve the performance of a hazard detection system. Our approach combines a Real-NVP model with an additional margin loss term, optimized jointly with the Real-NVP loss. Quantitative experiments on our dataset show that exposure to even a small number of anomaly frames significantly improves anomaly detection performance and that the proposed approach significantly outperforms alternative approaches. On a disjoint test set, our approach outperforms alternatives and shows that exposing even a small number of anomalous frames yields significant performance improvements.

5.1 Background

Anomaly detection techniques assume that a large amount of non-anomalous data is available (e.g., data from past patrolling missions); only these data are used to train a model. For a given new observation (e.g., a frame acquired by the robot camera), the model outputs an anomaly score, that quantifies how likely it is that the observation comes from a different distribution than the learned distribution of normal data. In real-world robotics scenarios, this approach is difficult to apply because normal data has a large variability and relevant anomalies might affect the image data in subtle ways (e.g., a puddle of water on the floor only affects a small part of the image). Absent a prior on the expected appearance of relevant anomalies, it is difficult to learn anomaly detection models that perform well enough to be useful in practice.

However, in most realistic use-cases one can expect that at least some examples of anomalies will be available when training the anomaly detection model, thus extending beyond the pure anomaly detection paradigm. For example, samples of possible anomalies might be collected during system design, and others after robot deployment and with partial human supervision. These examples might not represent all possible types of anomalies, and might be available in much smaller quantities than normal data. Still, they represent useful pieces of information that should be integrated into the model. To handle these additional training data, one baseline solution is to use the resulting dataset, composed of many normal and few anomalous samples, to train a binary classifier. This approach is suboptimal because the few available examples of anomalies do not cover the space of all possible anomalies.

The main contribution of this work is methodological: we combine Real-NVP [24], a state-of-the-art anomaly detection approach, with the concept of *Outlier Exposure*, which was recently proposed in the deep learning literature [36] to train more robust anomaly detectors by providing large amounts of examples of outliers, sampled from completely different datasets than the dataset of normal samples. In our context, we instead assume that the exposed anomalies are few and correspond to relevant anomalies for the robot. This is implemented by using an additional margin loss term during training of a Real-NVP model; this additional term enforces that the exposed anomalies are assigned higher anomaly scores than random normal samples, but does not imply any expectation that the exposed anomalies are representative of all anomalies. We evaluate the method on a new real-world dataset, which we release as supplementary material as a secondary contribution, acquired by a ground patrolling robot; experiments show that exposing even just a few anomalous frames significantly improves performance, comparing favorably with alternative approaches.

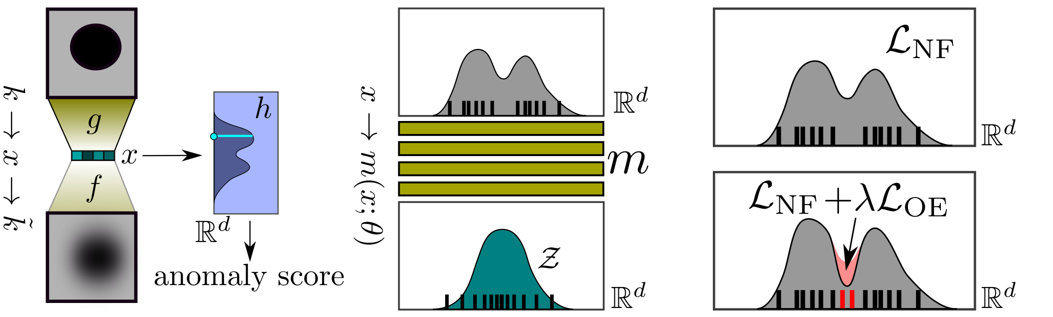
5.1.1 Models and Approaches

While already explained previously in this thesis, here we quickly recap the key components for this work.

Reconstruction-based methods A standard deep learning approach to visual anomaly detection uses undercomplete autoencoders [49, 18]. The typical architecture of autoencoders consists of two modules, the encoder, and decoder, separated by a bottleneck. The encoder and decoder are convolutional networks that, respectively, encode the input image into a low-dimensional embedding and decode the high-level informative embedding into an image. The bottleneck limits the embedding size thus inducing a reproduction error in the output. This error can be exploited to detect anomalies when an autoencoder is trained only on normal images. Such a model, when tasked to reproduce an image that contains an anomaly, will fail to correctly reconstruct it; the reconstruction error can therefore be used as an anomaly score for the given input.

Normalizing Flows Normalizing Flow models are used both for generative purposes [2, 47, 37] and in the context of anomaly detection [97, 10]. These models rely on coupling layers, invertible transformations that map the inputs to a chosen latent distribution, such as a normal distribution. For anomaly detection, one uses the learned mapping to directly estimate the likelihood of a sample with respect to the distribution of non-anomalous samples used for training. In this Chapter, we use Real-NVP [24], a type of Normalizing Flow model.

Outlier Exposure Our approach combines Real-NVP with the concept of Outlier Exposure [36]: training anomaly detectors against an auxiliary dataset of known outliers. In the existing literature, these known outliers are sampled from large benchmark datasets (e.g., 80 Million Tiny Images [91] or ImageNet-22K [23]), which contain a huge amount of data that is semantically



(a) Anomaly detector: the anomaly score is computed from the estimated likelihood of the encoding of image k. The auto-encoder is trained to reproduce $\tilde{k} \approx k$.

(b) Real-NVP learns the best model m that maximizes samples' likelihood by mapping them to a target distribution $\mathcal{Z}$ through a series of invertible coupling layers.

(c) Outlier exposure: (above) the original Real-NVP loss does not consider anomalous samples; (below) adding an outlier exposure loss reduces the likelihood assigned to known anomalies.

Figure 5.1. The method to compute anomaly scores presented in Section 5.2. Gray plots in (b) and (c) represent probability densities over the encoding space, where dataset samples are drawn as small rectangles (normal in black, anomalous in red)

very different than in-distribution data; in this case, the approach helps the model to generalize and detect unseen anomalies. Anomaly detection tasks on real-robot datasets differ, since much smaller datasets are available, and anomalies can be semantically very similar to normal images.

5.2 Method

5.2.1 Problem Statement and Definitions

Our goal is to decide, at run-time, whether the current camera frame k is anomalous or not. We do this by training machine-learning models on a dataset $\mathcal{K}$ of previously acquired images. The dataset is composed of two disjoint subsets: $\mathcal{K}_n$, with images labeled as normal, and $\mathcal{K}_a$, generally much smaller, with images labeled as anomalies.

The anomaly detector takes as input an image and outputs a real-valued anomaly score. Following a common approach [97], the anomaly detector consists of two parts: dimensionality reduction and density estimation (see Fig. 5.1a).

Dimensionality reduction We train an autoencoder $f \circ g$ to reconstruct images in $\mathcal{K}_n$. Once trained, we use the encoder to compute lower-dimensional representations $x = g(k) \in \mathbb{R}^d$. In particular, we obtain two datasets $\mathcal{X}_n = g(\mathcal{K}_n)$ and $\mathcal{X}_a = g(\mathcal{K}_a)$. Adding samples to $\mathcal{K}_a$ does not require re-training the encoder; in fact, for this work, the encoder has been trained once and then shared by the different anomaly detectors.

Density estimation A second machine-learning model $h(\cdot;\theta)$ estimates the probability density of the distribution $\mathcal{D}_n$ from which $\mathcal{X}_n$ has been sampled. Parameters θ are trained on both $\mathcal{X}_n$ and $\mathcal{X}_a$.

Once both models have been trained, we define the anomaly score of an image k as the negative log-likelihood of its encoding, assuming it belongs to $\mathcal{D}_n$, i.e.:

$$-\log h(g(k);\theta) \in \mathbb{R}\,. \tag{5.1}$$

We devote the rest of the section to describe the density estimator h: a variant of Real-NVP that additionally takes known anomalies into account.

5.2.2 Real-NVP

Normalizing flow models estimate the density of a distribution $\mathcal{D}_n$ from a set of samples. They are based on learning the parameters θ of a bijection $m(\cdot;\theta) : \mathbb{R}^d \rightarrow \mathbb{R}^d$ that maps $\mathcal{D}_n$ to a known target distribution $\mathcal{Z}$. Following a change of variables, the probability density of the original samples is recovered from the target probability density as:

$$p(x|\theta) = p_{\mathcal{Z}}(m(x;\theta)) \left|\det\left(\frac{\partial m(x;\theta)}{\partial x}\right)\right|\,. \tag{5.2}$$

For the rest of the Chapter, we fix the target to be a multi-variate normal distribution of zero mean and unit variance $\mathcal{Z} = \mathcal{N}(0, \mathbb{1})$.[1]

Normalizing flow models are trained to maximize the likelihood of the samples in $\mathcal{X}_n$, or equivalently to minimize their average negative-log-likelihood loss function:

$$\begin{aligned} \mathcal{L}_{\mathrm{NF}}(x) &= -\log(p(x|\theta)) \qquad (5.3)\\ &= -\log(p_{\mathcal{Z}}(m(x;\theta))) - \log\left(\left|\det\left(\frac{\partial m(x;\theta)}{\partial x}\right)\right|\right)\,. \end{aligned}$$

While the first term is easily computed, and equals to $\frac{d}{2}\log(2\pi) + \frac{1}{2}|m(x)|^2$ for the chosen $\mathcal{Z}$, clever solutions are required to keep the second term tractable. We make use of one of them: Real-NVP, a deep-learning normalizing flow model [24], where m consists of a sequence of *affine coupling layers* designed to simplify the computation of the determinant of the derivative (see Fig. 5.1b). Each coupling layer contains parametric translation and scaling on a subspace of $\mathbb{R}^d$, modeled as a dense neural network.

Once the model is trained, we can infer the negative-log-likelihood of a sample, which is then used as an anomaly score:

$$-\log h(x) = \mathcal{L}_{\mathrm{NF}}(x)\,. \tag{5.4}$$

5.2.3 Real-NVP With Outlier exposure

Density estimation through Real-NVP makes no use of known anomalous samples. To solve this limitation, we combine it with outlier exposure.

The goal of outlier exposure is to improve the performance of an anomaly detection model m by exposing it to samples $\mathcal{X}_a$ drawn from an outlier distribution $\mathcal{D}_a$, using an additional loss $\mathcal{L}_{\mathrm{OE}}$ applied to pairs of samples $(x_n, x_a) \in \mathcal{X}_n \times \mathcal{X}_a$. In its generic form, the model should now minimize a weighted sum of the original loss (in our case $\mathcal{L}_{\mathrm{NF}}$) and of $\mathcal{L}_{\mathrm{OE}}$:

$$\langle \mathcal{L}_{\mathrm{NF}}(x_n)\rangle_{\mathcal{X}_n} + \lambda \langle \mathcal{L}_{\mathrm{OE}}(x_n, x_a)\rangle_{\mathcal{X}_n \times \mathcal{X}_a}\,, \tag{5.5}$$

[1]This approach is known as Gaussianization [17] when $\mathcal{Z}$ is a normal distribution.

where $\langle\cdot\rangle_{\mathcal{X}}$ denotes the average over a set $\mathcal{X}$.

Following [36], we define $\mathcal{L}_{\mathrm{OE}}$ as a margin ranking loss:

$$\mathcal{L}_{\mathrm{OE}}(x_n, x_a) = \max(0, \gamma + \mathcal{L}_{\mathrm{NF}}(x_n) - \mathcal{L}_{\mathrm{NF}}(x_a)) \, , \tag{5.6}$$

where γ represents the margin hyperparameter; $\mathcal{L}_{\mathrm{OE}}$ is added to $\mathcal{L}_{\mathrm{NF}}$ loss as in Eq. (5.5). The resulting loss, besides maximizing the likelihood of normal samples, further encourages the model to assign a higher likelihood to normal samples than to known anomaly samples (see Fig. 5.1c).

5.3 Experimental Setup

In the work of this Chapter we use v2 of our dataset as detailed in Section 3.6.3, it contains two additional scenarios of drones flying in indoor environments, which are not used in this Chapter. The dateset allow us to create all the sets needed for our experiments, namely T_{train} composed of k_n samples, T_{val} composed of k_n samples, T_{out} composed of k_a samples, and T_{test} composed of both k_n and k_a samples; note that the T sets are all disjoints.

5.3.1 Autoencoder: Reconstruction and Dimensionality Reduction

We train an undercomplete autoencoder to detect anomalies via the reconstruction method (AE). The autoencoder implements a simple convolutional architecture often used for similar purposes [31, 33]: it takes as input a 64×64 RGB image; compresses it to a 128-dimensional representation; then reconstructs it as a 64×64 RGB image as output. The model is trained on normal samples of the training set by minimizing the mean squared error loss between input and output. At inference time, the mean squared reconstruction error of a given frame is used as an anomaly score.

The same autoencoder is also used for dimensionality reduction: the trained encoder g extracts 128-dimensional visual features from input images, which are used by the approaches described in Section 5.3.2 (see Fig. 5.1a).

Fig. 5.2 illustrates the model architecture: both the encoder and the decoder consist of four convolutional layers interleaved by LeakyReLU activation functions; the last activation function of the decoder is linear. The encoder convolution layers have stride 2 thus halving the input size, while before each decoder convolutional layer the input size is doubled. The bottleneck module is composed of two dense layers that take the output of the encoder and compress it to a 128-dimensional vector.

The autoencoder is trained for 100 epochs using the Adam [45] optimizer, with a learning rate of 0.001 and a reduction factor of 10 in case of a validation loss plateau. During training, samples are randomly augmented with horizontal flipping, rotation ($\pm 10°$), cropping, contrast and brightness variations, and mild additive Perlin noise [72]. The augmentation pipeline is provided as supplementary material.

5.3.2 Density Estimation (RNVP, RNVP+OE)

We detect anomalies using Real-NVP density estimators; we separately report results without outlier exposure (RNVP, Section 5.2.2) and with outlier exposure (RNVP+OE, Section 5.2.3). Both models share the same architecture: they take a 128-dimensional image encoding as input

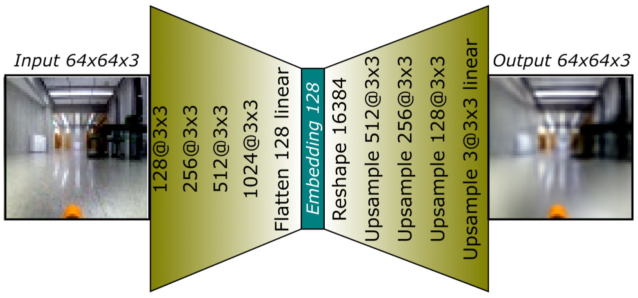


Figure 5.2. Autoencoder architecture. Encoder layers (left) use a stride of two. Inputs of decoder layers (right) are up-scaled before convolutions. When not specified, layers use a LeakyReLU activation function.

(see Fig. 5.1b), and output a 128-dimensional vector. The models are composed of four coupling layers whose scaling and translation modules have a single hidden layer with 128 neurons, using *odds* input masking.

The RNVP model is trained by minimizing the loss defined in Eq. 5.3; the RNVP+OE model is trained by minimizing the loss defined by Eq. 5.5 and Eq. 5.6, where we set $\lambda = 1$ and $\gamma = 100$. For each experimental run, both models are trained from scratch, with randomly initialized weights, for 500 epochs using Adam [45] with a starting learning rate of 0.001. We reduce the learning rate by a factor of 10 in case the validation loss plateaus for more than 10 epochs. For all models, the checkpoint that minimizes the validation loss (which only accounts for normal samples) is selected.

For inference, both models use the same anomaly score defined in Eq. (5.4).

5.3.3 Binary Classifier Baseline (BCLASS)

We compare RNVP and RNVP+OE with a neural network binary classifier (BCLASS) that operates on the same 128-dimensional image embedding as input. The model c is composed of three fully connected layers (with 256, 64, and 1 neurons respectively) interleaved by a ReLU activation function; a Sigmoid activation function is used at the output.

The model is trained on $T_{train} \cup T_{out}$ using the Binary Cross Entropy loss to estimate whether the input is an anomalous frame. The rest of the training details are shared with RNVP and RNVP+OE. For inference, a given frame k is assigned an anomaly score $c(g(k))$ corresponding to the output of the trained classifier.

5.3.4 Wide Residual Networks with Outlier Exposure (WRN+OE)

In addition to BCLASS, we also compare to Wide Residual Networks (WRN+OE) [99], leveraging an architecture of size 16-2 and the implementation used by Hendricks et al. [36] for the SVHN dataset [67], with the hyperparameters and augmentation used for training the WRN on Tiny ImageNet [53]. To make the approaches comparable, in the training and validation sets we included a new 3-class label representing the environment where the samples were recorded; this label is used only during training and is ignored for testing. For inference, we use the Maximum Softmax Probability (MSP) [35] over the WRN's output to detect anomalies [36]. The approach is thus trained to classify which of the three environments each sample was acquired in; the uncertainty in the prediction is used as an anomaly score of the sample.

5.3.5 Metrics

We evaluate each model on the same testing set, which is composed of both normal and anomalous samples. Also in this work we use the AUC to measure the predictive performance of the anomaly detection models.

5.3.6 Computational Costs

Experiments are run on an NVIDIA 2080 Ti using Python 3.8 and PyTorch 1.7.1. On this platform, training time amounts to 100 minutes for training AE, 20 minutes for training the RNVP model, and BCLASS, 60 minutes for WRN+OE (over 20 epochs but converged after 4), and on average 30 minutes for RNVP+OE. Inference time amounts to 1 ms for BCLASS, 2 ms for WRN+OE, and 5 ms per frame for both RNVP and RNVP+OE.

5.4 Experimental Results

In this section, we report the results of an experimental investigation on the performance of RNVP+OE on the dataset described in Section 3. As an initial step, we encoded all images in the dataset using the autoencoder's encoder, which yields a dataset of embeddings; this dataset is used in all RNVP, BCLASS, and RNVP+OE experiments.

5.4.1 Hyperparameter Exploration

We explored various hyperparameters of autoencoder, RNVP, BCLASS, WRN, and RNVP+OE. Most of the hyperparameters were set using a preliminary search or were inspired by previous research [97, 24, 36]: the autoencoder's bottleneck size, learning rate, maximum number of epochs, input size and architecture details; the RNVP input size, learning rate, coupling layer size and number, and input masking; WRN's architecture and hyperparameters. For all experiments except AE and WRN+OE, we let the model run for 500 epochs, then choose the best performing model over the validation set. For WRN+OE we observed that after 4 epochs the model converged, thus we let the model train for at most 20 epochs with early stopping.

Fig. 5.3 illustrates how we selected parameters λ and γ in Eq. (5.6) and Eq. (5.5), which are specific to our proposed contribution. Fig. 5.3:left shows the impact of λ, for fixed $\gamma = 0$, on 10 experiments for each value: we select $\lambda = 1$ as the best parameter. Similarly, Fig. 5.3:right shows the impact of γ for $\lambda = 1$ on 10 experiments for each value: we select $\gamma = 100$ as the best parameter.

5.4.2 Comparison with Baselines

We compare the performance of RNVP+OE with the baselines AE, WRN+OE, RNVP and BCLASS. RNVP+OE, WRN+OE and BCLASS are trained on the full training set, RNVP only on the normal samples of the training set and AE on the normal frames; all are tested on the same testing set (AE is tested on the corresponding frames). For RNVP, RNVP+OE, and BCLASS, we run the experiment 10 times to reduce the noise originating from the stochastic training of the models.

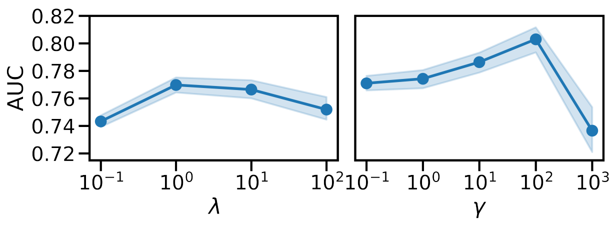


Figure 5.3. The impact of parameters λ and γ (x-axes) on the performance of the model measured with the AUC (y-axis)

AE and WRN+OE are trained only once due to the greater computation requirements. For the AE we report two AUCs based on two different anomaly scores, Mean Absolute Error (MAE) and Mean Squared Error (MSE) over the reconstruction error.

Table 5.1. AUC values for our model and baselines (* indicates average over 10 runs)

no OE			with OE		
AE-MAE	AE-MSE	RNVP	BCLASS	WRN+OE	RNVP+OE
0.64	0.67	0.73*	0.73*	0.55	**0.80***

Results Table 5.1 reports the AUC of all baselines. WRN+OE (AUC=0.55) yields a low performance on the task, which is probably caused by the fact that many considered anomaly types do not hinder the prediction of the environment in which the sample is acquired. Despite not using exposed outliers for training, AE yields a better performance both when using the reconstruction MAE (AUC=0.64) and MSE (AUC=0.67) as anomaly scores. BCLASS (AUC=0.73) and RNVP (AUC=0.73) show promising performance on the dataset. Finally, RNVP+OE has a significantly better performance than all alternatives (AUC=0.80).

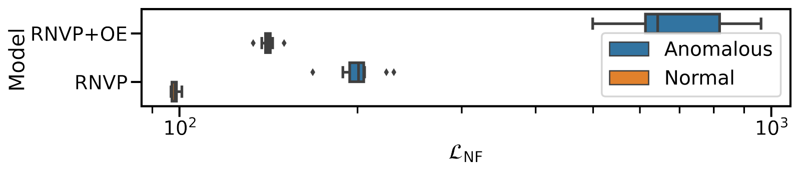
5.4.3 Effect on the RNVP Target Space

We investigate the effect of exposing RNVP to anomalies.

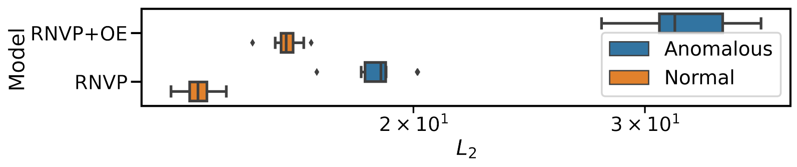
Results Fig. 5.4 shows how the additional loss defined in Eq. (5.6) increases the negative-log-likelihood of anomalous testing samples, leading to a wider separation from normal samples (Fig. 5.4a). Using outlier exposure on Real-NVP, anomalous samples are mapped further away from the origin (i.e., the mean of the target multivariate normal distribution) by RNVP+OE than by RNVP (Fig. 5.4b).

5.4.4 Impact of the Number of Exposed Anomaly Frames

Our contribution targets situations where limited anomaly samples are known. Therefore, we measure how much the number of anomaly samples available at training impacts the performance of RNVP+OE and BCLASS on the testing set, starting from just a few samples. We test



(a) The loss function $\mathcal{L}_{\text{NF}}$, i.e., the negative-log-likelihood, evaluated over the testing set. Larger values mean less likely.



(b) The Euclidean norm of transformed samples $m(x)$ evaluated over the testing set.

Figure 5.4. The effect on loss function and L_2 norm of exposing the model to anomalies during training.

exposed outlier set sizes ranging from 2 to 16384 samples; for each size, we train both models 10 times: for each run, we randomly sample a subset of the corresponding size from the exposed outliers in the training set. Note that RNVP+OE trained on 0 anomalies is equivalent to RNVP.

Results Fig. 5.5 illustrates that, as expected, AUC increases when training with more exposed anomaly frames. While both models exhibit an increase in performance by seeing more anomalies during training, RNVP+OE always outperforms BCLASS. For RNVP+OE, we observe that the performance tends to saturate after 1024 outlier frames are exposed. Interestingly, even when RNVP+OE is exposed to just a few anomalies (AUC=0.75 for $N = 64$), there is a noticeable increase in performance,while BCLASS performs significantly worse than RNVP for $N \leq 64$.

5.4.5 Impact of the Number of Exposed Anomaly Types

As described in Section 3, the dataset set is composed of different types of anomalies. We want to explore the impact of the heterogeneity of the training set by varying the number of types of anomalies available for training an RNVP+OE model. Using the complete training and testing sets, for $N = 1, 2, 3, 4, 6, 12$ types of anomalies, we run the following experiment 30 times: (1) we randomly pick a subset of N anomaly types; (2) we train RNVP+OE exposing it only to the selected anomaly types; (3) we compute the AUC over the whole testing set and also over the testing set limited to or excluding the anomaly types used for training.

Results Fig. 5.6 illustrates how the AUC varies over the different subsets of the testing set (with all anomaly types, only with the anomaly types exposed in training, only with anomaly types not exposed in training). We observe that exposing additional anomaly types does not

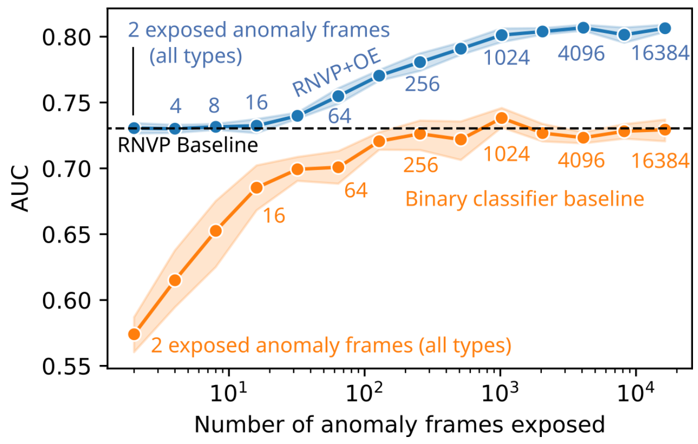


Figure 5.5. Impact of the number of exposed anomaly frames (x-axis, log scale) on the AUC value (y-axis) for the RNVP+OE model (blue) and the binary classifier baseline (orange); line and shaded area correspond to the mean and 95% c.i. over 10 runs of each model.

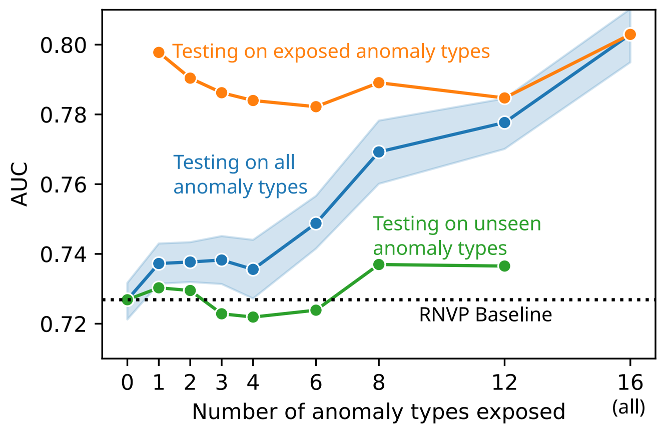


Figure 5.6. Performance (y-axis) for RNVP+OE models (colored lines), when trained with an increasing number of anomaly types exposed (x-axis). Performance is reported for: all anomaly types (blue); only the exposed anomaly types (orange); only the not-exposed anomaly types (green). Points correspond to the mean over 10 runs of each model; the shaded blue area represents the 95% c.i..

hinder the detection of non-exposed anomaly types (green line): we notice a small improvement when more than 6 types are exposed. The most significant improvement is on already seen types of anomalies (orange line), where the performance does not change significantly when the exposed anomalies are more heterogeneous (i.e., more types are exposed). As expected, similar to the previous experiment, the more anomalies are exposed, the better the model performs on the whole testing set (blue line).

5.5 Conclusion

In this Chapter, we considered the problem of improving visual anomaly detection systems for mobile robots, in situations where a limited amount of examples of some anomalies is available. Our approach combines a Real-NVP model with an additional outlier exposure loss. Quantitative experiments over 16 different anomaly types show that our approach is effective even with few anomaly samples, improving significantly the detection performance. Our proposed method paves the way for further research aimed at the safe autonomous operation of mobile robots in unstructured, unpredictable environments. In this Chapter, we assumed that we had already a selection of labeled anomalous samples to be used for training our model; in reality, the data collected from a robot deployed in a real application would be massive and not precisely labeled. In the next Chapter, we tackle this problem by proposing two new active learning queries.

Chapter 6

Active Learning Approaches for Anomaly Detection

As explained in the Introduction, during the continued use of a robot with a hazard detection system, we would collect thousands of samples containing mostly normal samples and some anomalies. If we want to exploit those anomalous samples using the approaches introduced in the previous Chapter, we would need to have a human expert to check and label all samples; this approach is inefficient. In this Chapter, we propose to use active learning to select informative samples to be labeled by an expert and further improve the detection system performance. We benchmark 8 different active learning strategies, of which two are novel; our results show that our approach has the best performance overall, but choosing the right query strategy strongly depends on external factors.

6.1 Background

Industrial use of autonomous robots for inspection or patrolling is increasing; more robots are now facing unexpected unknown events in new environments. At the same time, robust sensors and deep learning models allow a better understanding of a robot's surroundings. In this context, Anomaly Detection has seen a new breath of research [97, 10, 60] focused on robotics applications, becoming an important feature of a vision system for robots. One overlooked question regards the improvement of such models deployed on robots using data collected over time.

In other fields a tried approach is to use active learning to efficiently use data collected for model performance improvements [84, 86]; to the best of our knowledge, active learning has never been applied to a robot's Visual Anomaly detection system.

We focus on a specific scenario, an autonomous robot that explores underground tunnels where communication with the outside world is not available. While we expect the tunnels to be traversable, unexpected events can happen at any time; these events can become hazardous, impairing the robot or destroying it altogether. We emulate this scenario using our dataset, see Chapter 3. Similarly to recent research [97, 19], the robot uses a visual anomaly detection deep learning model to detect anomalies in frames coming from the front-facing camera. In the scenario, we expect the training data to be limited to a set of nominal frames of the first safely traversable meters of a tunnel.

To circumvent the limited availability of training samples, we propose to apply active learning to the data collected during the mission and carefully ask an expert to label only the most informative frames; then add these labeled samples, retrain the detector, and measure its performance using AUC and Average Precision on the testing set.

Our contribution is a benchmark of 8 active learning approaches to see which is the most efficient, from a sample labeling point of view, at improving the AUC and Average Precision. Our second contribution is two new query strategy approaches that exploit our model's prediction.

6.1.1 Active Learning

Active learning is an iterative process that aims at finding efficient ways to improve model performance by adding selected samples from an unlabeled set $\mathcal{U}$ to the training set $\mathcal{T}$. In active learning approaches the model selects the K unlabeled samples to be labeled by an expert/oracle. For us, $\mathcal{U}$ is the data collected during missions. Settles [84] identify three settings that define the general query strategies one can use.

Membership Query Synthesis - Introduced by Angluin et al[3], in this setting the model asks generation of new samples that are not part of $\mathcal{T}$; the samples are then labeled by an oracle, but it could happen that the generated samples were unintelligible for a human expert [5]. New samples are generated by Generative Adversarial Networks *GANs* [32, 11].

Stream-Based Selective Sampling - Here it is assumed that continuous inexpensive or free streams of data are available [21]. After collection, the learner decides for the annotation or not. Labeling is made in isolation for each sample; thus we cannot exploit contextual information for choosing which samples to annotate.

Pool-Based Sampling - In this scenario the samples are selected from a large pool of unlabeled data $\mathcal{U}$ [55]. Using an acquisition function the learner makes a greedy selection of samples to be labeled by the oracle; the learner can exploit the sample surroundings and neighborhood to make a decision. Pool-based sampling requires evaluation of each sample of $\mathcal{U}$; this process can become expensive, however, it has become the most studied approach for real-world scenarios [84, 11, 93, 89, 52].

6.1.2 Active Anomaly Detection

While there is a plethora of research on active learning [84, 11] and even recent efforts on active learning for anomaly detection on small-scale tabular data [100, 93, 89, 92], no research exists at the intersection of visual deep anomaly detection and active learning in a robotic context.

6.2 Method

We aim to compare, under a set of realistic assumptions, 8 different approaches of active learning to improve the performance of an anomaly detection system of a robot. We compare approaches with an automated pipeline that trains a custom Real-NVP model on the image's feature embeddings. The embeddings are part of our dataset, see Chapter 3 for studying visual anomaly detection in the robotic context; the embeddings are extracted from images by a pretrained CLIP [73] model. To trace the performance of the detector we use AUC and AP over the test set of the dataset.

6.2.1 Anomaly Detection

The embedding we use are 512-sized vectors; these are passed in input to our custom Real-NVP. The model learns to map the vectors to latent multivariate Normal distribution with the same dimensions. There we measure the probability of a sample being normal or anomalous. The model setup is similar to [60] but adapted to the larger embedding size; both the input and hidden layers have 512 neurons. For the outlier exposure loss($\mathcal{L}_{OE}$) component we use the original paper values of $\lambda = 1$ and $\gamma = 100$.

6.2.2 Active Learning

Assumptions

We follow [93] and set the following assumptions for our benchmark.

General Assumptions - We focus on label *feedback* with a fixed *budget* of 64 samples to be *interpreted* by our perfect oracle.

Specific Assumptions - For *class distribution* we assume that anomalies are unusual observations with no underlying distribution. Our *learning objective* is to improve the AUC and Average Precision of our model on a test set. Our *initial pool* is an unlabeled pool of assumed nominal samples.

Query strategies Overview

Trittenbach et al. [93] in their overview, propose a categorization of query strategies based on the *informativness* sources: *data-based* use statistical approaches on the samples, *model-based* rely on functions based on the learned model and, *hybrid* strategies that combine the two above; we propose an extension to the *data-based* category to consider strategies that extract information from the data itself (i.e. using spatial or geometrical [6] information).

We implement 8 different strategies: three are model-based (DB, HC, IHC), two are data-based (MM CLIP, MM RNVP), two are hybrids (HMM, HMM RNVP) and one is a baseline (RAN).

Decision Boundary - We base our implementation of *Decision Boundary* (DB) on the marginal strategy defined by Zhang et al. [100]; they use Neural Autoregressive Flow (NAF) [39] to detect anomalies and to select new samples. In the paper, the samples chosen are those that are closer to a decision boundary set as the α-quantile ($\alpha \in [0.9, 0.95]$) of the log-likelihood distribution of samples generated with NAF. We avoid that and set the decision boundary as $b = \text{icdf}(\alpha)$ where α is set to 0.95, thus $b = 1.96$, and icdf is the inverse cumulative density function. To select samples u^*_{DB} to be labeled we use:

$$u^*_{DB} = \arg\min_{u_\mathcal{N} \in \mathcal{U}_\mathcal{N}} |(||u_\mathcal{N}||_2 - b)| \tag{6.1}$$

where $u_\mathcal{N}$ corresponds to the Real-NVP latent representation of an unlabeled sample.

High-Confidence - Another model-based strategy is *High-Confidence* [4](HC); HC selects the most anomalous samples of $\mathcal{U}$ as follows:

$$u^*_{HC} = \arg\max_{u \in \mathcal{U}} \mathcal{L}(u) \tag{6.2}$$

where $\mathcal{L}$ is the negative log-likelihood of the samples u of $\mathcal{U}$.

We also implemented *Inverse High-Confidence* (IHC) that consists in selecting the least anomalous samples from $\mathcal{U}$. Note that both HC and IHC in unsupervised anomaly detection are special cases of DB; HC is DB where the boundary is set at an infinite distance from the nominal distribution and IHC corresponds to DB with a boundary at the center of the distribution.

Minimax - Under the data-based category we implement two strategies that explore the data space; inspired by Sener [83], Minimax (MM) select those samples from $\mathcal{U}$ that have the largest minimum distance from any sample t of $\mathcal{T}$.

We apply MM both on the CLIP space (MM CLIP) and on the RNVP latent space (MM RNVP).

$$u^*_{MM} = \arg\max_{u \in \mathcal{U}} dist(u,t), \ \forall t \in \mathcal{T} \tag{6.3}$$

where $dist$ is the $\mathcal{L}_2$ distance in the chosen data-space.

Hybrid Minimax - We propose a novel active learning Pool-based hybrid query strategy called Hybrid Minimax.

Inspired by MedAL [86], it combines MM and HC. Differently than MedAL, we first select a set of $K \times m$ samples using MM, from these HC selects the K most anomalous samples. m is a hyperparameter and is set to 3; if set to 1 Hybrid Minimax would match MM on the same data space, on the opposite a large enough m would result in a Hybrid Minimax behaving like HC.

We propose two versions of Hybrid Minimax, depending on the data space of the first $K \times m$ samples selection; if selected in the CLIP-space we call it HMM, if the selection is made in the Real-NVP latent space then it is called HMM RNVP.

Our approach combines the coverage of distance-based selection with the informativeness of the model prediction, resulting in a different selection than the two parts alone.

Random - We also consider a naive baseline where K random samples are selected from the unlabeled set.

6.3 Experimental Setup

The dataset we use is our Hazards&Robots v3 described in Chapter 3. The dataset is split in disjoint sets T_{train} composed of k_n samples and acting as the $\mathcal{T}$ set of the active learning setting, $\mathcal{U}$ composed of k_n and k_a samples, T_{val} composed of k_n samples, and T_{test} composed of both k_n and k_a samples.

6.3.1 Implementation

We release the code to replicate our results[1]. We implemented everything using Python 3.8 and PyTorch 1.13 and we ran our experiment on an NVIDIA 2080 Ti.

Models Training - We run parallel training[2] of models on a single GPU; each model is trained for 100 epochs in 2 minutes. More than 6500 experiments are run amounting to 17 days of computation time.

Query Strategy Computation - For model-based query strategies, the computation is done on GPU, and predictions for the entirety of $\mathcal{U}$ take less than 1 second. For data-based query strategies, distance computation between data samples is done with Faiss [41] from FAIR. The library can compute distances on GPU using large tensors already in VRAM; the result is an almost instantaneous distance computation.

[1]code is available on GitHub https://github.com/idsia-robotics/ActiveAnomalyDetection

[2]using GNU Parallel [90]

6.3.2 Metrics and Tracking

Evaluation of each model detection performance is done over the dataset test set with the following metrics. As in the previous Chapters, also here we use the AUC as our metric. Following [89], in this work we also consider the Average Precision (AP) of a model; AP describes the precision-recall curve as the weighted average of precisions at consecutive thresholds using recalls of preceding thresholds as weights.

We trace both *progress curves* [93] of AUC and AP and their averages along 10 active learning cycles.

6.3.3 Experiment Run

An experiment is defined by a query strategy, an initial size of $\mathcal{T}$, and a $\mathcal{U}$ with a fixed percentage of anomalous samples. An experiment corresponds to 10 active learning cycles of a query strategy; the samples of $\mathcal{T}$, $\mathcal{U}$, and the u^* are traced. In a learning cycle we train and test the Real-NVP on $\mathcal{T}$ and the u^* samples selected before; then using the query strategy we select the next u^* from $\mathcal{U}$. We use four initial sizes for $\mathcal{T}$ (64, 512, 1024, 4096) and four percentage of anomalies in $\mathcal{U}$ (0%, 0.1%, 1%, 10%). We fix the number of samples chosen by the query strategies (K) to 64. These values are chosen to resemble realistic scenarios and other datasets [89].

Each experiment setup is run 5 times and each time the initial training sample, the anomalous one available in the unlabeled is randomly chosen and fixed for the rest of the run.

6.4 Experiment Results

6.4.1 Overall Performance

In Tables 6.2, 6.3 we collect the query strategies metrics averaged across learning cycles and experiment repetitions. We highlight the best performance of each query strategy in italics; we color grade the values across all experiments where the better the performance the darker the color, note that the grading is also local to each subset of runs relative to a certain percentage of anomalies. In Table 6.1 we report the average performance of a Real-NVP model trained only on the initial $\mathcal{T}$. While not reported in a Table, on average a model that sees the whole $\mathcal{T}$ and $\mathcal{U}$ sets achieve AUC 0.960 and AP 0.982.

In terms of AUC and AP, the most performing query strategy is HMM (AUC 0.951, AP 0.978).

6.4.2 Initial Training Size and Unlabeled Anomaly Presence

When data is scarce, i.e few samples are available initially or anomalies are scarce in $\mathcal{U}$, it is better to explore the unlabeled set than to exploit what is learned; explorative strategies such as MM CLIP, MM RNVP, and HC lead to higher metrics than exploitative ones like DB and HMMs; note that if the initial size is very limited (64) the best strategy is RAN. Conversely, large initial $\mathcal{T}$ or high percentage of anomalies in $\mathcal{U}$, lead to better performances from hybrid queries(HMM, HMM RNVP). Since DB and IHC do not improve the performance of the model over the initial model (Table 6.1 we will exclude these two query strategies for the rest of the experiment analysis. We assume that the decision boundary of DB is set regardless of the learned model,

Table 6.1. AUC and AP of the initial models trained only on the initial training size.

Initial Size	64	512	1024	4096
AUC	0.9126	0.9342	0.9390	0.9434
AP	0.9583	0.9693	0.9719	0.9741

Table 6.2. Detailed analysis of average AUC performance

Initial Size				64				512				1024				4096
% of Anom in $\mathcal{U}$	0.0	0.01	1.0	10	0.0	0.01	1.0	10	0.0	0.01	1.0	10	0.0	0.01	1.0	10
DB	0.912	0.914	0.913	0.914	0.934	0.934	0.933	0.934	0.938	0.938	0.939	0.939	***0.943***	0.943	0.943	0.943
HC	0.935	0.936	0.925	0.913	0.943	0.943	0.938	0.935	0.944	0.944	0.942	0.939	0.946	***0.946***	0.945	0.943
IHC	0.911	0.914	0.913	0.914	0.934	0.934	0.933	0.934	0.938	0.938	0.939	0.939	0.943	0.943	***0.943***	0.943
MM CLIP	0.936	0.936	0.927	0.913	0.943	0.943	0.938	0.935	0.945	0.944	0.942	0.939	0.946	***0.947***	0.945	0.943
MM RNVP	0.935	0.936	0.925	0.913	0.943	0.944	0.937	0.935	0.944	0.945	0.942	0.939	0.946	***0.946***	0.944	0.943
HMM	0.911	0.915	0.926	0.924	0.933	0.935	0.945	0.947	0.938	0.938	0.948	0.949	0.943	0.943	0.950	***0.951***
HMM RNVP	0.910	0.914	0.925	0.922	0.933	0.934	0.943	0.946	0.938	0.937	0.946	0.949	0.943	0.943	0.947	***0.950***
RAN	0.936	0.936	0.937	0.940	0.942	0.942	0.942	0.945	0.944	0.943	0.945	0.946	0.946	0.946	0.946	***0.947***

Table 6.3. Detailed analysis of average AP performance

Initial Size				64				512				1024				4096
% of Anom in $\mathcal{U}$	0.0	0.01	1.0	10	0.0	0.01	1.0	10	0.0	0.01	1.0	10	0.0	0.01	1.0	10
DB	0.958	0.959	0.958	0.959	0.969	0.970	0.969	0.970	0.971	0.971	0.972	0.972	***0.974***	0.974	0.974	0.974
HC	0.971	0.971	0.965	0.959	0.974	0.975	0.972	0.970	0.975	0.975	0.974	0.972	0.976	***0.976***	0.975	0.974
IHC	0.958	0.960	0.958	0.959	0.969	0.970	0.969	0.970	0.972	0.971	0.972	0.972	0.974	***0.974***	0.974	0.974
MM CLIP	0.971	0.971	0.966	0.959	0.974	0.975	0.972	0.970	0.975	0.975	0.974	0.972	0.976	***0.976***	0.975	0.974
MM RNVP	0.971	0.971	0.966	0.959	0.974	0.975	0.971	0.970	0.975	0.975	0.974	0.972	0.976	***0.976***	0.975	0.974
HMM	0.958	0.960	0.966	0.964	0.969	0.970	0.975	0.975	0.971	0.971	0.977	0.977	0.974	0.974	0.978	***0.978***
HMM RNVP	0.958	0.960	0.966	0.963	0.969	0.970	0.974	0.975	0.971	0.971	0.976	0.976	0.974	0.974	0.976	***0.977***
RAN	0.970	0.970	0.971	0.973	0.973	0.973	0.973	0.975	0.974	0.974	0.975	0.975	0.975	0.975	0.975	***0.976***

thus not providing a meaningful selection of samples; instead, IHC by sampling only the least anomalous data does not provide useful information to the model.

6.4.3 Anomaly Selection

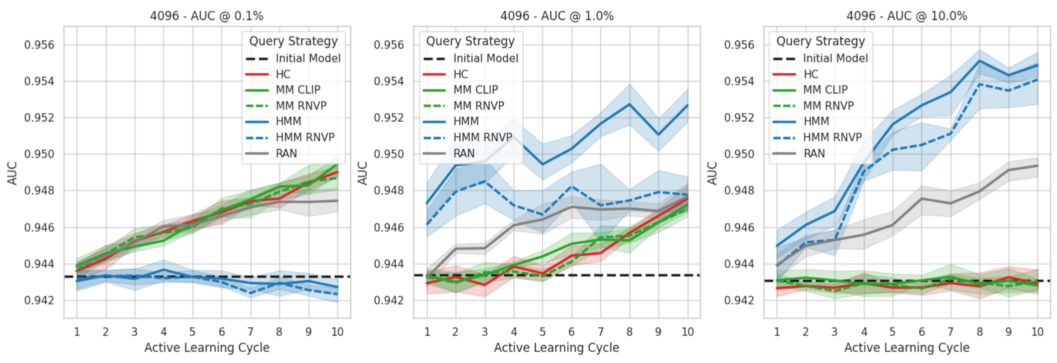


Figure 6.1. Progress curves of query strategies with 4096 initial samples.

In Figure 6.1 we plot the progress curves of HMMs, MMs, HC, and RAN. In Figure 6.2 we see the percentage of anomalies selected by each query strategy along the learning cycles. In Figure 6.3 instead, we see the number (variety) of anomaly classes in selection. For these Figures we focus only on the 4096 initial size. From the Figures, we observe that the learning model benefits from selecting anomalies; in all percentages of Figure 6.2, the model with a

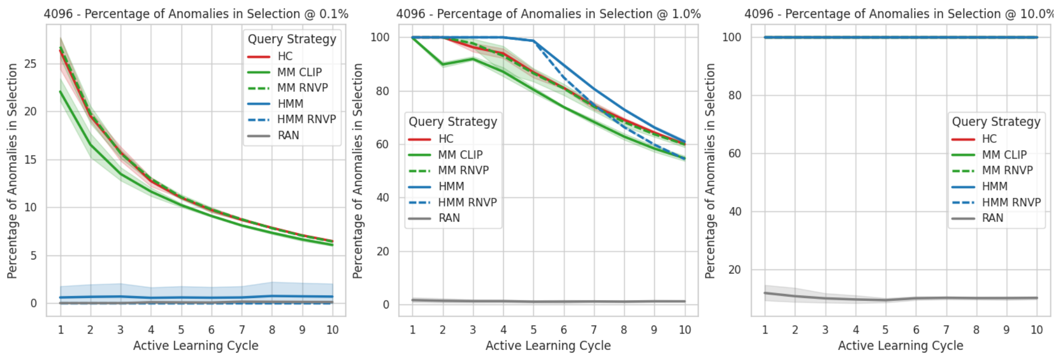


Figure 6.2. Progress curves comparing the total percentage of anomalies in the unlabeled set with 4096 initial samples.

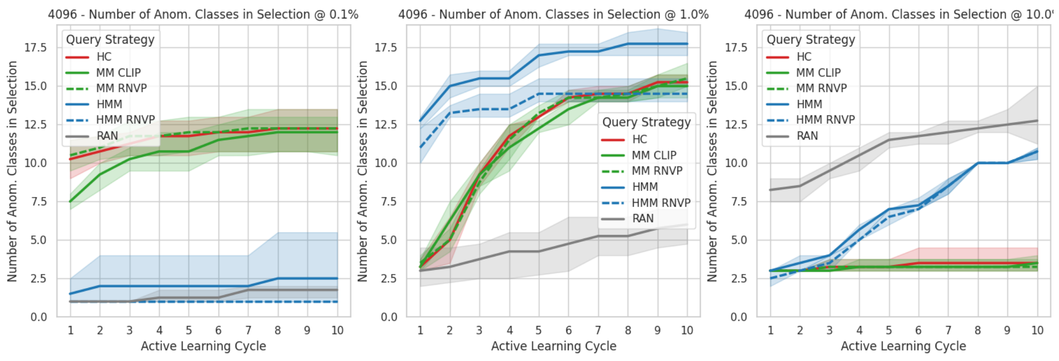


Figure 6.3. Progress curves comparing the variety of anomalies in the unlabeled set with 4096 initial samples.

larger anomalies selection produced the better models (see Figure 6.1). When the selection size is matched (see the central plot of Figure 6.2 and 6.3 and 6.1) the variety of anomalies is dominant; in the plot, the variety is expressed as number of unique anomaly classes selected).

It is safe to say that the resulting performance differences between query strategies, in the context of anomaly detection, do not only depend on the source of informativeness - as classic strategy classification implies - but also on the way a certain strategy selects anomalies, both in number and variety. We think that this later observation is novel to the active learning field and should be explored more in future research.

6.4.4 Drone in Train Tunnel Demonstrator

At M36 of the 1-SWARM project, we performed an offline demonstration of the detector lifecycle using both outlier exposure and active learning techniques to exploit the data collected by Hovering Solutions in a real outdoor tunnel. After the detector training only on samples without hazards, we run an offline cycle as follows:

1. Perform a mission, running the anomaly detector and collecting new samples and anomaly scores.

2. Use our active learning module (HMM RNVP) to select some samples to show to an expert operator and ask for them if they represent hazards.
3. Retrain the anomaly detector.
4. Back to step 1 for a new mission, comparing the classification performance with respect to the previous version during the new mission.

By adding a small amount of manually labeled data (750 frames), the AUC improved from 0.68 to 0.76. At M40, we performed the final demonstration in the target environment. We selected a variant of Real-NVP described in this Chapter that has proven the best for the task, which we deployed as a Python library onboard Hovering Solutions drones, similarly to what we achieved in the demonstrator of Sectio 4.4.2. Hovering Solutions integrated the detector into the control stack so that the drone retracts to the take-off position when the detector perceives an anomaly. The detector runs onboard of the drones at 1 fps due to camera limitation, but the detector could potentially run at 20 fps as one inference requires about 50 ms. Hovering Solution performed 4 runs, flying the drone for a total of about 200 meters in an outdoor tunnel. Note that they could perform fewer runs than planned due to strong wind. Like in the previous demonstration, the drone detected all hazards.

6.5 Conclusion

In this Chapter, we benchmarked 8 different active learning strategies for the task of visual anomaly detection in a robotic scenario. Two strategies are novel and are proposed as an alternative to the other ones. In our benchmark, we compare how the selected strategies improve the AUC and AP when tested on a large-scale dataset for anomaly detection. Our experiments show that at the beginning of a deployment cycle of an anomaly detection model, it is better to explore the data space, for example with a distance-based query strategy. Later when more data is collected and anomalies are found, it is better to move to a hybrid model like the one we propose, to improve the detection performance. We introduced a new setup for our detector, we used CLIP as a feature extractor for the RGB images in conjunction with Real-NVP for the detection over the feature embeddings; we acknowledge that this setup might not work for other robots' sensory configurations, thus, a limitation of this study is the input constraint that CLIP imposes. Nonetheless, with the work described, we can finally close the training-deployment-improvement cycle.

In the next Chapter, we will move our focus to the core vision technology and study what alternatives we have for detecting anomalies in 3D point clouds.

Chapter 7

Applications to 3D Point Clouds

In previous Chapters, we focused our research on 2D visual data as images and videos to detect hazards for robots. In this Chapter, motivated by the reduced costs and wider availability of high-resolution depth cameras and LIDaR sensors, we explore the use of 3D visual data, such as point clouds, as an alternative to 2D visual data for robotic environmental perception. We explore 3D anomaly detection using deep learning methods designed for point cloud data. We achieved this thanks to a recently released 3D dataset for Anomaly Detection of industrial products [7]. Using the MVTec-3D dataset, we contribute a comparison between a 3D point cloud features extractor, a 2D image features extractor, a combination of the two, and three baselines. We also compare our work with other models on the dataset's DETECTION-AUROC benchmark.

7.1 Background

Off-the-shelf pre-trained image feature extractors are increasingly being used in academic and industrial research for building deep-learning models to solve computer vision tasks. Recently, Vision Transformer [26](ViT) paved a new road for researchers to build even more complex and performing models for computer-vision tasks. A notable example of ViT-based models is CLIP from OpenAI [73] a very large and complex computer vision model trained on an enormous corpus of captioned images to solve any kind of vision task. Due to their size and reliance on large and complex datasets, models such as CLIP can be only developed and trained by a limited set of companies and research labs. However, most of these large models share an open-source nature with pre-trained models available online [1]. By removing the need to train an ad-hoc feature extractor, researchers can focus on solving the task at hand using the extracted feature embeddings; regularly smaller than images that contain low-level semantic information, the embeddings encode visual data into high-level semantic features allowing researchers to train computer vision models with fewer samples or smaller models (excluding the extractor).

Similarly, we see wider use, in both academia and industry, of 3D data through depth cameras, LIDARs, photogrammetry representation, or Neural Radiance fields for solving computer vision tasks. Nonetheless, one 3D task that is still understudied [30] is Anomaly Detection on 3D Data. The task of 3D Anomaly Detection has potential applications in many fields such as health care, industrial product inspection, industrial asset maintenance, site surveillance, and

[1] https://github.com/openai/CLIP

robotics; currently, all of these fields only rely on 2D Anomaly Detection.

A few recent works [38, 74, 96, 20, 64, 29] approached the task of Anomaly Detection on 3D data, with most [38, 74, 96, 20] focusing on Segmentation of Anomalies.

In this work, we ask ourselves if, with the current state of the art in deep learning models for 3D Point Clouds, it is possible to solve the task of anomaly detection on 3D point clouds without training an ad-hoc features extractor, similarly to what we achieve in our previous work from chapter 6. The use of pre-trained 3D feature extractors would remove the need for a difficult-to-develop and train 3d features extractor, lowering the entrance barrier to 3D visual data analysis.

7.2 Related Work

7.2.1 Dataset

For this work, we use the MVTec-3D dataset [7]. To the best of our knowledge, this is the only existing open-access dataset for the task of 3D Anomaly Detection, and more specifically, 3D Anomaly Segmentation.

The MVTec-3D dataset is built for studying the task of 3D anomaly segmentation in the context of industrial mass production; the dataset is composed of more than 4000 high-resolution point clouds and RGB images of 10 different objects with 10 different anomalies, captured using an industrial 3D sensor. The dataset is already subdivided into training, validation, and testing sets; all sets contain normal samples, classified into different object categories, but only the testing set contains anomalous samples.

For each anomalous test sample, a precisely annotated ground truth is provided; in Figure 7.1 a selection of the dataset is shown.

7.2.2 Models and Approaches

Image Anomaly Detection

In the previous Chapters, we discussed at length how anomaly detection can be achieved using images coming from robot cameras.

3D Anomaly Detection

While Image Anomaly Detection has been studied in different settings, 3D Anomaly Detection, due to the limitation to only MVTec 3D as the only representative dataset, is focused only on the topic of anomaly detection of industrial products.

One of the earlier studies on 3D anomaly detection, after the release of MVTec-3D, is from Horwitz et al [38]. In their work, the authors study the task of 3D anomaly detection and segmentation (3DAD&S) and compare some non-purposely made models with their proposed approach on the MVTec dataset. Their objective is to better understand if, for this task, the 3D data is useful or not; from their results, it's clear that while 2D approaches still beat the 3D purposely built ones, at least for the latter the 3D data is essential. Then they provide an analysis of the key properties for successful 3DAD&S representation, leading to their proposed approach called BTF; while their model achieves very good segmentation performances, the

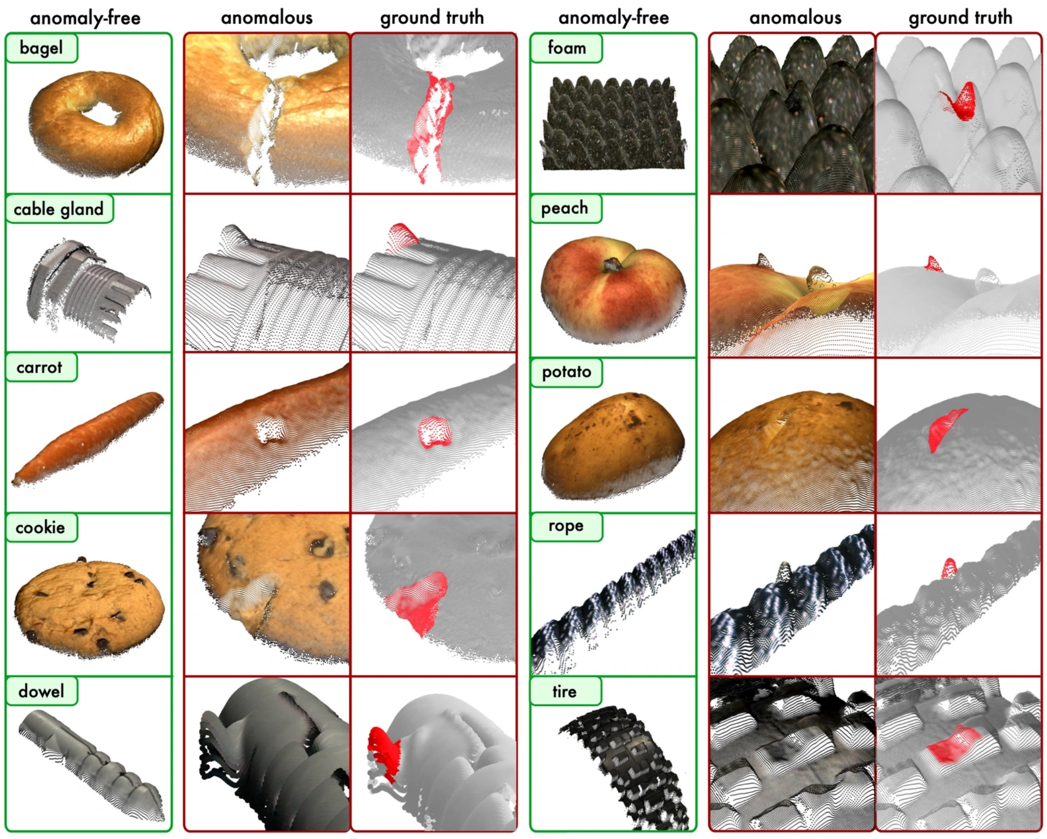


Figure 7.1. Examples of samples from the Mvtec3D dataset

authors recognize the model limitation on the image level accuracy, the same task we set to analyze in this work.

Rudolph et al. [74] propose an Asymmetric Student Teacher network to solve the Anomaly segmentation task on both the MVTec-3D and the original image-only MVTec dataset [8]. Different from other student-teacher networks, their approach uses normalizing flow models for the teacher and conventional feed-forward convolution blocks for the students creating a discrepancy in the student prediction outside of the normal data on which the student network is trained on.

In their paper, Wang et al [96] use the MVTec-3D dataset to propose a multimodal approach to 3D anomaly detection called Multi-3D-Memory (M3DM). With M3DM the authors combine features extracted from both 3D point clouds and images; first, patches of point clouds are produced using the farthest point sampling, then the points in each patch are encoded using Point Transformer [103] and the resulting features are remapped onto a 2D plane with the same size of the RGB picture and are then averaged into patch-wise features; at the same time patch features are extracted from the RGB image. Given the sets of patch features for both RGB and point cloud, the authors propose two new learnable modules called Unsupervised Feature Fusion and Decision Layer Fusion that are used, respectively, to learn the interaction between multimodal features and to deal with possible information loss that happens during the information fusion; the latter module uses multimodal memory banks during inference to produce the final anomaly and segmentation predictions.

Chu et al. [20], differently from others, propose a shape-guided approach for integrating the information from both RGB and point clouds. Their approach uses neural implicit functions to represent local areas of the point clouds. Similarly to others, they first split the point cloud in 3D patches, then these patches are passed to a PointNet network, and the resulting features are used by the Neural Implicit Function module, to extract components that are used to define signed distance functions that implicitly encode normal local representations; these are then combined with ResNet extracted RGB features and used to define segmentation maps of the anomalies.

3D Feature Extractor

Image Based A logical approach to 3D feature extraction is to use approaches well-tested on 2D data and adapt them to the additional dimension. In their paper [102], the authors propose to bridge the gap between a pre-trained CLIP [73] vision transformer and the point clouds from ModelNet [98] and ScanObjectNN [95] using point-projection images of different views of a single object. For each object, several views are generated and the resulting set of images is used to extract features. In their work, the authors note that using a zero-shot approach leads to poor performances, but with an additional trainable component after a few-shot training, the performance on the classification task increases.

Dong et al. [25] use pre-trained image vision transformers as part of the teacher encoder to, then, train a point cloud-only student encoder module. Their approach, called ACT, uses only *x,y,z* information and achieves the best performance on both point cloud classification and semantic segmentation.

Point Cloud Based In contrast to the aforementioned approaches, Zhang et al. [101] propose a point cloud-only approach that does not use images or image pre-trained models as part of the pipeline. With Point-M2AE, the authors define and train a multi-scale masked autoencoder with the objective of using it as a zero-shot point cloud encoder. The model is composed of an encoder and decoder with skip connections; the encoder is fed with differently scaled masked point clouds from the same sample. As for images, the masked autoencoder is trained using a proxy reconstruction task. In the original paper, the approach achieves promising results across different tasks; finally, the authors provide both code and pre-trained models.

In this work, we set to use a pre-trained version of the encoder, used in combination with an SVM to solve a Linear classification task. The major drawback for Point-M2AE is the limited input size; the point clouds used for training the encoder are limited to 1024 points while the MVTec-3D point clouds contain hundreds of thousands of points per sample. For this reason, part of the code provided by Zhang et al. has been adapted and a point sampler is introduced to reduce the MVTec point cloud to the correct size.

7.3 Experimental Setup

While the MVTec 3D dataset is built for 3D anomaly segmentation, in this work we will limit ourselves to the binary classification task of anomaly detection; for each point cloud our model will predict if it contains an anomaly or not. One of the metrics that is used in the dataset benchmark is the DETECTION-AUROC (in some cases called Image-AUROC); this metric indicates the AUC for detecting anomalies in the samples, without considering the segmentation. We will compare our results to those of the DETECTION-AUROC benchmark, available on the dataset page of *papers with code* [2].

In this work we use a two-part model, a feature extractor and an anomaly detection head; the latter receives as input the feature embedding from the extractor and produces an anomaly score for each embedding. The detection head is a Real-NVP model and - excluding adaptation to different embedding sizes - it is not changed throughout the experiments; thus the only changing part will be the feature extractor.

Real-NVP

This model has been already used in a recent paper [97], and by us in Chapters 5 and 6, as an anomaly detector based on latent embeddings. For this work we explore the Real-NVP hyper-parameters using an empirical process, ultimately landing on a similar setup to those used in the previous Chapters 5 and 6. The only differences between these experiments and previous works are the internal size of the layers and input size; these are input-dependent. Note that during training or hyper-parameter search, the Real-NVP always converged with the mapping, excluding its influence on the experiment results. All Real-NVPs are trained for 100 epochs with early stopping.

Baselines

We define two baselines, *Random* and *Ones*. The first substitutes the features extractor component with a random signal sampled from a Normal distribution; the latter produces a 1s feature vector as input for the Real-NVP. We use two different baselines to demonstrate that the Real-NVP component is not relevant to our experiment.

Handcrafted Baseline

We also define a set of handcrafted feature extractors to serve as an additional baseline. These basic features are heuristics chosen to be easy to compute and informative enough to detect macroscopic anomalies (e.g. a large piece of an object missing). The 11 features are the following:

- ft1: number of points in the point cloud
- ft2 to ft5: number of points in 4 quadrants (i.e. split the point cloud into 4 quadrants from a top view)
- ft6 and ft7: maximum and minimum z value for any point cloud's points
- ft8 and ft9: maximum and minimum x value for any point cloud's points
- ft10 and ft11: maximum and minimum y value for any point cloud's points

[2] https://paperswithcode.com/sota/rgb-3d-anomaly-detection-and-segmentation-on

XYZ Model

The first, non-baseline, approach proposed uses Point-M2AE as a feature extractor. This XYZ (i.e. point cloud data) encoder, uses positional information from the entire point cloud to produce a feature vector. The Point-M2AE encoder takes as input a 1024pts point cloud and produces 384 features; these are passed to the Real-NVP that maps them to a latent space where the "normality" probability can be extracted.

RGB model

Since the MVTec-3D dataset provides RGB images of the sample scans, we took the CLIP+Real-NVP model from our previous work, chapter 6, on image anomaly detection, and we used it to identify anomalies in the dataset. Notice that, while the RGB images are more informative for some specific anomalies (see the anomaly color for the object foam), the overall information provided to the Real-NVP is more limited than total information in a point cloud.

This setup uses the Vision Transformer(ViT) module of CLIP [73]; the ViT takes the RGB image as input and produces a 512-sized feature vector. As for the previous approaches, the vector is then passed to the Real-NVP component.

RGB+XYZ model Finally, we test an RGB+XYZ model by simply concatenating the 512 RGB-derived features and the 384 XYZ-derived features in a single vector for each sample.

Even in this case with an 896 size vector, the Real-NVP correctly converged and learned a mapping.

7.4 Experimental Results

We report all the AUC results of the 46 runs (excluding the hyper-parameter searches) in Tables 7.3,7.4,7.5,7.6 and 7.2. The results are color-coded, any value of AUC equal to or lower than 0.5 is colored red; higher values shift from red to yellow and towards green, which is the color for values near or equal to 1.

In each table, we report the performances split by anomaly type and object class, with the addition of the AUC considering the whole test set as a binary problem, and the averaged AUC, built by averaging the AUC of each object class per se.

As detailed by the tables, for all models except *random* and *ones*, we also consider the AUC for the models trained and tested on samples from a single object class; for example, all lines with *bagle* as object class, represent models that during training, validation, and testing, only saw bagels. To the best of our knowledge, we are the first to introduce this kind of experiment for this dataset; our motivation is to study the effects of each object class's characteristics (shape and color) on each specific model (and thus features extractor) performance.

The best-performing model is the RGB, CLIP-based one followed closely by the RGB+XYZ one. The RGB model, using all objects, achieved an AUC of 0.69 for the test set. We acknowledge that this performance is surpassed by other, more complex, models benchmarked on the MVTec-3D dataset, but are nonetheless better than the baselines and handcrafted. Moreover, both the RGB and RGB+XYZ beat the performance of the purposely built approaches proposed in [7], namely Voxel VM, Voxel AE, and Voxel GAN.

Table 7.1. Comparing our approaches to the MVTec-3D benchmark

Shape-Guided [20]	0.95
M3DM [96]	0.95
AST [74]	0.94
Back To Feat. [38]	0.87
RGB (Ours)	*0.69*
RGB+XYZ (Ours)	*0.67*
Voxel VM [7]	0.61
Voxel AE [7]	0.54
XYZ (Ours)	*0.53*
Voxel GAN [7]	0.52

Table 7.2. Baselines based Model AUC

Baseline	Obj seen	Bent	Color	Comb.	Contam.	Crack	Cut	Hole	Open	Thread	Test	Test Avg
random	**all**	**0.49**	**0.53**	**0.48**	**0.50**	**0.50**	**0.51**	**0.51**	**0.55**	**0.56**	**0.50**	**0.51**
ones	**all**	**0.50**	**0.50**	**0.50**	**0.50**	**0.50**	**0.50**	**0.50**	**0.50**	**0.50**	**0.50**	**0.50**

Table 7.3. XYZ Model AUC

	Bent	Color	Comb.	Contam.	Crack	Cut	Hole	Open	Thread	Test	Test Avg
all	**0.32**	**0.96**	**0.51**	**0.51**	**0.58**	**0.52**	**0.55**	**0.61**	**0.68**	**0.53**	**0.58**
bagel			0.71	0.67	0.71		0.68			0.69	0.69
cable gland	0.65					0.44	0.56		0.56	0.55	0.55
carrot			0.56	0.53	0.65	0.65	0.59			0.60	0.60
cookie			0.70	0.47	0.58		0.48			0.56	0.56
dowel	0.48		0.44	0.45		0.50				0.47	0.47
foam		0.48	0.65	0.49		0.54				0.54	0.54
peach			0.46	0.49		0.41	0.46			0.46	0.46
potato			0.43	0.47		0.33	0.40			0.41	0.41
rope				0.63		0.66		0.96		0.72	0.75
tire			0.30	0.49		0.50	0.46			0.48	0.44

Table 7.4. RGB Model AUC

	Bent	Color	Comb.	Contam.	Crack	Cut	Hole	Open	Thread	Test	Test Avg
all	**0.63**	**0.99**	**0.78**	**0.64**	**0.83**	**0.59**	**0.72**	**0.53**	**0.61**	**0.69**	**0.70**
bagel			0.92	0.63	1.00		0.79			0.84	0.83
cable gland	0.65					0.73	0.68		0.62	0.67	0.67
carrot			0.77	0.68	0.71	0.59	0.72			0.69	0.69
cookie			0.62	0.56	0.77		0.52			0.62	0.62
dowel	0.87		0.91	0.77		0.72				0.82	0.82
foam		1.00	0.82	0.70		0.75				0.82	0.82
peach			0.60	0.56		0.66	0.49			0.57	0.58
potato			0.60	0.59		0.43	0.41			0.51	0.51
rope				0.63		0.63		0.87		0.69	0.71
tire			0.48	0.54		0.52	0.59			0.55	0.53

Table 7.5. XYZ+RGB Model AUC

	Bent	Color	Comb.	Contam.	Crack	Cut	Hole	Open	Thread	Test	Test Avg
all	**0.53**	**1.00**	**0.75**	**0.63**	**0.81**	**0.60**	**0.66**	**0.62**	**0.61**	**0.67**	**0.69**
bagel			0.93	0.67	0.99		0.81			0.85	0.85
cable gland	0.74					0.69	0.73		0.69	0.71	0.72
carrot			0.76	0.72	0.70	0.62	0.73			0.70	0.70
cookie			0.67	0.58	0.84		0.58			0.67	0.67
dowel	0.83		0.89	0.76		0.63				0.78	0.78
foam		1.00	0.84	0.67		0.63				0.78	0.78
peach			0.57	0.59		0.60	0.49			0.56	0.56
potato			0.60	0.55		0.43	0.37			0.49	0.49
rope				0.62		0.67		0.87		0.70	0.72
tire			0.30	0.54		0.53	0.55			0.52	0.48

Table 7.6. Handcrafted based Model AUC

	Bent	Color	Comb.	Contam.	Crack	Cut	Hole	Open	Thread	Test	Test Avg
all	**0.35**	**0.78**	**0.53**	**0.51**	**0.58**	**0.48**	**0.48**	**0.30**	**0.13**	**0.49**	**0.46**
bagel			0.52	0.44	0.48		0.48			0.48	0.48
cable gland	0.47					0.49	0.64		0.43	0.51	0.51
carrot			0.51	0.42	0.46	0.49	0.35			0.45	0.45
cookie			0.61	0.58	0.67		0.73			0.65	0.65
dowel	0.59		0.52	0.64		0.51				0.57	0.56
foam		0.67	0.60	0.65		0.69				0.65	0.65
peach			0.52	0.60		0.48	0.48			0.52	0.52
potato			0.00	0.00		0.00	0.00			0.00	0.00
rope				0.60		0.59		0.69		0.62	0.62
tire			0.60	0.48		0.58	0.77			0.61	0.61

7.5 Discussion

In this Chapter, we compared different off-the-shelf pre-trained feature extractors combined with a Real-NVP model to solve the task of 3D anomaly detection on the MVTec-3D dataset.

From our experiments, it is clear that all approaches tested while better than the baselines are not sufficient for an anomaly detection task. We attribute the limited performances to the performance of the features extractors; while performing excellently on their original tasks, the models available are still too limited to solve this task; for example, the XYZ Point-M2AE extractor is strongly limited by the number of points it accepts in input and thus losing important local details that might help to detect small anomalies. In addition, we think that the dataset is too small; this implies that Real-NVP, while correctly converging, fails to learn a mapping of the normal samples to correctly identify anomalies.

With our results, we demonstrate that a combination of a dataset limitation and additional complexity when dealing with point clouds versus images leads to a lack of *off-the-shelf* models for solving complex 3D vision tasks.

We believe that models such as ours could be used in those situations for which not enough data is already available to train ad-hoc models. Later in deployment, when enough data is collected and labeled, more performing bespoke models can be trained and used for the application.

Chapter 8

Conclusions

In this thesis, we focused our contributions and research efforts on exploring solutions for the task of hazard detection for mobile robots using visual anomaly detection. Here we outline our contributions and delineate what we believe might be future steps to improve research in the field.

Using Visual Anomaly Detection for Finding Hazards for Mobile Robots

Our first two contributions, Chapters 3 and 4, focused on building a dataset for the training of deep learning-based hazard detection models for robots and on the design of the hazard detector using visual anomaly detection models on images coming from robots' front-facing cameras. The dataset is composed of more than 320‘000 frames collected in three scenarios by two robots, a drone and a wheeled robot, that traverse tunnels or corridors. In the dataset, normal samples are composed of images of a tidy empty, and traversable environment; anomalous samples instead contain different types of objects or situations that might be hazardous for the robot; examples of anomalies are simple boxes, foam on the floor, or roots from the ceiling. To solve the hazard detection task we used an undercomplete convolutional autoencoder trained on the task of reconstructing the input while subject to an information bottleneck at the embedding level. The autoencoder was trained using only normal samples as these are the only ones required by anomaly detection models. The experimental results showed that this method for detecting hazards works and can be used in real-time detection. We demonstrated this last statement deploying the resulting model in two separate demonstrators showing the realistic applicability of such an approach. The demonstrators showed the qualitative performance of real-time hazard detection for robots by correctly detecting all anomalies encountered by a drone and a wheeled robot.

Defining a Deployment Life Cycle for a Hazard Detection Model

With the work achieved in our previous works and motivated by the real-life performances of our method, in Chapters 5 and 6, we researched the additional components of a hazard detection system necessary to employ it in a real-life application. We modified our previous model with the addition of a Real-NVP model after the undercomplete autoencoder's encoder. This new component acted as the anomaly detector and allowed us to introduce the outlier exposure component to the Real-NVP loss. Using outlier exposure we introduced a technique for training also on anomalous samples eventually collected during robot's missions. Our experimental setup showed that our solution is effective at increasing the separation degree, in the Real-NVP's latent space, between normal and known anomalous samples, while improving the overall detection performance. With the new opportunity of effectively training on additional anomalous samples, we then proposed two new active learning methods compatible with deep learning based anomaly detection. To do so we changed our model architecture by switching out the to-be-trained autoencoder's encoder for a pre-trained feature extractor built using vision transformers called ViT-B/32 from the large multimodal model CLIP. With this setup, we proposed two active learning queries and compared them with six other ones. The results indicated that when some data is collected in the field, ours are best for selecting informative samples to be used for retraining the model. With these results, we deployed our model on a real drone that had to traverse train tunnels. Even in this qualitative demonstration, we detected all the anomalies that the drone faced.

Exploring 3D Vision as an Alternative to 2D Vision

Having achieved our original objective of detecting hazards for robots using 2D vision, in Chapter 7 we extended our research to 3D vision. We explored if a general-purpose 3D point cloud feature extraction model, such as Point-M2AE, can be used as the backbone for a 3D anomaly detector.While promising, our results showed that some limitations are still present in current backbones and need to be dealt with to achieve a performance comparable to what we can achieve for 2D anomaly detection.

8.1 Future work

Across our thesis work, we collected future research directions and applications, some explored in practice and others only thought of. Here is a list of the most interesting to us.

Multimodal Anomaly Detection Given our work on image-based anomaly detection and the large availability of multimodal vision sensors, such as Azure Kinect camera or thermal and vision camera or even the presence of simple microphones, we think that one promising direction to be investigated for the task of hazard detection is the use of multimodal sensory information for improving the detection of hazards. We recognize that multimodal vision has been already explored for other vision tasks such as navigation and obstacle detection, but the nuances of anomaly detection might render working solutions from other fields unusable in our context.

Domain adaptation One key aspect of environmental understanding in the context of learning the scenario for developing the hazard detection system is its variability. In our experiments, we implicitly assumed that across experiments and demos, the environments would be mostly the same; real-world applications in robotics often cannot assume how the environment will look like, potentially limiting a model application only to similarly looking scenarios to the training one. We propose as a future research direction the focus on domain adaptation solutions for anomaly detection models exploring solutions to adapt the normality concept to change and adapt to different environments.

New Datasets and New General 3D Vision Models Our research on 3D visual anomaly detection showed that to achieve good performances with this task ad-hoc models have to be put in place; nonetheless models such as ours can be used as a placeholder before enough data is collected. Achieving the performance of foundation models such as CLIP in the image field, will require novel datasets that span multiple contexts, point cloud resolutions, and sizes to be used to train the backbone model; we recognize that capturing environment vs capturing an object with a point cloud are two different processes with different results but recent Gaussian Splat representation are indicating a possible research direction. With such datasets and approaches, we think that it would be easier to train proper models that generalize across different contexts and scopes.

Developing a Commercial Solution for The Market From the inception, my doctoral studies have been focused on industrial applications of robotics and deep learning-based computer vision; with the results achieved, during the last months of this Ph.D., we started to follow the steps required to found a startup. In August 2023, we were accepted to the Boldbrain challenge [1], and in December 2023 we were awarded the second prize. Even more recently we got accepted at the prestigious Swiss startup competition *»venture»* [2]. Now we are focusing on bringing anomaly detection to the market as a software solution compatible with robotic inspection of industrial sites.

[1] boldbrain.ch
[2] venture.ch

Appendix A

Appendix

List of Publications in Chronological Order

[57] Dario Mantegazza, Jérôme Guzzi, Luca Maria Gambardella, and Alessandro Giusti. Video: Learning Vision-Based Quadrotor Control in User Proximity. *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 369–369, 2019a.

[58] Dario Mantegazza, Jérôme Guzzi, Luca Maria Gambardella, and Alessandro Giusti. Vision-based Control of a Quadrotor in User Proximity: Mediated vs End-to-End Learning Approaches. *2019 International Conference on Robotics and Automation (ICRA)*, 6489–6495, 2019b.

[12] Gülcan Can, Dario Mantegazza, Gabriele Abbate, Sébastien Chappuis, and Alessandro Giusti. Semantic segmentation on Swiss3DCities: A benchmark study on aerial photogrammetric 3D point cloud dataset. *Pattern Recognition Letters*, 150: 108–114, 2021.

[1] Gabriele Abbate, Dario Mantegazza, Gülcan Can and Alessandro Giusti. 3D Semantic Segmentation of Urban Point Clouds: a Study of Generalization across Datasets and Cities. *Technical Report to appear on ArXiv*, 2021.

[14] Elia Cereda, Marco Ferri, Dario Mantegazza, Nicky Zimmerman, Luca M. Gambardella, Jérôme Guzzi, Alessandro Giusti, and Daniele Palossi. Improving the Generalization Capability of DNNs for Ultra-low Power Autonomous Nano-UAVs. *2021 17th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 327–334, 2021.

[28] Marco Ferri, Dario Mantegazza, Elia Cereda, Nicky Zimmerman, Luca M. Gambardella, Daniele Palossi, Jérôme Guzzi, and Alessandro Giusti Training Lightweight CNNs for Human-Nano drone Proximity Interaction from Small Datasets using Background Randomization. *Workshop paper published on ArXiv*, 2021.

[59] Dario Mantegazza, Alessandro Giusti, Luca M. Gambardella, Andrea Rizzoli, Jérôme Guzzi. Challenges in Visual Anomaly Detection for Mobile Robots. *Workshop paper presented at the ICRA 2022 Workshop on Safe and Reliable Robot Autonomy under Uncertainty, distributed on ArXiv*

[61] Dario Mantegazza, Carlos Redondo, Fran Espada, Luca M. Gambardella, Alessandro Giusti, and Jérôme Guzzi. Sensing Anomalies as Potential Hazards: Datasets and Benchmarks. *Towards Autonomous Robotic Systems. TAROS 2022. Lecture Notes in Computer Science*, 13546: 205–219, 2022b.

[60] Dario Mantegazza, Alessandro Giusti, Luca Maria Gambardella, and Jérôme Guzzi. An Outlier Exposure Approach to Improve Visual Anomaly Detection Performance for Mobile

Robots. *IEEE Robotics and Automation Letters*, 7 (4): 11354–11361, 2022a.
[63] Dario Mantegazza, Alind Xhyra, Alessandro Giusti, and Jérôme Guzzi. Active Anomaly Detection for Autonomous Robots: A Benchmark *Towards Autonomous Robotic Systems. TAROS 2022. Lecture Notes in Computer Science*, 315-327 , 2023.
[56] Dario Mantegazza, and Alessandro Giusti. Detecting Anomalous 3D Point Clouds Using Pre-trained Feature Extractors. *VISAPP*, to appear in proceedings, 2024.

Notes on Other Publications We published a conference paper [58] and a video paper [57] related to our research on Human pose estimation from frontal drone's camera images using reactive models based on Convolutional Neural Networks. While this work is the baseline for the following research done by others at IDSIA on drone perception [14, 28, 68, 104, 69], [57, 58] are only marginally related to the content of this Ph.D. thesis.

Technical Reports

- Innosuisse Nomoko - *Research Plan*, Tech. Rep. 2019.
- 1-SWARM - *D4.1: Functional architecture of the Swarm Intelligence Algorithmic Framework*, Tech. Rep. 2021.
- 1-SWARM - *D4.2: Algorithms for seamless CPSoS navigation and planning in human environments*, Tech. Rep. 2022.

Datasets

Swiss3DCities In [12] we released a semantically labeled dataset of 3D point clouds from five different cities. The dataset is available at https://zenodo.org/record/4390295.

Hazards&Robots In [60, 61] we released an image-based dataset acquired in robot exploration scenarios for the task of Anomaly Detection. The dataset is available at https://github.com/idsia-robotics/hazard-detection.

Software Releases

API for Semantic Segmentation of Point clouds For the Innosuisse project (grant 51NF40_185543) we released documented API to train, test, and deploy semantic segmentation models for analyzing 3D point clouds. This tool worked in combination with a custom database (built by SUPSI) that stored the point clouds collected by Nomoko. In addition, our API provides a simple web-based GUI to explore the segmentation results The API, written in Python and based on PyTorch, is a crucial part of the Nomoko segmentation pipeline.

API for Hazard detection for autonomous drones For the 1-SWARM project, in Chapter 4 we released over various release versions, the API to train, test, and deploy an Anomaly Detection model. This model is deployed, onboard an autonomous drone, by our partner Hovering Solutions to detect environmental hazards for the drone itself. The API, in addition to the model and relative documentation, provides three different GUIs for visualizing the model prediction. This API is built and optimized to run onboard a Jetson TX2 using Python, PyTorch, and Docker.

Paper code release As part of our publications process we release, when possible, the relative code. Our papers [58, 57, 1, 60, 61] have relative repositories that provide documented source code.

Bibliography

[1] Gabriele Abbate, Dario Mantegazza, Gülcan Can, and Alessandro Giusti. 3d semantic segmentation of urban point clouds: a study of generalization across datasets and cities. *Technical Report*, 2021. To appear on ArXiv.

[2] Abdelrahman Abdelhamed, Marcus A Brubaker, and Michael S Brown. Noise Flow: Noise Modeling with Conditional Normalizing Flows. In *International Conference on Computer Vision (ICCV)*, 2019.

[3] Dana Angluin. Queries and Concept Learning. *Machine Learning*, 2(4):319–342, April 1988. ISSN 1573-0565. doi: 10.1023/A:1022821128753. URL https://doi.org/10.1023/A:1022821128753.

[4] Vincent BarnabÃ©-Lortie, Colin Bellinger, and Nathalie Japkowicz. Active learning for one-class classification. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 390–395, 2015. doi: 10.1109/ICMLA.2015.167.

[5] Eric B Baum and Kenneth Lang. Query learning can work poorly when a human oracle is used. In *International joint conference on neural networks*, volume 8, page 8. Beijing China, 1992.

[6] William H. Beluch, Tim Genewein, Andreas NÃ¼rnberger, and Jan M. KÃ¶hler. The Power of Ensembles for Active Learning in Image Classification. pages 9368–9377, 2018. URL https://openaccess.thecvf.com/content_cvpr_2018/html/Beluch_The_Power_of_CVPR_2018_paper.html.

[7] Paul Bergmann., Xin Jin., David Sattlegger., and Carsten Steger. The mvtec 3d-ad dataset for unsupervised 3d anomaly detection and localization. In *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5: VISAPP,*, pages 202–213. INSTICC, SciTePress, 2022. ISBN 978-989-758-555-5. doi: 10.5220/0010865000003124.

[8] Paul Bergmann et al. The mvtec anomaly detection dataset: a comprehensive real-world dataset for unsupervised anomaly detection. *Int. Journal of Computer Vision*, 129:1038–1059, 2021.

[9] Zachary Birnbaum et al. Unmanned aerial vehicle security using behavioral profiling. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1310–1319, 2015. doi: 10.1109/ICUAS.2015.7152425.

[10] Hermann Blum et al. The fishyscapes benchmark: Measuring blind spots in semantic segmentation. *Int. Journal of Computer Vision*, 129(11):3119–3135, 2021.

[11] Samuel Budd, Emma C. Robinson, and Bernhard Kainz. A survey on active learning and human-in-the-loop deep learning for medical image analysis. *Medical Image Analysis*, 71: 102062, 2021. ISSN 1361-8415. doi: https://doi.org/10.1016/j.media.2021.102062.

[12] Gülcan Can, Dario Mantegazza, Gabriele Abbate, Sébastien Chappuis, and Alessandro Giusti. Semantic segmentation on swiss3dcities: A benchmark study on aerial photogrammetric 3d pointcloud dataset. *Pattern Recognition Letters*, 150:108–114, 2021.

[13] Andrea Castellani, Sebastian Schmitt, and Stefano Squartini. Real-World Anomaly Detection by Using Digital Twin Systems and Weakly Supervised Learning. *IEEE Transactions on Industrial Informatics*, 17(7):4733–4742, July 2021. ISSN 1551-3203, 1941-0050. doi: 10.1109/TII.2020.3019788. URL https://ieeexplore.ieee.org/document/9179030/.

[14] Elia Cereda, Marco Ferri, Dario Mantegazza, Nicky Zimmerman, Luca M. Gambardella, JÃ©rÃ´me Guzzi, Alessandro Giusti, and Daniele Palossi. Improving the generalization capability of dnns for ultra-low power autonomous nano-uavs. In *2021 17th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 327–334, 2021. doi: 10.1109/DCOSS52077.2021.00060.

[15] Punarjay Chakravarty, Alan Miao Zhang, Raymond Austin Jarvis, and Lindsay Kleeman. Anomaly detection and tracking for a patrolling robot. In *Proc. of the Australiasian Conference on Robotics and Automation 2007*, pages 1 – 9, 2007. ISBN 978-0-9587583-9-0.

[16] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3), jul 2009. ISSN 0360-0300. doi: 10.1145/1541880.1541882.

[17] Scott Chen and Ramesh Gopinath. Gaussianization. In *Advances in Neural Information Processing Systems (NIPS)*, volume 13, 2000.

[18] Kyunghyun Cho et al. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, October 2014. doi: 10.3115/v1/D14-1179.

[19] Peter Christiansen et al. Deepanomaly: Combining background subtraction and deep learning for detecting obstacles and anomalies in an agricultural field. *Sensors*, 16(11): 1904, Nov 2016. ISSN 1424-8220. doi: 10.3390/s16111904.

[20] Yu-Min Chu, Chieh Liu, Ting-I Hsieh, Hwann-Tzong Chen, and Tyng-Luh Liu. Shape-guided dual-memory learning for 3D anomaly detection. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 6185–6194. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/chu23b.html.

[21] David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, May 1994. ISSN 1573-0565. doi: 10.1007/BF00993277. URL https://doi.org/10.1007/BF00993277.

[22] Marius Cordts et al. The cityscapes dataset for semantic urban scene understanding. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3213–3223, 2016.

[23] Jia Deng et al. Imagenet: A large-scale hierarchical image database. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009.

[24] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp, 2016.

[25] Runpei Dong, Zekun Qi, Linfeng Zhang, Junbo Zhang, Jianjian Sun, Zheng Ge, Li Yi, and Kaisheng Ma. Autoencoders as cross-modal teachers: Can pretrained 2d image transformers help 3d representation learning? In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=8Oun8ZUVe8N.

[26] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, June 2021. URL http://arxiv.org/abs/2010.11929. arXiv:2010.11929 [cs].

[27] Sarah M. Erfani and Sothers. High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning. *Pattern Recognition*, 58:121–134, 2016. ISSN 0031-3203. doi: 10.1016/j.patcog.2016.03.028.

[28] Marco Ferri, Dario Mantegazza, Elia Cereda, Nicky Zimmerman, Luca M Gambardella, Daniele Palossi, Jérôme Guzzi, and Alessandro Giusti. Training lightweight cnns for human-nanodrone proximity interaction from small datasets using background randomization. *arXiv preprint arXiv:2110.14491*, 2021.

[29] Alberto Floris, Luca Frittoli, Diego Carrera, and Giacomo Boracchi. Composite layers for deep anomaly detection on 3d point clouds, 2022. URL https://arxiv.org/abs/2209.11796.

[30] Luca Frittoli. *ADVANCED LEARNING METHODS FOR ANOMALY DETECTION IN MULTIVARIATE DATASTREAMS AND POINT CLOUDS*. PhD thesis, Politecnico Milano, 2022.

[31] Jie Geng et al. High-resolution sar image classification via deep convolutional autoencoders. *IEEE Geoscience and Remote Sensing Letters*, 12(11):2351–2355, 2015.

[32] Ian J. Goodfellow et al. Generative adversarial networks, 2014.

[33] Xifeng Guo, Xinwang Liu, En Zhu, and Jianping Yin. Deep clustering with convolutional autoencoders. In *International conference on neural information processing*, pages 373–382. Springer, 2017.

[34] Matthias Haselmann, Dieter P. Gruber, and Paul Tabatabai. Anomaly detection using deep learning based image completion. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1237–1242, 2018. doi: 10.1109/ICMLA.2018.00201.

[35] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv*, 2016.

[36] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. *arXiv*, 2018.

[37] Jonathan Ho et al. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *International Conference on Machine Learning*, pages 2722–2730. PMLR, 2019.

[38] Eliahu Horwitz and Yedid Hoshen. Back to the Feature: Classical 3D Features Are (Almost) All You Need for 3D Anomaly Detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2967–2976, 2023. URL https://openaccess.thecvf.com/content/CVPR2023W/VAND/html/Horwitz_Back_to_the_Feature_Classical_3D_Features_Are_Almost_All_CVPRW_2023_paper.html.

[39] Chin-Wei Huang, David Krueger, Alexandre Lacoste, and Aaron Courville. Neural Autoregressive Flows. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2078–2087. PMLR, July 2018. URL https://proceedings.mlr.press/v80/huang18d.html.

[40] Marco Hutter et al. Anymal - a highly mobile and dynamic quadrupedal robot. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 38–44, 2016. doi: 10.1109/IROS.2016.7758092.

[41] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

[42] Hannah R Kerner et al. Novelty detection for multispectral images with application to planetary exploration. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33 (01):9484–9491, Jul. 2019. doi: 10.1609/aaai.v33i01.33019484.

[43] Eliahu Khalastchi, Meir Kalech, Gal A Kaminka, and Raz Lin. Online data-driven anomaly detection in autonomous robots. *Knowledge and Information Systems*, 43(3):657–688, 2015. doi: 10.1007/s10115-014-0754-y.

[44] Eliahu Khalastchi et al. Online anomaly detection in unmanned vehicles. In *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '11, page 115-122, Richland, SC, 2011. ISBN 0982657153.

[45] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[46] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013.

[47] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018.

[48] Ivan Kobyzev, Simon J.D. Prince, and Marcus A. Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11):3964–3979, 2021. doi: 10.1109/TPAMI.2020.2992934.

[49] M.A. Kramer. Autoassociative neural networks. *Computers & Chemical Engineering*, 16 (4):313–328, 1992. ISSN 0098-1354. doi: 10.1016/0098-1354(92)80051-A. Neutral network applications in chemical engineering.

[50] A Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, University of Toronto, 2009.

[51] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25: 1097–1105, 2012.

[52] Punit Kumar and Atul Gupta. Active Learning Query Strategies for Classification, Regression, and Clustering: A Survey. *Journal of Computer Science and Technology*, 35 (4):913–945, July 2020. ISSN 1860-4749. doi: 10.1007/s11390-020-9487-4. URL https://doi.org/10.1007/s11390-020-9487-4.

[53] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

[54] Yann LeCun et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[55] David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '94, pages 3–12, Berlin, Heidelberg, August 1994. Springer-Verlag. ISBN 978-0-387-19889-7.

[56] Dario Mantegazza and Alessandro Giusti. Detecting Anomalous 3D Point Clouds Using Pre-trained Feature Extractors. Rome, IT, 2024. To appear.

[57] Dario Mantegazza, JÃ©rÃ´me Guzzi, Luca M Gambardella, and Alessandro Giusti. Video: Learning vision-based quadrotor control in user proximity. In *HRI '19: 2019 ACM/IEEE International Conference on Human-Robot Interaction, March 11-14, 2019, Daegu, Rep. of Korea*, March 2019. doi: 10.1109/HRI.2019.8673022.

[58] Dario Mantegazza, JÃ©rÃ´me Guzzi, Luca Maria Gambardella, and Alessandro Giusti. Vision-based control of a quadrotor in user proximity: Mediated vs end-to-end learning approaches. *2019 IEEE International Conference on Robotics and Automation (ICRA)*, 2019. doi: 10.1109/ICRA.2019.8794377. URL https://github.com/idsia-robotics/proximity-quadrotor-learning.

[59] Dario Mantegazza, Alessandro Giusti, Luca M. Gambardella, Andrea Rizzoli, and JÃ©rÃ´me Guzzi. Challenges in visual anomaly detection for mobile robots, 2022. URL https://arxiv.org/abs/2209.10995. Workshop paper presented at the ICRA 2022 Workshop on Safe and Reliable Robot Autonomy under Uncertainty.

[60] Dario Mantegazza, Alessandro Giusti, Luca Maria Gambardella, and JÃ©rÃ´me Guzzi. An outlier exposure approach to improve visual anomaly detection performance for mobile robots. *IEEE Robotics and Automation Letters*, pages 1–8, 2022. doi: 10.1109/LRA.2022.3192794.

[61] Dario Mantegazza, Carlos Redondo, Fran Espada, Luca M. Gambardella, Alessandro Giusti, and Jérôme Guzzi. Sensing anomalies asÂ potential hazards: Datasets andÂ benchmarks. In Salvador Pacheco-Gutierrez, Alice Cryer, Ipek Caliskanelli, Harun Tugal, and Robert Skilton, editors, *Towards Autonomous Robotic Systems*, pages 205–219.

Springer International Publishing, 2022. ISBN 978-3-031-15908-4. doi: 10.1007/978-3-031-15908-4_17.

[62] Dario Mantegazza, Alind Xhyra, Luca M. Gambardella, Alessandro Giusti, and Jerome Guzzi. Hazards&Robots: A Dataset for Visual Anomaly Detection in Robotics, April 2023. URL https://zenodo.org/record/7859211.

[63] Dario Mantegazza, Alind Xhyra, Alessandro Giusti, and JÃ©rÃ´me Guzzi. Active Anomaly Detection for Autonomous Robots: A Benchmark. In Fumiya Iida, Perla Maiolino, Arsen Abdulali, and Mingfeng Wang, editors, *Towards Autonomous Robotic Systems*, Lecture Notes in Computer Science, pages 315–327. Springer Nature Switzerland, 2023. ISBN 978-3-031-43360-3. doi: 10.1007/978-3-031-43360-3_26.

[64] Mana Masuda, Ryo Hachiuma, Ryo Fujii, Hideo Saito, and Yusuke Sekikawa. Toward unsupervised 3d point cloud anomaly detection using variational autoencoder. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 3118–3122, 2021. doi: 10.1109/ICIP42928.2021.9506795.

[65] Matthew D. Mitchell et al. Living on the edge: how does environmental risk affect the behavioural and cognitive ecology of prey? *Animal Behaviour*, 115:185–192, 2016. ISSN 0003-3472. doi: 10.1016/j.anbehav.2016.03.018.

[66] Lucia Moretti, Marleen Hentrup, Kurt Kotrschal, and Friederike Range. The influence of relationships on neophobia and exploration in wolves and dogs. *Animal Behaviour*, 107: 159–173, 2015. ISSN 0003-3472. doi: 10.1016/j.anbehav.2015.06.008.

[67] Yuval Netzer et al. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[68] Daniele Palossi. *On the Autonomous Navigation of Nano-UAVs*. PhD thesis, ETH Zurich, 2019.

[69] Daniele Palossi, Nicky Zimmerman, Alessio Burrello, Francesco Conti, Hanna Müller, Luca Maria Gambardella, Luca Benini, Alessandro Giusti, and Jérôme Guzzi. Fully on-board ai-powered human-drone pose estimation on ultralow-power autonomous flying nano-uavs. *IEEE Internet of Things Journal*, 9(3):1913–1929, 2021.

[70] Daehyung Park et al. Multimodal execution monitoring for anomaly detection during robot manipulation. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 407–414, 2016. doi: 10.1109/ICRA.2016.7487160.

[71] Daehyung Park et al. A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder. *IEEE Robotics and Automation Letters*, 3(3):1544–1551, 2018. doi: 10.1109/LRA.2018.2801475.

[72] Ken Perlin. An image synthesizer. *ACM Siggraph Computer Graphics*, 19(3):287–296, 1985.

[73] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision, February 2021. URL http://arxiv.org/abs/2103.00020. arXiv:2103.00020 [cs].

[74] Marco Rudolph, Tom Wehrbein, Bodo Rosenhahn, and Bastian Wandt. Asymmetric Student-Teacher Networks for Industrial Anomaly Detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2592–2602, 2023. URL https://openaccess.thecvf.com/content/WACV2023/html/Rudolph_Asymmetric_Student-Teacher_Networks_for_Industrial_Anomaly_Detection_WACV_2023_paper.html.

[75] Lukas Ruff et al. Deep one-class classification. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4393–4402, 10–15 Jul 2018.

[76] Lukas Ruff et al. A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*, 109(5):756–795, 2021. doi: 10.1109/JPROC.2021.3052449.

[77] Mohammad Sabokrou, Mohammad Khalooei, Mahmood Fathy, and Ehsan Adeli. Adversarially learned one-class classifier for novelty detection. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3379–3388, 2018. doi: 10.1109/CVPR.2018.00356.

[78] Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, MLSDA'14, page 4-11, New York, NY, USA, 2014. ISBN 9781450331593. doi: 10.1145/2689746.2689747.

[79] Natasa Sarafijanovic-Djukic and Jesse Davis. Fast distance-based anomaly detection in images using an inception-like autoencoder. In Petra Kralj Novak, Tomislav Šmuc, and Sašo Džeroski, editors, *Discovery Science*, pages 493–508, Cham, 2019. ISBN 978-3-030-33778-0. doi: 10.1007/978-3-030-33778-0\textunderscore37.

[80] Thomas Schlegl et al. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *Information Processing in Medical Imaging*, pages 146–157, Cham, 2017. ISBN 978-3-319-59050-9. doi: 10.1007/978-3-319-59050-9\textunderscore12.

[81] Thomas Schlegl et al. f-anogan: Fast unsupervised anomaly detection with generative adversarial networks. *Medical image analysis*, 54:30–44, 2019.

[82] Luke Scime and Jack Beuth. A multi-scale convolutional neural network for autonomous anomaly detection and classification in a laser powder bed fusion additive manufacturing process. *Additive Manufacturing*, 24:273–286, 2018. ISSN 2214-8604. doi: 10.1016/j.addma.2018.09.034.

[83] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=H1aIuk-RW.

[84] Burr Settles. Active learning literature survey. 2009.

[85] David Sloan Wilson, Anne B. Clark, Kristine Coleman, and Ted Dearstyne. Shyness and boldness in humans and other animals. *Trends in Ecology & Evolution*, 9(11):442–446, 1994. ISSN 0169-5347. doi: 10.1016/0169-5347(94)90134-1.

[86] Asim Smailagic, Pedro Costa, Hae Young Noh, Devesh Walawalkar, Kartik Khandelwal, Adrian Galdran, Mostafa Mirshekari, Jonathon Fagert, Susu Xu, Pei Zhang, and AurÃ©lio Campilho. Medal: Accurate and robust deep active learning for medical image analysis. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 481–488, 2018. doi: 10.1109/ICMLA.2018.00078.

[87] Mareike StÃ¶we, Thomas Bugnyar, Bernd Heinrich, and Kurt Kotrschal. Effects of group size on approach to novel objects in ravens (corvus corax). *Ethology*, 112(11):1079–1088, 2006. doi: 10.1111/j.1439-0310.2006.01273.x.

[88] Christian Szegedy et al. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015. doi: 10.1109/CVPR.2015.7298594.

[89] Xuning Tang, Yihua Shi Astle, and Craig Freeman. Deep Anomaly Detection with Ensemble-Based Active Learning. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 1663–1670, December 2020. doi: 10.1109/BigData50022.2020.9378315.

[90] Ole Tange et al. Gnu parallel-the command-line power tool. *The USENIX Magazine*, 36 (1):42–47, 2011.

[91] Antonio Torralba, Rob Fergus, and William T Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):1958–1970, 2008.

[92] Holger Trittenbach and Klemens BÃ¶hm. One-Class Active Learning for Outlier Detection with Multiple Subspaces. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, pages 811–820, New York, NY, USA, November 2019. Association for Computing Machinery. ISBN 978-1-4503-6976-3. doi: 10.1145/3357384.3357873. URL https://doi.org/10.1145/3357384.3357873.

[93] Holger Trittenbach, Adrian Englhardt, and Klemens BÃ¶hm. An overview and a benchmark of active learning for outlier detection with one-class classifiers. *Expert Systems with Applications*, 168:114372, April 2021. ISSN 0957-4174. doi: 10.1016/j.eswa.2020.114372. URL https://www.sciencedirect.com/science/article/pii/S0957417420310496.

[94] Nina Tuluptceva, Bart Bakker, Irina Fedulova, and Anton Konushin. Perceptual image anomaly detection. In *Asian Conference on Pattern Recognition*, pages 164–178. Springer, 2019.

[95] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[96] Yue Wang, Jinlong Peng, Jiangning Zhang, Ran Yi, Yabiao Wang, and Chengjie Wang. Multimodal Industrial Anomaly Detection via Hybrid Fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8032–8041, June 2023.

[97] Lorenz Wellhausen, RenÃ© Ranftl, and Marco Hutter. Safe robot navigation via multi-modal anomaly detection. *IEEE Robotics and Automation Letters*, 5(2):1326–1333, 2020. doi: 10.1109/LRA.2020.2967706.

[98] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[99] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv*, 2016.

[100] Jiaxin Zhang, Kyle Saleeby, Thomas Feldhausen, Sirui Bi, Alex Plotkowski, and David Womble. Self-Supervised Anomaly Detection via Neural Autoregressive Flows with Active Learning. October 2021. URL https://openreview.net/forum?id=LdWEo5mri6.

[101] Renrui Zhang, Ziyu Guo, Peng Gao, Rongyao Fang, Bin Zhao, Dong Wang, Yu Qiao, and Hongsheng Li. Point-M2AE: Multi-scale Masked Autoencoders for Hierarchical Point Cloud Pre-training. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 27061–27074. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/ad1d7a4df30a9c0c46b387815a774a84-Paper-Conference.pdf.

[102] Renrui Zhang, Ziyu Guo, Wei Zhang, Kunchang Li, Xupeng Miao, Bin Cui, Yu Qiao, Peng Gao, and Hongsheng Li. Pointclip: Point cloud understanding by clip. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8552–8562, June 2022.

[103] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip Torr, and Vladlen Koltun. Point Transformer, September 2021. URL http://arxiv.org/abs/2012.09164. arXiv:2012.09164 [cs].

[104] Nicky Zimmerman. *Embedded Implementation of Reactive End-to-End Visual Controller for Nano-Drones*. PhD thesis, Università della Svizzera Italiana, 2020.