Image-based anomaly detection using β -Variational Autoencoder for surface vehicle collision avoidance

Johan Ahlqvist

André Skoog

LUND UNIVERSITY

Department of Automatic Control

MSc Thesis
TFRT-6121
ISSN 0280-5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Printed in Sweden by Tryckeriet i E-huset
Lund 2020

Abstract

Unmanned vehicles need robust systems to ensure the safety of the vehicle and its environment. Being able to find and avoid perilous situations is paramount to such a system. In this paper we suggest an unsupervised image-based anomaly detection algorithm using a variational autoencoder and a superpixel segmentation algorithm, which is adapted to the maritime obstacle detection task. The algorithm locates potentially hazardous objects and calculates the distance to them by measuring the images' reconstruction error over segmented regions. The algorithm's results on the public MODD2 dataset shows that it has difficulties finding small objects and that it cannot compete with the current state-of-the-art supervised segmentation algorithms on the same dataset, with an F1 score of 26.6% compared to 82.7%. Although further research and optimization is required to utilize the algorithm in a production level product, the results indicate that the algorithm is worth investigating further as it is able to detect many of the objects in our testing videos and due to it having applications in several areas.

Acknowledgements

Six months is a long time and when you are writing your thesis, it seems even longer. It is a time filled with countless hardships, challenges and accomplishments. Making it through this process on our own would have been difficult without our supervisors' support.

We would like to start with thanking our supervisor at Axis Communications, Mikael Lindberg. You have constantly worked hard to ensure that we have had everything necessary to succeed and maybe even more importantly, for always being willing to take extra time during your already busy work schedule to discuss our project.

We would also like to thank our contacts at SAAB Kockums, Jens-Olof Lindh and Jonas Rosqvist. Your enthusiasm and interest for our project as well as your deep domain-specific knowledge simplified our work tremendously.

From the university we would like to thank Anders Robertsson, Mikael Nilsson and Johan Grönqvist. Due to some unfortunate events, we were standing there in the beginning of February without a master's thesis project. Even though we had project prospects, we had no supervisor and the outlook for finishing the project this semester looked grim. That is when we were introduced to Anders Robertsson who took it upon himself to be our supervisor, which allowed us to start our project in March. We do not know how to thank you enough for all the time you put in to handle the administrative work and always being available for us.

Sometimes we have been lost for ideas as to how to improve our algorithm. During these moments, it has been comforting having the support of Mikael Nilsson and Johan Grönqvist. With their knowledge, they have always provided new perspectives and ideas for us to explore.

Finally we would like to thank everyone who has supported us along the way, our friends and loved ones without whom we might never have reached the finish line.

Contents

1

Introduction

A driver of a vehicle observes and acts on his surroundings. Traffic signs and other cues tell the driver what can and can not be done. In an autonomous vehicle, the same task may be performed by a system trained using supervised learning, in which an object detection model is taught to detect the presence of such cues. The performance of object detection algorithms has increased significantly in recent years, which has led to them being deemed sufficiently reliable to be used in modern, semi-autonomous vehicles present on public roads.

In 2016, a Tesla Model S with its semi-autonomous mode engaged was heading down a highway in Gainesville, Florida. Further up the highway, a tractor dragging a semi-trailer was crossing the highway perpendicular to the lanes, effectively blocking them. Due to the height of the trailer's undercarriage and its unusual positioning, the Tesla's software did not acknowledge the trailer as being an obstacle, resulting in the system not performing evasive actions [Golson, 2016].

The cause of this crash is likely complex, but it highlights a necessity for autonomous systems. While such a system can be taught to recognize many situations, there is a near infinite number of possible situations that can arise, however unlikely they may be. For this purpose, an autonomous system must also be able to recognize when it is facing an anomalous situation, i.e., a data sample which is unlikely to belong to the distribution known to the system [Zimek and Schubert, 2018]. In the case of the autonomous system, detecting these anomalous data points can be used to aid in the decision making process, possibly avoiding dangerous situations.

With the advancements in the field of deep learning, new techniques for obstacle detection algorithms are available. With this report, the authors aim to answer if it is possible to create an obstacle detection algorithm for surface vehicles using an autoencoder trained through unsupervised learning and what measures can be taken to increase the performance of such an algorithm.

2

Related work

A large amount of research has been conducted in order to provide autonomy for unmanned surface vehicles, or USVs. The research areas include autonomous navigation, obstacle detection and collision avoidance using a wide variety of sensor combinations.

Several methods have been tested for ship detection. One option is to segment the water line and then subtracting the water from the segmented area leaving only objects found in the water [Hu et al., 2011]. A similar approach uses discrete cosine transformation for the background subtraction followed by foreground segmentation [Zhang et al., 2017b]. An edge segmentation-based ship detector, using Sobel filters and support vector machines was proposed in 2015 [Eum et al., 2015].

Deep learning has broadened the possibilities. Deep learning architectures such as RPN and YOLO have introduced new methods for object detection, which were applied to enhance performance by Zhang et al [Ren et al., 2015; Redmon and Farhadi, 2017; Zhang et al., 2017b].

Combinations of sensors have also been tested. One approach is to fuse radar and camera data, using the radar data to locate objects of interest and the camera data to help the process of discarding false positives [Zhang et al., 2017a]. Another suggested approach used a combination of radar, GPS, AIS, cameras and many more to tackle the problem [Elkins et al., 2010].

Outside the naval area, a few unsupervised image anomaly detection algorithms have been proposed. One algorithm suggested the use of a hybrid autoencoder (HAE) which would combine features from deep Boltzmann machines, autoencoders and support vector machines to detect anomalies [Dairi et al., 2018]. A second variation was presented in 2016 where the developers used a hybrid solution using a deep belief network and a support vector machine for classification of anomalies [Erfani et al., 2016].

3

Theory

In this chapter, a brief theoretical background on the different components that constitute our anomaly detection system is given. Section 3.1 and Section 3.2 explain the terms anomaly detection and collision avoidance. This is followed by an overview of neural network related components, metrics and optimization methods in Sections 3.3 - 3.6. Relevant information regarding digital imagery, image preprocessing, segmentation and depth estimation is given in Sections 3.7 - 3.10. Finally, a short comment about the data set used for evaluation is presented in Section 3.11.

3.1 Anomaly detection

In 2002 former U.S secretary of defence Donald Rumsfeld delivered what is now known as "Rumsfeld information theory". In response to a question about the status of WMDs in Iraq the secretary answered:

> Reports that say that something hasn't happened are always interesting to me, because as we know, there are known knowns; there are things we know we know. We also know there are known unknowns; that is to say we know there are some things we do not know. But there are also unknown unknowns – the ones we don't know we don't know. And if one looks throughout the history of our country and other free countries, it is the latter category that tend to be the difficult ones. [United States Department of Defense, 2002]

Rumsfield information theory defines three subcategories of information. The first subcategory is the known knowns. This category handles the information which we know to be true e.g events that we know will happen. The second category is the known unknowns, which handles conditional information e.g events that have a probability of happening. And the final subcategory is the unknown unknowns which handle unknown conditional information e.g events that could happen which we are unaware of. The objective of an anomaly detector is to locate informational patterns which relate to the two latter categories.

The tragic accident in Gainesville described in Section 1 was caused by the system failing to handle an unknown unknown. The events that transpired underline the necessity for a collision avoidance system to handle previously unknown circumstances. Given a collision avoidance system which relies on camera sensors, the initial process may consist of an anomaly detection system.

An image-based anomaly detection system for collision avoidance can be divided into three sub-problems: anomaly classification, anomaly segmentation and anomaly localization. Given an input image **I**, anomaly classification can be considered as calculating a performance metric P_c over **I** such that if $P_c(I) \geq T_c$ where T_c is the classification threshold, **I** is labeled as anomalous.

Anomaly segmentation relates to classifying regions of an image as anomalous or not. Given an anomalous image I_a, anomaly segmentation can be described as extracting a feature set F from I_a and from that feature set finding a subset f, such that $\mathrm{f} \subseteq \mathrm{F}$ where $f_i \geq T_s$ where T_s is the segmentation threshold used to determine if feature f_i is to be considered responsible for labeling I_a as anomalous. An example of this process can be seen in Figure 3.1.

Anomaly localization is the process of calculating where a segmented anomaly is in relation to the camera. Anomaly localization can be described as, given an image segmentation, I_s, calculate the angle θ and the distance Δ to the segmented areas.

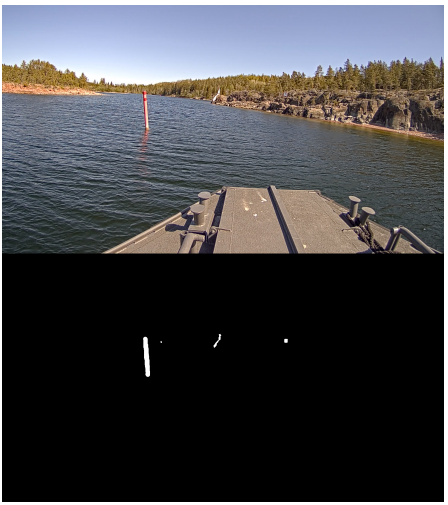


Figure 3.1: An illustration of how per pixel anomaly segmentation may look. The anomalous image I_a is shown on top and its anomaly segmentation is shown below it. The segmentation displays anomalous and non-anomalous pixels shown in white and black respectively. The most prominent white region corresponds to the closest navigation mark.

3.2 Collision avoidance

A collision avoidance system is an automated system designed to prevent or reduce the impact of a collision [Lim and Taeihagh, 2019]. One widely used collision avoidance system is the Adaptive Cruise Control (ACC). ACC uses sensors, e.g., lidar and radar, to maintain the distance between the sensor and the vehicle ahead by regulating the acceleration. In a fully autonomous vehicle, the collision avoidance system will need to be able to control the vehicle in all its degrees of freedom.

A collision avoidance system can be divided into two main components: sensors and a control module. These two components have different but related purposes. The task of a sensor is to detect features indicating obstacles whilst the control module's objective is to determine what action leads to the safest outcome given the sensor data as well as information regarding the vehicle and its environment. The algorithm suggested in this paper only constitutes a sensor component in this division. This is done in order to keep the solution as general as possible, as a control module requires data specific to the vehicle it is to control.

3.3 Neural networks

Basics

A neural network is an algorithm which estimates a function to transform a given input to a given output. This is done by passing the input data X through a series of nodes that manipulate the data. The nodes, which are also known as neurons, each have an associated weight and bias that together controls how the input is changed.

The weights are updated by an algorithm known as backpropagation [Rumelhart et al., 1986]. Backpropagation computes how to set the weights in order to reduce the error of a prediction $\hat{y}$ as compared to the ground truth y. The error between a prediction and the corresponding ground truth is given by a loss function L, which is chosen to fit the task at hand.

Two common problems types that herald distinct loss functions are *regression* and *classification*. Regression is the process of predicting a continuous quantity, while classification is predicting a label. Due to this fundamental difference, two different classes of loss functions are normally used. For regression, such a function is mean squared error while classification is normally done by some variation of cross-entropy.

Neurons are often followed by an activation function. They play an important role in neural networks as they allow controlled manipulation of the values flowing through the network, e.g., for normalizing a vector in order to interpret the output as a probability distribution over the output classes. Most activation functions are also nonlinear, which enables the network to estimate complex functions. A common such nonlinearity is the ReLU activation function, which is defined in (3.1) and illustrated in Figure 3.2, effectively allowing the neuron to be shut off if its input is

not needed [Nair and Hinton, 2010].

$$f(x) = \begin{cases} 0 & x < 0 \\ x & x \geq 0 \end{cases} \tag{3.1}$$

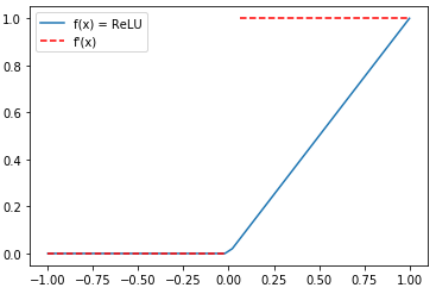


Figure 3.2: Illustration of the ReLU activation function, as shown in blue, and its derivative depicted as a dotted red line.

A neural network is commonly conceptualized as layers, where each layer performs a type of operation on the data. Some of the layers are trainable, i.e., having learnable parameters, whereas others are not, e.g., pooling operations and activation functions. Some common, trainable layer types include fully-connected layers, convolutional layers and recurrent layers, each with their own parameters such as the number of neurons and other, type-specific settings.

A neural network architecture is a set of layers connected in some fashion. The types of layers, their parameters and their intermutual connections together define the architecture, which is chosen to suit the task. For image-based tasks, architectures commonly incorporate convolutional layers, which unlike e.g., fully-connected layers utilize the local connectivity of the data.

The fully-connected layer

A fully-connected layer learns a regression from its input to its output. The regression can be defined formally as in (3.2), where f is the activation function, X is the output from the previous layer, ω are the weights, b are the biases and y_i is the output from neuron i.

$$y_i = f(X\omega + b) \tag{3.2}$$

The main characteristic of a fully connected layer is the connectivity. Specifically, all neurons of layer i are connected to all neurons of layer $i+1$. The structure of a network utilizing two fully-connected layers with five and four neurons respectively can be seen in Figure 3.3. The lines in between the nodes indicate connections.

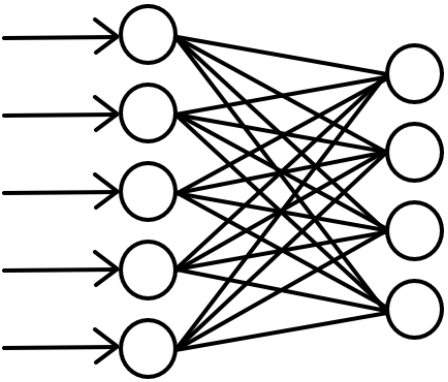


Figure 3.3: A visualization of a fully-connected neural network with two layers with five and four neurons respectively. Between the two layers are the connections which indicate that the layers are of the fully-connected variety, i.e., that every node in a given layer is connected to all nodes of the adjacent layers.

The convolutional layer

A convolutional neural network is a network which incorporates convolutional layers. [Goodfellow et al., 2016]. The convolutional operation describes the response of a system if a function f is subject to another function g. Formally the operation is defined as in (3.3).

$$s(t) = \int f(\tau)g(t-\tau)d\tau \tag{3.3}$$

The operation can be discretized and applied to higher dimensional data. Given a 2-dimensional image I and a kernel K, a 2-dimensional convolution can be defined as in (3.4).

$$s(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n) \tag{3.4}$$

An example of this operation can be seen in Figure 3.4. The operation passes a kernel, also known as a filter, over all positions in the original image and calculates a response for the given location, which results in a feature map.

Convolutional layers are beneficial due to three core characteristics: sparse connectivity, parameter sharing and local connectivity. Sparse connectivity means that only a few neurons from the input layer is connected to each neuron in the output layer. This reduces the memory requirements of the algorithm, allowing for deeper networks. Parameter sharing means that the parameters used to map from input to output are the same for the entire input. This is due to the only parameters being those in the kernel, which reduces the memory requirement of the network. Local

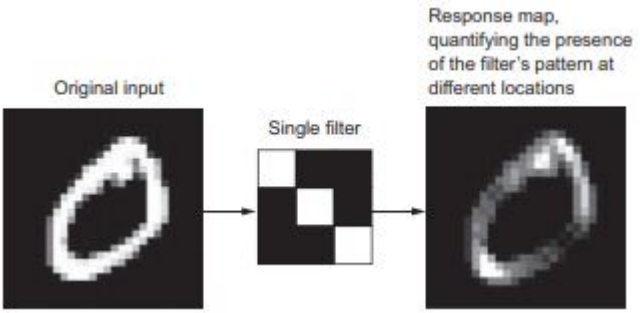


Figure 3.4: Illustration of the convolutional operation on an image. To the left is the original image depicting the digit zero. In the middle is the filter applied by the convolutional operation, and to the right is the resulting image. Image source [Chollet, 2017]

connectivity means that the convolution is performed over the local area and as such is independent of features outside of the current scope.

Autoencoders and variational autoencoders

An autoencoder is a type of neural network commonly used for representation learning [Bengio et al., 2012]. The network's architecture consists of two main parts, an encoder and a decoder. The objective of the encoder is to map an input, $x \in \mathrm{R}^d$ to a lower dimensional latent vector z. The decoder takes z as input and maps it to an output $\hat{x} \in \mathrm{R}^d$. The reconstruction error **rec_err**, i.e., the distance between x and $\hat{x}$ in some metric, is then used to update the network to better reconstruct the input. An example of such a reconstruction error is shown in (3.5).

$$L(x,\hat{x}) = \textbf{rec\_err} = \|x-\hat{x}\|^2 \tag{3.5}$$

The vector z, also known as the code, is a compressed representation of the input x. By updating the autoencoder's weights using samples from the domain, domain-specific encodings can be extracted which promotes sparse, approximate representations of similar input data. In the case of an autoencoder trained to reconstruct images of human faces, the code might contain values that are decoded to features in the reconstruction such as happiness, beardiness and whether the person is wearing glasses or not. An example of this can be seen in Figure 3.5.

Variational autoencoders (VAE) are structurally related to autoencoders, but makes the assumption that the latent vector variables follows a distribution. As such, the goal of the variational autoencoder is to find the distribution $q_\phi(z|x)$ of the latent variables in z. The latent variables will then be passed to the decoder, which will generate a new sample $\hat{x}$ based on its distribution $P_\theta(\hat{x}|z)$.

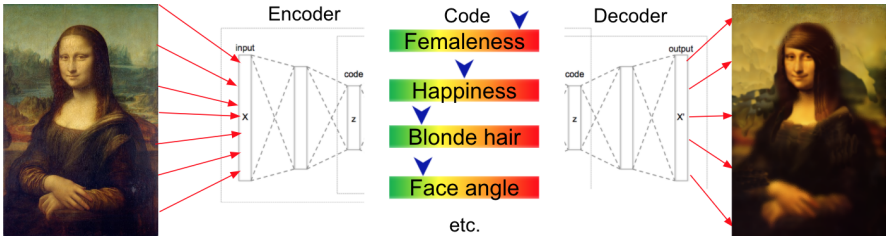


Figure 3.5: Illustration of how a variational autoencoder trained to reconstruct human faces might function. To the left is the input image, which is fed into the encoder component of the autoencoder. The encoder encodes the image into a code representation consisting of disentangled values where each value is related to some domain-specific feature, as depicted by the scales in the middle. The code is then decoded by the decoder and a reconstructed image is outputted, as shown to the right. Notice the imperfections in the reconstructed image.

To regulate the distributions of the code variables, a regularization term is introduced to the loss function. The regularization term is the Kullback-Leibler divergence, defined in (3.7), between p_θ and q_ϕ, which will penalize the network relative to the scale of the difference in the two distributions. This in turn will cause the latent variables to *disentangle*, i.e., prompting a one-to-one correspondence between input features and code values, as it penalizes them for relying on linear combinations of features. The loss function of a VAE can be seen in (3.6).

$$L(\theta,\phi;x) = \textbf{rec\_err} - KL(q_\phi(z|x)||p_\theta(z)) \tag{3.6}$$

$$KL(q_\phi(z|x)||p_\theta(z)) = \int_{-\infty}^{\infty} p_\theta(x)\textbf{log}(\frac{q_\phi(z|x)}{p_\theta(x)})dx \tag{3.7}$$

VAEs are generative models. The code of the VAE is stochastic, which means that the values of the code are sampled from a distribution. Therefore, if a VAE was tasked to reconstruct the same image N number of times, it would lead to N different reconstructions. Thus, unseen samples can be produced.

The reparametrization trick is necessary for the code to utilize sampled values. A stochastic function is not derivable and if one were used, the network would not be able to perform backpropagation. To solve this problem, it is necessary to separate the stochastic part from the network.

This separation can be performed by providing the random value through an auxiliary input to the network. The input provides values sampled from an $\mathcal{N}(0,1)$ distribution, henceforth denoted as ε. This is possible since inputs are not parameters that the network needs to update. The way to determine the value of the latent code variable $z_i \in z$ is described in (3.8) and illustrated in Figure 3.6, where μ and σ is the mean and standard deviation, respectively.

$$z_i = (\varepsilon_i \sigma_i + \mu_i) \tag{3.8}$$

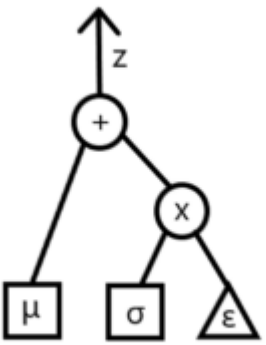


Figure 3.6: Image illustrating the reparameterization trick. Operations are shown as circles, trained parameters are shown as squares and inputs are shown as triangles. Together, the structure calculates the equation z = $(\varepsilon\sigma + \mu)$

β - variational autoencoders

A β-variational autoencoder (β-VAE) was introduced to provide disentanglement of the latent variables of a VAE [Mathieu et al., 2018]. The disentanglement is a consequence of the hyperparameter, β, added to the KL divergence term in the VAE's loss function. By changing the value of β, it is possible to change the ratio of importance between the reconstruction loss and the KL divergence. By incrementing β, the impact on the loss caused by the KL divergence will increase, which will force the distributions closer to each other. The loss function for a β-VAE is defined as in (3.9).

$$L(\theta, \phi; x) = \textbf{rec\_err} - \beta KL(q_\phi(z|x)||p_\theta(z)) \tag{3.9}$$

Anomaly detection with machine learning

There are many ways to perform anomaly detection. Some of the methods regard the nature of the data used for training the algorithm. Three such possibilities are the supervised, weakly supervised and unsupervised methods.

Unsupervised anomaly detection uses a data set without labels and tries to label samples as anomalous using statistical methods. Under the assumption that the majority of the data set comes from the same distribution, the probability of a data sample belonging to the same distribution can be calculated using the mean and standard deviation of the data set.

Supervised anomaly detection uses labeled data. This data is used to train a classifier to find a decision boundary such that it separates anomalous and normal samples.

Weakly supervised anomaly detection uses a data set for which imprecise labelling methods have been used. When using a weakly supervised annotation scheme, the data has been labeled using a set of rules or functions determined by the user. This means that there is no proof that the label of each sample is correct, but the labeling process can be done automatically.

What data labeling scheme to use depends on the problem domain. For a task where the amount of labeled data is insufficient in relation to the project's goals, using a supervised labeling scheme may not be the best option. A better option under these circumstances could be a weakly supervised or unsupervised scheme. In turn, in a situation where the margin for error is really small, it would be inadvisable to use a weakly supervised labeling scheme.

Perceptual loss

Basing the reconstruction loss on the distance between pixels might not be a sufficient method to solve an image reconstruction task. By calculating the mean squared error per pixel in an input image and its reconstruction, a metric for how well the system has reconstructed the image is obtained. However, this metric is solely based on the intensity values of the individual pixels and ignores the spatial information of sets of pixels. A problem such a naive reconstruction loss may cause is illustrated by Pihlgren et al., in Figure 3.7 [Pihlgren et al., 2020].

Transfer learning enables using perceptual loss. Transfer learning is the process of using knowledge gained through previous experiences and applying it to new problems. In case of image-based problems this means that parts of networks, like AlexNet, which have been trained on other tasks can be added to new networks in order to improve the networks performance on the new task [Krizhevsky et al., 2012].

Perceptual loss as described by Pihlgren et al., is defined as a loss calculated over some activations of an independent network. Formally this can be described as: given an image x and its reconstruction $\hat{x}$, extract features from x and $\hat{x}$ using the output of network N at layer i and calculate the distance between the feature sets. An example of perceptual loss using the euclidean norm is described in (3.10).

$$||N_i(x) - N_i(\hat{x})||_2 \tag{3.10}$$

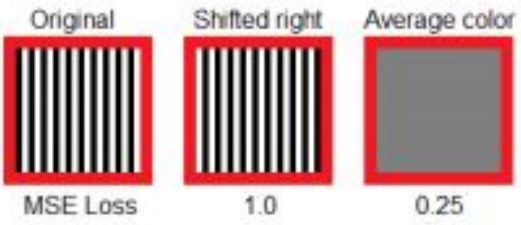


Figure 3.7: Example illustrating the issues with a MSE-based loss for image reconstruction. To the left is the input image and in the middle is the same image but with every pixel shifted slightly to the right. The rightmost image is simply the average color of the input image. While the middle image to a human observer looks more similar to the input than the rightmost image, a per pixel loss such as MSE would give the middle image a significantly higher reconstruction error than the rightmost image, as shown below the examples. Image courtesy of [Pihlgren et al., 2020].

3.4 Hyperparameter search

Hyperparameters are settings that affect the algorithm, but which are not optimized automatically. They may play a big role in the performance of the algorithm and thus needs to be chosen wisely.

While good value ranges can generally be calculated, finding the optimal values is commonly done by automated searching. There are many methods of hyper parameter searching. Two of them are grid search and random search. Grid search lets the user set a list of potential values for each parameter. The combinations are then evaluated and the combination with the best performance is saved. Random search works by the user setting ranges for the parameters from within their values are to be sampled. This method has been proven to be the better of the two options for hyper parameter search [Bergstra and Bengio, 2012].

3.5 Common neural network issues

When training a neural network it is necessary to consider a few common problems. Two commonly encountered issues in deep learning that have been accounted for in this project are the vanishing gradient and dying ReLU problems.

Vanishing gradient

A neural network learns through backpropagation. It is through backpropagation that the error contribution of every parameter is determined. By utilizing the chain rule, updates can be made for layers beyond the last one. Since each subsequent layer is dependent on the output from the previous layer, the network's hidden func-

tion can be considered a composite function of the functions of the layers. Thus, when the derivative of a layer is computed, the inner derivatives need to be considered as well. This is illustrated in (3.11).

$$\frac{d}{dx} f \circ g = f'(g(x))g'(x) \tag{3.11}$$

Choosing activation functions poorly can stop early layers from receiving updates. One of the functions that can cause this behaviour is the sigmoid function, which is illustrated in Figure 3.8. The sigmoid function takes an input in the range between -∞ and ∞ and returns a value between 0 and 1. More importantly, the derivative of the sigmoid function is strictly less than 1 for all possible inputs. This means that the inner derivative of a layer using the sigmoid activation functions will act as a shrinking factor for the gradient. If sufficiently many sequential layers use activation functions with similar properties, the gradient will shrink to the point where it will not provide updates to earlier layers. Mathematically this problem is presented in (3.12), where x is the average inner derivative and n is the number of layers.

$$\lim_{n \to \infty} x^n = 0, \quad x < |1| \tag{3.12}$$

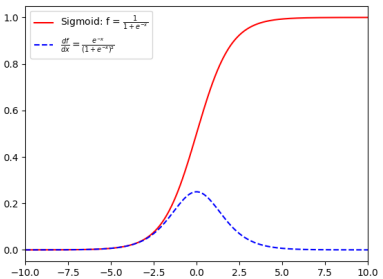


Figure 3.8: The sigmoidal activation function and its derivative. The sigmoid activation function squishes the output to be between 0 and 1. This causes the derivative to be small and converges on 0 as $|x|$ increases.

Dying ReLU

The dying ReLU problem can cause parts of the network to become inactive. This is due to the function setting all negative values to 0, as demonstrated in Figure 3.2. Consider the derivative shown in the figure, which is defined mathematically in (3.13). It is 0 for all negative input values and one for all positive values. In a state where all inputs are negative, the network would not be updated due to the

activation function always multiplying the gradient with 0. This condition is known as the dying ReLU problem.

$$f(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases} \tag{3.13}$$

Using the Leaky ReLU (LReLU) activation function mitigates the dying ReLU problem. LReLU is defined in (3.14) and illustrated in Figure 3.9. It differs from ReLU by leaking a fraction of the input for negative values. This ensures that the activation function's derivative is 0 only when the input is 0. If the network would reach a state where it would normally encounter the dying relu problem, LReLU can still cause updates which could resolve the problem. Noticeable is that choosing an α value of 0 makes the LReLU a ReLU activation function and choosing an α value of one makes it a linear activation function. It is thus advisable to choose an α value close enough to 0 such that it closely approximates a ReLU function, while still being large enough to cause updates to the network.

$$f(x) = \begin{cases} \alpha x & x < 0 \\ x & x \geq 0 \end{cases} \tag{3.14}$$

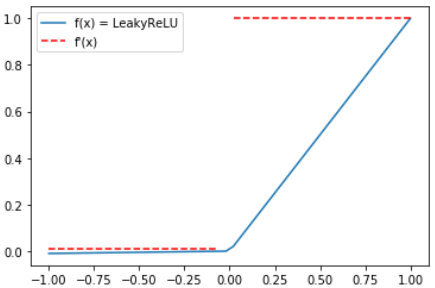


Figure 3.9: Figure illustrating the LReLU activation function and its derivate. The function's α is in this case set to 0.01.

3.6 Metrics

To be able to evaluate the performance of the anomaly detector, several metrics are necessary. The ones covered here are mean squared error, cosine similarity, binary cross entropy, receiver operating characteristics, F1-score and intersection over union.

Mean squared error

Mean squared error (MSE) is a loss function commonly used in neural networks. MSE calculates the average squared difference between feature sets. Formally it can be defined as in (3.15), where n is the number of samples, X_i is the feature vector of the i:th sample in X and $\hat{X}_i$ is the predicted feature vector of the same sample. MSE is a suitable loss function for regression type problems.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(X_i - \hat{X}_i)^2 \tag{3.15}$$

Binary cross entropy

Binary cross entropy (BCE) is a loss function commonly used in neural networks. Given the true label and the predicted probability of the sample belonging to that label, the binary cross entropy can be calculated as seen in (3.16), where y_i is the ground truth and $\hat{y}_i$ is the predicted probability for that class and n is the number of samples. Binary cross entropy is a suitable loss function in classification problems.

$$L(y,\hat{y}) = \frac{1}{n}\sum_{i=1}^{n} y_i \cdot log(\hat{y}_i) + (1-y_i)(log(1-\hat{y}_i)) \tag{3.16}$$

Cosine similarity

The cosine similarity is a similarity metric that examines the cosine value of the angle between two feature vectors. Formally it is defined as in (3.17).

$$\textbf{similarity} = cos(\theta) = \frac{A \cdot B}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{3.17}$$

Thus, this metric calculates a score based on how large the angle between the two vectors A and B are. A consequence of this is that vector scale is disregarded.

Receiver operating characteristic

Receiver operating characteristic (ROC) is a metric which evaluates a binary classifier. It outputs a confidence by varying the decision boundary of the classifier and comparing the true positive rate and the false positive rate. The related ROC area under curve (ROC AUC) metric is evaluated by calculating the area created under the ROC curve. The true positive rate (TPR) and the false positive rate (FPR) is described in (3.18) and (3.19) respectively.

$$\textbf{TPR} = \textbf{Recall} = \frac{\textbf{True positives}}{\textbf{False positives + True positives}} \tag{3.18}$$

$$\textbf{FPR} = \frac{\textbf{False positive}}{\textbf{False positives + True positives}} \tag{3.19}$$

The ROC curve illustrates the predictor performance for all confidence thresholds. The illustration simplifies choosing a decision boundary, which matches the requirements set by the application.

F1 score

The F1 score is a measurement related to a test's accuracy. It is computed as the weighted harmonic mean of the test's precision and recall. The F1 score is useful for getting an evaluation which balances both precision and recall and as such giving a more realistic measurement of the networks performance. Recall, precision and F1-score are defined as in (3.18), (3.20) and (3.21) respectively.

$$\textbf{Precision} = \frac{\textbf{True positives}}{\textbf{True positives + False positives}} \tag{3.20}$$

$$\textbf{F1} = 2 \cdot \frac{\textbf{Precision} \cdot \textbf{Recall}}{\textbf{Precision} + \textbf{Recall}} \tag{3.21}$$

Intersection over union

Intersection over union (IOU) is a metric which calculates how well two areas overlap. This is done by calculating the two areas' intersecting area and dividing it by their combined area. IOU is commonly used in object detection tasks to determine how well a predicted bounding box matches a ground truth bounding box. The metric is defined as in (3.22).

$$\textbf{IOU} = \frac{A \cap B}{A \cup B} \tag{3.22}$$

3.7 Digital image representation

A color model is a numerical representation of the color spectrum. Digitally, these color models can be represented as tuples, where each value indicates how much of a characteristic is prevalent in a pixel. A common color model is the RGB model which describes a pixel using red, green and blue color saturation.

The choice of color model is important for image segmentation performance. Previous research has shown that the choice of color model is an important factor to improve image segmentation results [Chebbout and Merouani, 2012]. Two of the commonly used color models are the HSV and LAB models. Comparing these two methods has shown that HSV generally performs better for image segmentation [Bora et al., 2015].

3.8 Data preprocessing and augmentation

Preprocessing Preprocessing the input data to a neural network is important. The preprocessing modifies the representation of the data that the network is given,

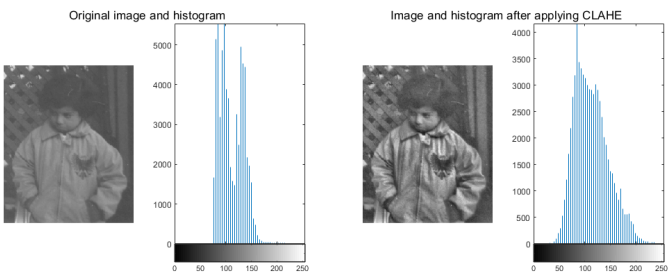
which in turn impacts how well it performs. For example, a representation of the data which makes important features easily interpretable may speed up learning and decrease model capacity requirements as less work is needed to extract important features from the input data.

A common preprocessing step in deep learning is normalization. Normalization is done by modifying the input data in some way in order to have the different input features be of similar sizes, which in turn simplifies adjusting the network's weights to fit the data. Commonly, this involves scaling the features to the (0,1) range, e.g., by dividing by the maximum possible value or through Z-score normalization, which is described in (3.23).

$$z = \frac{x - \mu}{\sigma} \tag{3.23}$$

Histogram equalization is a method used to increase contrast in images, which can also be used during the preprocessing step. By examining the histogram in Figure 3.10a, it can be seen that the image has a majority of the pixel intensities in the range between 75 and 150. Histogram equalization is a monotonic function that mitigates this issue by spreading the intensity values over the entire available spectrum.

Contrastive limited adaptive histogram equalization (CLAHE) is an advanced equalization method. It works on subregions of the image and limits the spectrum that a pixel can get its intensity value from. As such, CLAHE is able to limit the amplification of noise. Applying CLAHE to the photograph in Figure 3.10a modifies it as shown in Figure 3.10b.



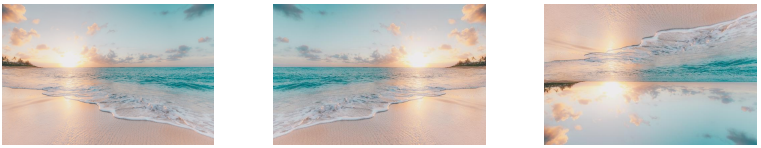
(a) (b)

Figure 3.10: Two images depicting a photograph with and without the histogram equalization method CLAHE applied. The image is accompanied by its histogram which shows the distribution of pixel intensities. The unmodified photograph is shown in 3.10a and its modified counterpart in 3.10b.

Data augmentation The amount of training data impacts the algorithm's performance on unseen data. A machine learning algorithm aims to learn a function from the input to the output. It does so by evaluating the function value for a sample and comparing the output to a given ground truth. Depending on the error given by this comparison, the function is subsequently updated to better reconstruct the ground truth. This means that the function the algorithm finds is reliant on the data it has been provided. If the input data provided does not represent the input domain well enough, the function's ability to generalize, i.e., make accurate predictions on unseen samples, will be poor.

Data augmentation is used to alleviate some of the issues related to the data requirements. This is done by manipulating an original sample to create a new sample with slightly modified features. In the image data case, such augmentations may include changes to e.g., rotation and scale.

It is important that the augmentations stay within the input domain. The augmentations can create a large variety of new samples, some of which would never be seen in the input domain. As an example, if an image portraying an ocean and sky is flipped vertically, it will create a sample image with the sky below the ocean, which is not how landscape images are generally represented. This example can be seen in Figure 3.11. This augmentation should therefore not be used to modify a data set containing such images.



(a) Original image (b) Horizontal flip (c) Vertical flip

Figure 3.11: Three images depicting possible augmentations of a landscape photo. 3.11a contains the original image while 3.11b and 3.11c are augmented images. While 3.11b is likely to be in the domain of possible samples, 3.11c is not as it strays from how a landscape photo is generally represented.

3.9 Image segmentation

Image segmentation is the process of dividing an image into partitions. By segmenting the image, it is possible to construct areas which only contain objects of interest. Using these areas, it is possible to separate objects from the background. An example of a segmented image is shown in Figure 3.12, in which an image containing a laptop, a cable, a glass, a table surface and a background wall has been segmented into five differently colored regions corresponding to the objects in the original image.

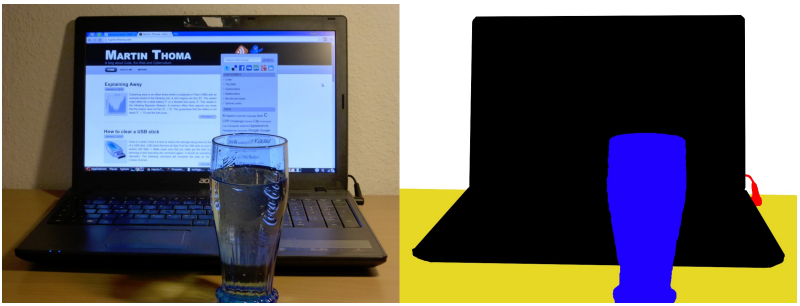


Figure 3.12: An illustration of image segmentation. To the left is the input image and to the right is the same image but segmented into five differently colored regions corresponding to the objects in the original image.

A wide variety of image segmentation methods are available, such as Simple Linear Iterative Clustering (SLIC), Bayesian adaptive superpixel segmentation [Uziel et al., 2019] (BASS) and QuickShift (QS). While these methods are generally quite slow, Fulkerson implemented a modification of the QuickShift algorithm which runs on GPU, which may significantly speed up the process [Fulkerson and Soatto, 2010].

QuickShift The QuickShift algorithm is defined as, for each pixel (x, y), QuickShift regards x, y, $I(x,y)$ as a sample from a $d+2$ dimensional vector space and calculates the Parzel density estimation as seen in (3.24). The pixels are then connected to the closest higher density estimation that fulfills the property described in (3.25). The algorithm can then choose useful superpixels using the maximum distance parameter τ [Abdelhameed et al., 2014].

$$E(x,y) = \sum_{x'y'} \frac{1}{(2\pi\sigma)^{d+2}} \exp\left(-\frac{1}{2\sigma^2}\begin{bmatrix} x-x' \\ y-y' \\ I(x,y)-I(x',y') \end{bmatrix}\right) \tag{3.24}$$

$$dist(x,y) = min_{(x',y')>P(x,y)}(x-x')^2+(y-y')^2+\|I(x,y)-I(x',y')\|_2 \tag{3.25}$$

3.10 Image-based depth estimation

Image depth estimation is the process of measuring distances using image data. The task of estimating the distances to objects in the world using ocular information is difficult. It requires the estimator to combine a set of 2D images to create a 3D estimation of the world.

Depth cues

Pictoral depth cues Pictoral depth cues are cues which are based on perceived sizes, orientations and occlusions of objects in a 2D plane. These can be used to determine objects' relative distances given some prior knowledge about the objects. Figure 3.13 shows a bookcase from which it is possible to determine which side of the bookcase is farther away by finding the edges of the bookcase and measuring which vertical line is longer.

Figure 3.13: A photograph depicting a bookcase with its edges marked with red lines. By comparing the lengths of the vertical lines, it is possible to determine which side of the bookcase is closest as both sides of the bookcase are of the same height in real life.

Given two identical objects, observed from the same direction, it is possible to determine which one is closest by comparing the image area cover. This is illustrated in Figure 3.14 where the trees get smaller and smaller the farther away they are.

Figure 3.14: A photograph depicting trees at the side of a road. Given that the trees are of similar size it is possible to determine which tree is closest by comparing the image area cover of each tree.

The observant reader will notice that in Figure 3.14 it is possible to find edges along the road similar to the ones found in Figure 3.13. By traversing along the

edge of the road towards the horizon, the trees also decrease in size showing the correlation between the cues.

Depth cues from motion The perception of an object's movement is relative to the distance between the object and the observer. This relativity is called motion parallax [Kim et al., 2016]. Motion parallax can be observed when two objects at different distances to the observer travel the same distance within a given time frame. They will seem to have travelled different distances to the observer. This phenomenon is due to the change in angle from the observer to the objects as caused by their movement. The change in angle can be described mathematically as in (3.26), where h is the distance to the object and b is the distance travelled. This indicates that objects at larger distances will have a smaller angular change and thus appear to move less, which is also illustrated in Figure 3.15.

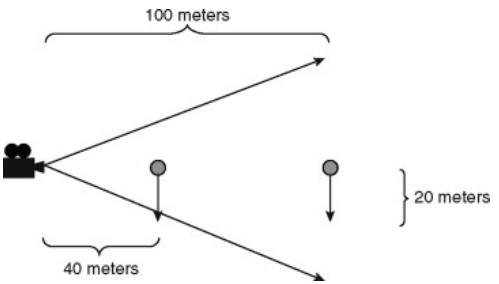


Figure 3.15: Illustration of an observer who observes two objects. When the objects travel a set distance, the angle at which the observer perceives the objects change. The difference in the angle is relative to the objects' distance from the observer. Image source [Brinkmann, 2008].

$$\theta = tan^1(\frac{b}{h}) \tag{3.26}$$

Stereopsis Stereopsis is the perception of a 3-dimensional environment using images from two sources. It is similar to motion parallax, but instead of having one observer taking two images with a time difference, two observers at slightly different positions are used to take images at the same time. Since the distance between the observers is known, it is possible to estimate the distance to an object by comparing the difference of an object's position in the two image sources.

Algorithmic implementations

Optical flow Optical flow is a motion parallax algorithm which can be used for computer-based depth estimation. The motion of an object in a two dimensional image plane indicates the object's motion relative to the observer. Using said motions,

tasks such as object segmentation and relative depth estimation can be performed [Beauchemin and Barron, 1995].

Stereo vision Reconstruction of a 3-dimensional space using stereo vision requires rectification of the images. Rectifying images from stereo cameras is the process of projecting the images onto the same plane such that they are coplanar. To be considered coplanar the two images need to conform to two properties: all epipolar lines need to be parallel to the horizontal axis and have corresponding points at identical vertical coordinates. The task of finding corresponding points is made easier by using a flat object depicting a known pattern, usually a checkerboard. With corresponding points known, the projective transformation of the images can be computed [Zhang, 2000]. If the cameras are parallel with the same focal lengths, calculating the distance to the point can be done using (3.27), where Z is the distance, f is the focal length of the cameras, B is the distance between the cameras and d is the disparity for the point [Lim et al., 2017].

$$Z = \frac{fB}{d} \tag{3.27}$$

Corresponding points matching When a pattern is not available, a matching strategy is required. This would be slow as it would require the algorithm to search all possible locations for a matching candidate. The coplanar constraint reduces the search space significantly as the only points necessary to consider are the points along the horizontal line, which makes the process more feasible.

Several matching strategies are available. A naive approach to the problem would be to examine each point along the horizontal line and matching with the pixel with the least difference. More sophisticated methods such as graph cuts, Markov random field minimization and semi-global block matching are also available [Boykov et al., 1999; Szeliski et al., 2008; Hirschmuller, 2008].

3.11 Evaluation dataset

The Multi-modal Marine Obstacle Detection Dataset 2 (MODD2) is a maritime data set for semantic segmentation [Bovcon et al., 2018b]. The data set consists of 28 video sequences where the water and maritime objects have been manually annotated.

Algorithm performance on the data set is determined using a supplied Matlab script. The script has two major evaluation areas: segmentation of the water line, i.e., where the water and the sky intersect, and object segmentation. How well the algorithm segments the horizon line is based on the MSE compared to the ground truth. How well an algorithm segments objects is determined by the intersection over union, true positive rate, false positive rate, and false negative rate.

Certain specific rules apply to the dataset. The minimum intersection over union for a detection to be valid is 0.15 and objects smaller than 5x5 pixels are disregarded.

Additionally, the dataset's definition of an anomalous object is one which have the entirety of its segmentation below the water line, i.e., all objects that cross the water line or are completely above the water line are disregarded.

Figure 3.16: An image from the MODD2 dataset with a cargo ship in the background. Using the evaluation scheme of MODD2, the ship would not be declared anomalous, even though it poses a potential problem for the USV, as its highest portion is above the water line.

4

Resources used

Access to cameras and surface vehicles were necessary for the project. Axis Communications provided cameras and SAAB Kockums provided surface vehicles and means of data gathering.

4.1 Cameras

FA1105

The camera used for this project is the FA1105 seen in Figure 4.1. The FA1105 sensor unit is a small camera measuring 29x20 mm making it a suitable choice for mounting on a small surface vehicle. The camera has a 111° horizontal field of view and a 61° vertical field of view. It is able to capture 30 frames per second in 1080p resolution.

Figure 4.1: FA1105 sensor unit. Image source [*FA1105 product page* 2017]

FA54 Main Unit

The FA1105 sensor unit requires a main unit. The main unit chosen for this project is the FA54 seen in Figure 4.2. The FA54 main unit allows for the usage of up to four cameras.

Figure 4.2: FA54 main unit. Image source [*FA54 product page* 2016]

4.2 Surface vehicle

The surface vehicle this project is designed for is the Piraya manufactured by SAAB Kockums and seen in Figure 4.3. The Piraya is a small unmanned surface vehicle measuring four meters in length. The Piraya allows for a top speed of 20 knots. The Piraya hosts numerous sensor capabilities such as lidar, radar, GPS and two FA1105 camera sensors and a FA54 main unit. However, for image data gathering, SAAB Kockums provided other surface vehicles.

Figure 4.3: Two unmanned surface vehicles of the Piraya class. Image source: [*Clearing the mine threat* 2017]

4.3 Computational resources

Axis Communications provided computational resources. Both cloud and local resources were made accessible, but most work was done on a desktop PC using an Intel 9700K CPU and a NVIDIA 2080 Ti GPU.

5

Methodology

5.1 Neural network

The model used in our approach is a convolutional neural network with a β-variational autoencoder (β-VAE) architecture. The input image, which has a size of 1024x1024 pixels and three channels is passed through the encoder. It encodes the image to a latent space, producing a code with 512 dimensions. The decoder then tries to reconstruct the image from the code.

The encoder and decoder consists of several consecutive ResNet-blocks [He et al., 2016]. Each such block consists of three convolutional layers forming two paths: one with two layers and one with one layer. The output from the two paths are summed before being connected to the next block. This allows for deeper networks as it mitigates the vanishing gradient problem by allowing the flow of the gradient to take shortcuts. To avoid the dying ReLU problem, in which an activation outputs nothing but zero and thus gets stuck in an unrecoverable state, every convolutional layer is followed by a LeakyReLU activation with a default α of 0.3.

In each block of the encoder, the spatial resolution is halved while the number of filters is doubled. This is done in order to continuously compress the data and distill it into more abstract features. After eight such blocks, the input has been reduced from 1024x1024x3 to 8x8x192 values and is then fed through the reparametrization trick logic of μ and $\log(\sigma)$ and through global average pooling to reduce the code parameters and then finally to the latent space z. The decoder follows the same structure but with the layers in reversed order and using transposed convolution instead of convolution. A visualization of the general structure of the network is shown in Figure 5.1.

Throughout the experiments, the neural network architecture and configuration remained largely the same, with changes mostly concerning the scale of the network.

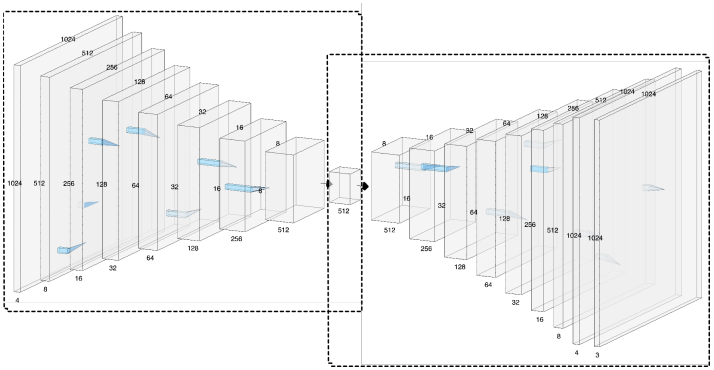


Figure 5.1: A visualization of the general structure of our autoencoder. The flow of data is from the left to the right, with the left dotted rectangle encapsulating the encoder related components and its right counterpart encapsulating the decoder components. The intersecting area of these rectangles holds the code. Each block is accompanied with its output dimensions, with the height on top, the width to the left and the number of channels below each block, except for the code which is one dimensional and thus only shows its number of neurons. The visualization is simplified: each visualized convolutional layer corresponds to a ResNet block in the actual model and neither the global average pooling used before the code nor the reparametrization trick logic is shown.

5.2 Neural network training data specifics

A neural network is wholly dependent on the data it is trained on. Due to this, Saab supplied the project with suitable video footage of a surface vehicle's perspective of the archipelago. This video was divided into six separate sessions, with one of the sessions being recorded in four different directions from the ship.

The video footage was recorded from two vehicles: a small civilian boat and a larger, military-class boat. In both cases, the camera placement includes some of the structure of the vehicle. The smaller boat has two railings clearly visible in the images while the recordings from the larger boat has much of the mast and forward area of the vehicle present. A typical image from the training dataset can be seen in Figure 5.2.

The data was divided into two data sets. As the algorithm is an anomaly detection system, the data which containing objects which could pose a danger to a ship and that were not naturally occurring and static were removed from the training set, in order to make these features anomalous. The argument for drawing the classification line at naturally occurring and static objects were that these objects, while posing a danger to ships, were easily avoidable by using nautical charts.

The process of data selection was done manually. Images were looked through for objects that met the criteria and subsequently placed in the clean or non-clean

Figure 5.2: A sample from the image dataset used to train the neural network. This specific image is taken from the small boat, which can be seen by the railings visible in the image.

categories. This process was quite fast as the images were frames of a video and thus sequential, which enabled labelling chunks of images as it came down to finding the first and last frame containing each object. For images containing very small objects or when the classification was for other reasons not obvious, a non-clean classification was given as to keep the training data as clean as possible.

The data consists of about 200000 images of archipelagos from the east coast of Sweden. Categorizing the frames into clean and non-clean data took roughly two days. After this process, about 88000 frames were retained as training data, 69000 frames were labelled as non-clean and 49000 frames were discarded for some reason, e.g., due to repositioning of the camera causing blocked views.

The images were resized prior to training. Resizing is necessary since the resolution of the original images is 1920x1080. This is due to the network requiring an input size of 1024x1024. As the process of resizing an image is slow, it was done using bicubic interpolation before training to reduce the input data loading time.

5.3 Data augmentation

Data augmentations were applied to the data set. Throughout all experiments data augmentations were performed to limit overfitting of the model. Augmentations were chosen carefully to not stray too far from the data distribution.

The random augmentations performed were rotation, cropping, histogram equalization and horizontal flipping. The cropping was performed independently for each side of the image. The specific augmentation settings used can be seen in Table 5.1.

Data augmentation settings	
Setting	*Value*
Rotation (deg)	± 3
Cropping (px)	[0,16]
Histogram equlization (p)	0.5
Horizontal flipping (p)	0.5

Table 5.1: The settings used to augment the autoencoder's input data during the fitting process.

5.4 Training scheme

The training scheme of the model regards the settings of its training process. These are important for making the training efficient and for the final performance of the model to be as good as possible.

Throughout the project, roughly the same training scheme settings were used. The number of epochs was set to 25, which took roughly 50 hours to complete. The initial learning rate produced instabilities when set to values higher or equal to 0.001 and was thus set to 0.0003. Additionally, it was decreased by multiplying the current learning rate with 0.1 every 10 epochs. The batch size was set to 8, with higher values leading to GPU memory issues due to the large dimensions of the perceptual loss features.

For evaluation purposes, a small validation data set of 5% of the data was used. The validation data was set up to be a contiguous sequence of frames in order to minimize fitting to images similar to the validation data. However, as other images from the same videos had been trained on, a much larger test data set on completely unseen videos was used to get a better measure of how well the learned features generalized. In order to also get qualitative comparisons during training, a specific validation image was reconstructed at the end of each epoch.

5.5 Tiling

Tiling divides the image into sub regions. In the experiments using tiling, it was used to split the image into quadrants. Tiling was used to allow the network to process the image at a higher resolution without increasing its capacity. Additionally, tiling allowed not having to downsample the reconstructed image to fit the perceptual loss network input. Thus, when using tiling, the input to the network was an image of size 512x512 pixels. Four such predicted images were then merged to form the reconstruction of the full, original image.

5.6 Code pertubations

A technique used in most of the experiments was code perturbation. By altering the code produced by the encoder processing an input sample, values mapping to image features are changed. By decoding the altered code, the resulting reconstruction will be different from the original reconstruction.

The technique can be used for direct classification. Given that the β-VAE has learned a disentangled distribution, it is possible to vary specific image features of a reconstruction by perturbing the code. Indicated by Wang et al, these changes to the code will cause larger variation in anomalous data samples due to structural consistency, which makes it possible to detect anomalous regions based on the variation produced by the code perturbations [Wang et al., 2020].

The pertubations were also used for smoothing. This was done by producing a set of randomly altered reconstructions and then using the minimum error per pixel as compared to the original image, i.e., choosing the best reconstructed pixel to represent the position within the reconstructed image.

5.7 Initial approaches

Thresholded reconstruction error The initial approach for image anomaly detection using a VAE was to use the reconstruction error in the simplest possible way. This was done using the absolute difference, the error, between the original image and the reconstruction.

A threshold on the error was used to classify pixels as anomalous or not. Pixels with an error magnitude exceeding the threshold were labelled as anomalous. This is shown in (5.1), where x is the input image, $\hat{x}$ the reconstruction, x_p an individual pixel and T the error threshold. Anomalous pixels are labeled with 1 and non-anomalous pixels are labeled with 0.

$$f(x_p, \hat{x}_p) = \begin{cases} 1 & |x_p - \hat{x}_p| > T \\ 0 & |x_p - \hat{x}_p| \leq T \end{cases} \tag{5.1}$$

Perturbation error The second iteration of the algorithm for anomaly classification used a β-VAE with perturbations. The code of a β-VAE was perturbed to produce slightly varied reconstructions. For each pixel in every perturbation, the reconstruction error was checked against a set threshold. Then, if a pixel had an error above the threshold in all perturbations, the pixel would be marked as anomalous.

The thought behind this experiment was that the perturbations would cancel out noise. This was due to the noise consisting mainly of imprecise water lines, tree lines and high frequency details. On the other hand, it was thought that regions containing anomalous objects would have a high error in all perturbations, retaining the true positives while cancelling the noise.

Applying perceptual loss Another attempted modification was to train a β-VAE using perceptual loss. When using perceptual loss, the image and its reconstruction are passed to a pre-trained image classification network such as AlexNet, for the purpose of extracting features from them. The differences in the feature sets are then used for classification purposes.

The network used for this task was the EfficientNet-B7 model [Tan and Le, 2019]. Its weights had been trained on the ImageNet dataset containing images of 1000 object classes with the self-supervised Noisy Student technique [Deng et al., 2009; Xie et al., 2020]. This network was chosen as it is one of the current top performers in object recognition and because of the unusually high resolution of its input. Specifically, the EfficientNet-B7 model uses images of size 600x600 as input, which means that less information would be discarded when downsampling the reconstructed image for the loss function, as compared to other models.

5.8 Superpixel segmentation methods

The final group of methods adds superpixel segmentation to the algorithm. These segmentation techniques were used to encapsulate objects in the image data. It was reasoned that by utilizing the probable object boundaries in conjunction with the reconstructed image, it would be easier to perform anomaly classification and segmentation. An overview of how this algorithm works can be seen in Algorithm 1.

To get as good results as possible, different superpixel segmentation methods had to be considered. The methods were examined based on time per image and the method's ability to create an encasing segmentation of objects in the image. The results of the experiment can be seen in Table 5.2 below. The tests were conducted using a computer with a NVIDIA 1050 Ti GPU and an Intel Core i5-7300HQ CPU.

Segmentation experiment				
Segmentation method	*Time/image*	*Segmentation performance*	*Usable*	*Platform*
QuickShift	18.84s	Good	No	CPU
Slic	1.6s	Poor	No	CPU
Felzenschwalb	2.13	Poor	No	CPU
BASS	27.54s	Excellent	Potentially	GPU
QuickShift++	0.67s	Good	Yes	GPU

Table 5.2: Table illustrating the results from testing the performance of different superpixel segmentation methods. The hardware used was a NVIDIA 1050 Ti GPU and an Intel Core i5-7300HQ CPU.

During testing, it was made obvious that only QuickShift++ filled the requirements of this project. This was due to the necessity of the segmentation algorithm to

segment objects well along their borders with an acceptable frame time of no more than 1 second. BASS was found to have the shortcoming of being remarkably slow, but with excellent segmentation performance. Given a faster implementation of the algorithm, it would be a contender. However, Quickshift++ was chosen due to its combination of frame time and segmentation performance. Its settings are presented in Table 5.3.

QuickShift settings	
Setting	*Value*
σ	2
τ	10

Table 5.3: The settings used for the QuickShift++ algorithm. The settings are σ, which is the Parzel density standard deviation and τ is the maximum distance between a pixel and its centroid.

Parallel to this, a β-VAE was used to create perturbed reconstructions of the original image. Four perturbations were created by adding random vectors sampled from $\mathcal{N}(0, 0.1)$ to the code outputted by the encoder. These were concatenated with the original code and used as input to the decoder, resulting in five slightly varying reconstructions of the input image.

Once the perturbed reconstructions and the segmentation were completed, error averaging was performed. This was done by calculating the absolute values of the subtraction between the input image and each of the reconstructions. This formed a volume of size (N, W, H, C), where N is the number of reconstructions and W, H, and C are the image dimensions. The (N, W, H, C) volume was averaged over the color channels, which reduced the volume to an (N, W, H) array. While giving the individual channels special treatment might prove advantageous to detecting certain situations, it was decided that the algorithm should be designed neutrally in this regard, as to encourage its ability to detect anomalies.

Next, the array was reduced to a 2D image. It was noted that the interesting objects, i.e., anomalies, usually had quite consistent pixel values over the reconstruction errors. However, this was not seen to the same degree with noise and errors the authors classified as uninteresting. This led to reducing the array's first dimension by taking the minimum of each reconstruction error, resulting in an image of size (W, H), where each pixel corresponded to the minimum reconstruction error for the reconstructions' channel-averaged pixel values.

The segmentations and the refined reconstruction error were then combined. This was done by averaging the error for every segment region, thus producing a value for how incorrectly that segment was reconstructed. As these regions correspond to objects or parts of objects in the image, the resulting combination would indicate how well an object is reconstructed. If a region had an error above a set threshold it was marked as anomalous.

Algorithm 1: Monocular superpixel anomaly detection

```
z ← encode input image
zp ← z + random vectors
reconstructions ← decode zp
segmentation ← segment input image
reconstruction error ← |input image − reconstructions|
reconstruction error ← average reconstruction error over its channels
reconstruction error ← minimum reconstruction error over its perturbations
anomaly mask ← False like input image
for all s: segmentations do
  if average reconstruction error of region s > T then
    anomaly mask[s] ← True
  end if
end for
return anomaly mask
```

The algorithm's speed depends in part on the number of segments. This is due to the averaging of the error being performed over each segment region. Thus, more regions lead to more masks being calculated and more averaging operations performed. Further, the QuickShift algorithm does not have a parameter which directly limits the number of segments produced, which resulted in certain images having thousands of tiny segments in homogenous regions, such as blue skies.

For performance and noise reduction reasons, segments with an area of less than 32 pixels were removed. This roughly corresponds to a 6x6 pixel object. Once neighbouring anomalous segments have been joined, the same removal of small objects was performed but with a minimum area of 128 pixels. The values were chosen simply based on how small objects detection was needed for. At a later stage in the algorithm's development, this removal of small objects was disabled in order to treat all detections equally.

Temporal matching addition

For the purpose of reducing the amount of false positives, the previously described superpixel segmentation method was extended with segment matching over time. This was done by extracting a few hand crafted features from the segments. These features were used to match segments from sequential time steps. The window size w, which decides the number of time steps, was set to five. Segments were considered to match if the mean squared error between their features was below a fixed threshold. An overview of how this algorithm works can be seen in Algorithm 2.

The algorithm for matching over time consisted of three simple steps. First, the erroneous segments stemming from the most distant time step $t-w$ was used to initialize the set of active segments S_{t-w}. Next, for each active segment, the best

match in the set of erroneous segments from the next time step, $t - w + 1$, was calculated. The matches that had a similarity meeting the threshold were then selected to be the new set of active segments. This step was iterated until the current time step $t - w + w$ was reached. The resulting set of active segments, if any, were then considered validated anomalous segments.

Algorithm 2: Temporal monocular superpixel anomaly detection

```
anomaly candidates list ← perform Algorithm 1 on input image
create segment features list
for all a: anomaly candidates do
    segment feature list ← extract features(a)
end for
validated segments ← match segment features list with previous iterations
anomaly mask ← False like input image
for all s: validated segments do
    anomaly mask[s] ← True
end for
return anomaly mask
```

The choice of features for the temporal matching process is task-specific. This is due to the features needing to be able to discern between different objects in sequential frames and because the properties of those objects differ based on the task. For an obstacle detection system in a car, the features must reflect that an object's position and direction may change a lot between frames and still be the same object. However, in the case of maritime obstacle detection, an object is unlikely to move a long distance between frames and its shape and size will likely remain similar.

The chosen features exploit the properties of maritime objects. As most such objects are seen from a distance and move slowly, their shape and position also changes slowly. Thus, width, height, centroid and size, i.e., the number of pixels an object occupies, were chosen as features. All features were normalized to the $[0, 1]$ range by dividing them by the individual image property they were measured in. In order to emphasize the importance of locality, the corresponding features were given weights of 5, while the other features had a weight of 1.

As the segmentation algorithm does not have the same concept of objects as a human might, it will likely produce several segments per object. To mitigate having several anomaly detections per object, neighbouring segments that had been detected as anomalous were joined into a single, larger segment. This is also a crucial component for temporal matching due to the shapes of the segmented regions changing in an unpredictable manner in each frame.

Stereo vision addition

For the purpose of reducing the amount of false positives further and finding the relative positions of anomalies, the previously described temporal matching method was extended with stereo vision. The stereo method of the algorithm runs two instances of the temporal matching, monocular method in parallel with some additional combining logic. Each instance is responsible for the image stream from one camera. An overview of how this algorithm works can be seen in Algorithm 3.

When an anomaly has been detected in the left image, the detection is compared to candidates from the right image. Due to the images having been rectified, it is known that corresponding points will exist on the same horizontal line in both views. Thus, a subset of the detections found in the right view can be extracted by filtering out such regions that are not on the same horizontal level as the segment in the left view.

Algorithm 3: Temporal stereo superpixel anomaly detection

```
left anomaly mask ← perform Algorithm 2 on left input image
right anomaly mask ← perform Algorithm 2 on right input image
anomaly mask ← False like input image
for all left anomalous segment: left anomaly mask do
    right anomalous segments ← horizontally similar right view segments
    if possible right anomalous segments exist then
        best matching segment ← feature match the left and right segments
        left points ← extract points from left segment
        right points ← extract points from right segment
        anomaly position ← triangulate left and right points
        anomaly mask[left segment] ← True
    end if
end for
return anomaly mask, anomaly positions
```

Anomalous segments in the right view were filtered by using multiple horizontal levels. Specifically, two bands were projected from the left segment, which were each 20% of the segment's height and extended toward the center of the region from its top and bottom pixel level. Only right detections which intersected with both of these bands were evaluated as a potential match. Once a potential match was found through this method, its features were compared to the features of the left segment in the same manner as for the temporal segment matching method. If the matching error was below a set threshold, the segments were marked as matching.

To get an accurate position, corresponding points from both of the views were needed. The points were chosen by projecting 20 equiangular rays, together spanning 2π, from the segment's centroid and selecting the farthest point belonging to

the segment. This was done as to get measurements from all sides of the object, reducing the likelihood of a poor segmentation in one area ruining the positioning.

With the corresponding points chosen, the location of the anomaly could be calculated. This was done by triangulating the positions of the point pairs by the views' projection matrices. As some point pairs were likely to be poorly matched, the resulting positions were sorted by relative distance and the position corresponding to the median distance in this list was selected as the anomaly's final position.

Certain tweaks were made to the previous methods to better exploit the advantages of the stereo setup. This included reducing the small segment removal from two passes of 32px and 128px area to a single pass of 15px, corresponding to removing objects covering less than 0.0015% of the view, which allowed the detection of smaller objects. This was possible as the two algorithm instances had to agree on a detection, which reduced noise. Additionally, the naive joining of neighbouring anomalous segments was rejected and a simple system for joining neighbouring anomalous segments based on the similarity of their hue was implemented instead.

6

Results

6.1 Network capacity

Several VAE code sizes were tested. Specifically, these were 8, 64, 512 and 4096. An illustration of the reconstructive performance of the models on a set of images from the validation dataset can be seen in Figure 6.2. In the figure, the leftmost column correspond to the original images followed to the right by the reconstructions. The reconstructions are sorted by the code sizes in the ascending order of 8, 64, 512 and 4096 from the left to the right. The final training losses for these models can be seen in Figure 6.1. It can be seen that the larger code sizes have a lower training loss.

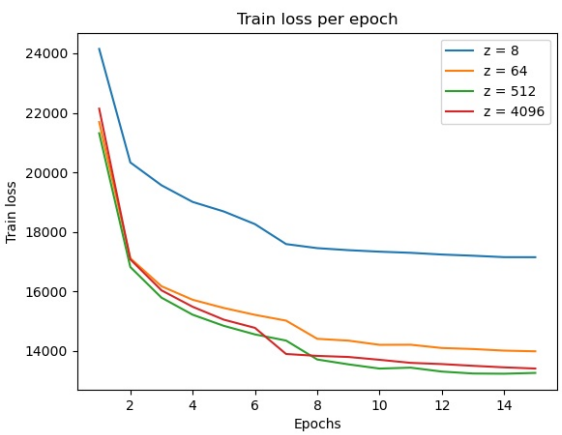


Figure 6.1: Illustration of the autoencoder's training loss. The four lines correspond to the four different code sizes used.

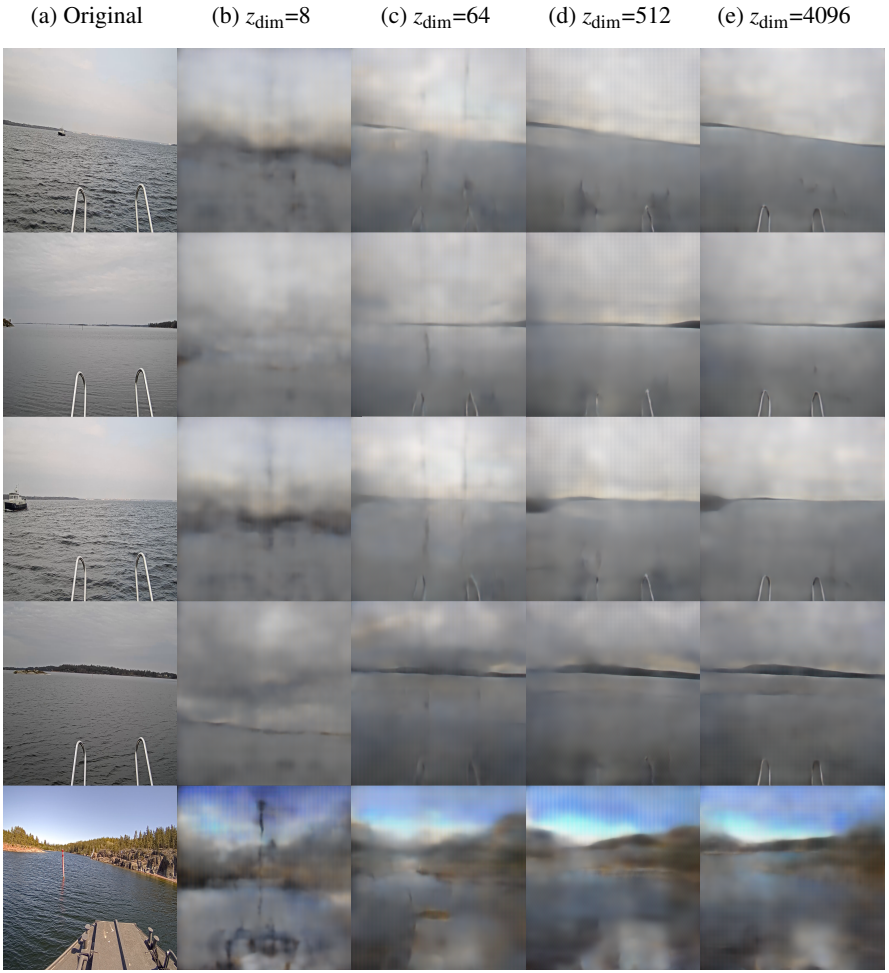


Figure 6.2: Illustration of the reconstructive performance of models with differing code dimensions on different five images from the validation dataset, handpicked to cover a broad range of scenes. The leftmost column consists of the original images followed to the right by the reconstructions produced by each model with a code size of 8, 64, 512 and 4096 respectively.

6.2 Initial approaches

The initial approaches showed results that indicated that the project was worth exploring. Heatmaps of the reconstruction error was created by having the β-VAE reconstruct an image. This is displayed in Figure 6.3, where the input image is shown

to the left, the reconstruction is shown to the right and a heatmap of the difference between the two is displayed in the middle.

The apparent loss of detail in the reconstruction leads to certain hotspots in the error heatmap. These mainly correspond to houses, tree tops extending above the tree line, a navigation mark and the two rails extending from the front of the boat.

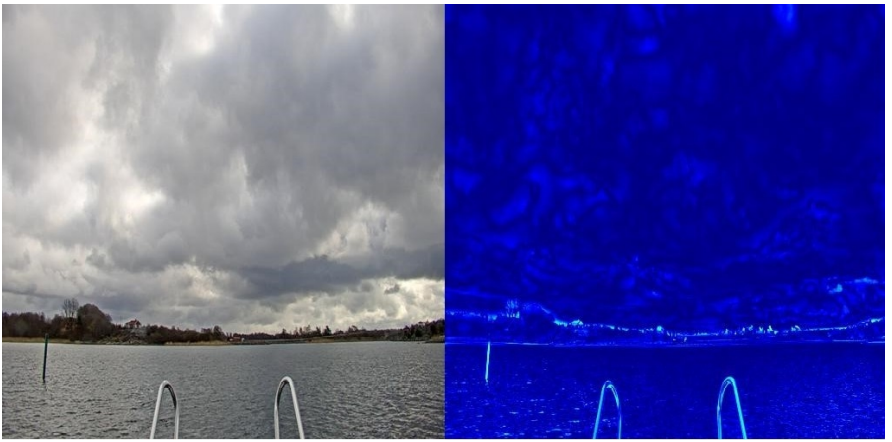


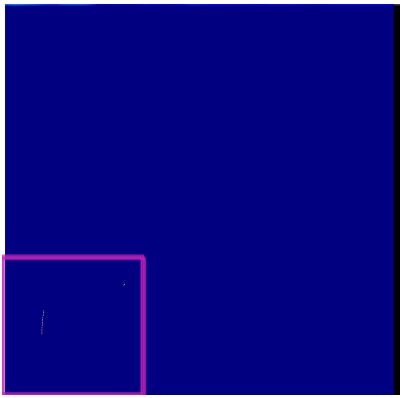
Figure 6.3: Illustration of an image sample and the heatmap created when looking at the difference between the original image and its reconstruction

To utilize the reconstruction error, the perturbation error approach was used. In this method, the pixel errors greater than a set threshold in all of the perturbations were marked as anomalous. An example of when this approach successfully classified a navigation mark as anomalous is shown in Figure 6.4, where the Figure 6.4a is the same as the leftmost image in Figure 6.3. The non-anomalous pixels are shown in blue, while the others are shown with a white color.

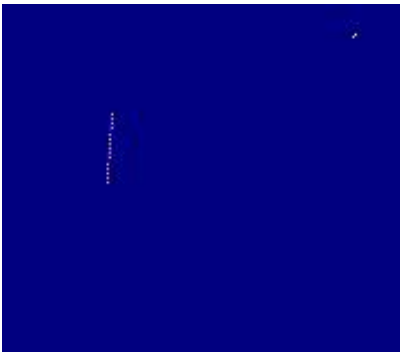
In Figure 6.4, three important features can be seen. The navigation mark is shown as a thin line of unconnected, anomalous pixels. The anomalous pixels are covering a fraction of the mark's area. To the right of the navigation mark, another small clump of anomalous pixels have been marked, corresponding to a house in the distance. As the work with the algorithms at this stage was solely experimental, no quantitative experiments were performed.

6.3 Superpixel segmentation

Superpixel segmentation clusters the pixels in an image depending on their characteristics. The segmentation was applied under the assumption that the image area covered by an object would have similar characteristics. Figure 6.5 shows a segmented image from the MODD2 dataset.



(a) Original heatmap with the area of interest marked using a bounding box



(b) Zoomed in image of the marked area in Figure 6.4a

Figure 6.4: Illustration of the results from a logical and operation performed over all perturbations. Figure 6.4a is the entire image's results where a magenta bounding box highlights the interesting area. Figure 6.4b displays the area inside the magenta bounding box from Figure 6.4a. In Figure 6.4b the navigation mark seen in the bottom left of Figure 6.3 is clearly visible.

The image illustrates positive and negative effects of using superpixel segmentation in this domain. The positives are that it is possible to find segments which encapsulates objects well, e.g., the buoy in the image. Thus, if the algorithm evaluates the errors over the segments, it is possible to find anomalies based on the magnitude of the error for a segment. However, as the segmentation algorithm has no understanding of objects, phenomena such as sun glitter, which can be seen in the bottom left of the image, will be segmented as well.

Using the superpixel segmentation method as described in Section 5.8 led to a significant amount of false positives. A test clip from the MODD2 dataset was used to gauge the influence of the different components of the algorithm on the resulting detections. Evaluating the algorithm on this clip, without the noise reduction techniques from Section 6.4 or Section 6.5, resulted in 6 true positives, 2941 false positives and 109 false negatives.

6.4 Temporal matching addition

Temporal matching addition to the algorithm was implemented to reduce the amount of noise causing false positives. The temporal matching addition to the algorithm was evaluated as described in Section 6.3, but with temporal matching enabled. Evaluating the algorithm on this clip resulted in 7 true positives, 1136

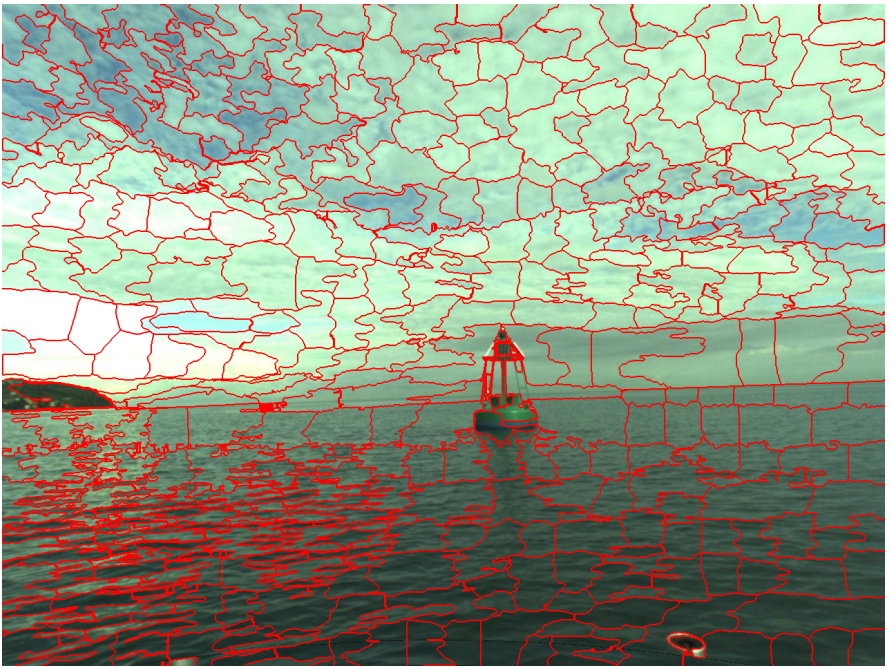


Figure 6.5: Image taken from the MODD2 dataset which has been segmented. In the image it is possible to see that the island and bouy are encapsulated in their own segments. Sun glitter in the bottom left corner is also segmented well.

false positives and 108 false negatives. Notably, this was a false positive reduction of 61% from the results in Section 6.3.

6.5 Stereo vision addition

The stereo vision addition was added to further reduce noise as well as enable anomaly positioning. The evaluation of the noise reduction was done as described in Section 6.3, but with both the temporal matching and stereo vision components enabled. Evaluating the algorithm on this clip resulted in 12 true positives, 176 false positives and 103 false negatives. Thus, this addition resulted in roughly 85% less false positives than the algorithm using the previous addition described in Section 6.4.

6.6 Evaluation of final algorithm

Multi-modal Marine Obstactle Detection Dataset 2

The algorithm was evaluated on the object detection component of the MODD2 data set. It was done by running the dataset evaluation script which is available at **https://github.com/bborja/modd**. The scores from the testing can be seen in Table 6.1.

Metric / Algorithm	*TP*	*FP*	*FN*	*F1*
BiSeNet	5014	1667	435	**82.7%**
$ISSM_{stereo}$	1828	**105**	3621	49.5%
Our (z=64)	772	2421	4677	17.9%
Our (z=512)	859	1424	4590	22.2%
Our (z=4096)	1359	3406	4090	26.6%
SegNet	**5106**	1852	**343**	82.3%

Table 6.1: Table displaying the scores of algorithms on the MODD2 data set. The best scores are marked with bold text. In the table, our algorithm is presented together with the supervised segmentation algorithms BiSeNet, $ISSM_{stereo}$ and SegNet, which were evaluated by the MODD2 dataset creators. [Yu et al., 2018; Bovcon et al., 2018a; Badrinarayanan et al., 2015]

In the same table are also the scores of three other algorithms which are top performers in the different metrics on the same task. These are supervised algorithms which, in the case of BiSeNet and SegNet, have been pretrained on the 14 million sample image classification dataset ImageNet and then had their supervised segmentation finetuned on the MaSTR1325 dataset, which contains 1325 annotated images similar to those found in the MODD2 dataset. Their scores were retrieved from the MODD2 website.

The metrics used to evaluate the performance of the algorithms are TP, number of true positives, FP, number of false positives, FN, number of false negatives and F1 which is defined in (3.21).

Depth estimation in laboratory setting

To calculate distances using stereo vision, the cameras had to be calibrated. In Figure 6.6a and Figure 6.6b, two images captured from the left and right camera and used for calibration can be seen. To measure the performance of the depth estimation, the distance to a point on the showcase seen in the images, highlighted in magenta, was measured. The measured distance was 408 centimeters, without taking into account the height difference. The algorithm evaluated the distance, including the height difference, to 421 centimeters.

(a) (b)

Figure 6.6: Two images taken from a stereo camera setup. Figure 6.6a is the image taken from the left camera and 6.6b is the image taken from the right camera. Image illustrates the process of calibrating the cameras using a checkerboard.

7

Discussion

7.1 Evaluation of our algorithm's performance

The anomaly detection algorithm described in this project was successful in some regards. During the testing, qualitative results indicated that most of what the authors regarded as anomalies were detected. This included navigation marks, other boats, buoys and some debris, which were found in the water in one recording. However, the algorithm performed worse on the newer recordings, which were slightly different in recording quality, weather and scenery to the training data. This indicates a need for the algorithm to have observed similar imagery to the setting it is to be used in for it to be efficient.

The used approach with combining an autoencoder and a segmentation algorithm has a few strengths. It is easy to train in the data gathering regard, as the input data simply needs to be void of anomalies. It is also mostly successful at detecting anomalous regions, but there are some conditions that need to be met in order to do this reliably. First, the background and the object needs to be of different enough colors to generate a high enough reconstruction error, an example of this can be seen in Figure 7.1. Second, the segmentation algorithm needs to be able to encapsulate the full object without its surroundings with one or multiple regions. Third, the background should be mostly homogenous, i.e., devoid of high frequency patterns such as checkerboards, as this has proven to the authors to be difficult for the autoencoder to reconstruct.

There are also downsides to this algorithm. It is quite hard to find a suitable network capacity, with large models showing a tendency to successfully reconstruct large anomalous objects. It is also not very robust, with the background color being important to the detection results. For example, a view of a white boat sailing into a sunset will prove difficult to detect due to the object's similarity to the background resulting in a low reconstruction error.

From the results described in Sections 6.3, 6.4 and 6.5, it is obvious that noise reduction techniques are needed. While the clip chosen to signify this was particularly hard with building reflections, glitter and objects in the water, it demonstrates how the algorithm functions without any postprocessing. The results also show that

(a) (b)

Figure 7.1: Two images from the MODD2 dataset. Figure 7.1a, depicts several boats that have sharp contrasts with the background. Figure 7.1b depicts a harbour and a buoy. The buoy has a color that blends well into the background making it difficult to see, which in turn makes it difficult to detect for the algorithm. For visualization purposes, it has been marked with magenta. It is likely that the reconstruction error for the boat will be larger than the error for the buoy since it differs more from its background.

the noise reduction additions work well. It should be noted that such additions may impair the performance of the anomaly detection system to detect *unknown unknowns* as our experiences on objects shapes how we design the noise reduction. In this case, the temporal matching mitigates detection of objects that quickly change shape, which may also stop certain legitimate anomalies with those properties from being detected.

Considerations on the training data

The algorithm was trained on roughly 200000 images. While this proved sufficient for getting decent performance on samples similar to the training data, evaluating the algorithm on recordings captured during significantly different conditions resulted in an increase in false positives. Thus, while the amount of data was likely enough, it was not varied enough to make the model generalize well to scenes that were very different to those seen in the training data. The takeaway from this is that when gauging data requirements, it is important to remember that sequential images from video recordings, such as those used in this project, contribute less to the domain subspace coverage due to their shared information. An illustration of this is shown in Figure 7.2.

The data used for training the algorithm is likely noisy. The data was manually annotated by removing what the authors classified as anomalies. While rules were set up to decide what was to be considered clean data, the classification was not always obvious. This could have been mitigated by a firmer understanding of what

Figure 7.2: Three sequential frames from the training dataset which demonstrates how much information is shared between the sequential video frames used to train the neural network.

is to be considered anomalous from the outset, which depends on having domain knowledge.

Data requirements depends on the variability of the domain. The environmental data used in this project varies with many factors, such as time of day, season of the year, sea region, water on the lens and their many interchanging variations. This makes data gathering for a general model difficult and time consuming. This is also a problem in our model, as can be seen in Figure 6.2a, where the top four are more similar to the training data than the bottom one, and also significantly better reconstructed.

Anomaly detection evaluation

Our domain-specific adaptations makes using existing anomaly detection evaluation data sets difficult. A wide variety of image data sets aimed at anomaly detection exist e.g., the pedestrian data set [Mahadevan et al., 2010]. However, in the case of this project, several domain-specific adaptations have been made to the algorithm which would have had to be removed or redesigned in order to make the algorithm useful on those data sets. Therefore, it was decided that the MODD2 data set was to be used. While it is not an anomaly detection evaluation dataset, it is a data set from the algorithm's domain of maritime obstacle detection.

The algorithm's score for the MODD2 dataset is shown in Table 6.1. While it is poor compared to the supervised algorithms' score shown in the same table,

they are not directly comparable. This is mainly due to two reasons: algorithm type and evaluation rules. Regarding the algorithm types, the three listed models are not made for anomaly detection, but rather supervised pixel classification. Supervised methods for detection tasks inherently have an advantage over their unsupervised counterparts, given that there is annotated data that covers the task at hand, as the approximated function can be more directly evaluated. In this case, the listed algorithms were trained on a dataset similar to MODD2, with pixel-level annotations for objects that were to be marked as obstacles.

The MODD2 evaluation rules are a bad fit for our algorithm. The evaluation script used by MODD2 disregards segments which crosses the horizon line. This means that only objects that are fully encapsulated by image regions containing water will be considered, which limits the evaluated objects to small ones, due to the low camera position of the recording. This poses an issue for the suggested algorithm as it depends on the segmentation algorithm being able to encapsulate the object alone within one or multiple segmented regions, which is difficult for objects that have an area of a few pixels. This issue is illustrated in Figure 7.3.

Figure 7.3: An image from the MODD2 dataset demonstrating the issue with using it as a performance indicator for obstacle detection. The image contains three evaluated objects, of which all are buoys, i.e., the two boats are not counted.

Large objects, such as other surface vehicles, are highly relevant to the obstacle collision task. Thus, to disregard these objects and focus the evaluation on very small objects, such as distant buoys, leads to a poor indication of performance on the maritime obstacle detection task this algorithm is adapted to. While this dataset

in particular is skewed toward favoring detection of certain object classes, the performance of an anomaly detection system in general is hard to measure as the evaluation data must cover a significant part of the anomaly space to provide a good measurement, and that coverage may be hard to gauge simply due to the anomalies being anomalies.

Algorithm refresh rate constraints

The time needed for inference puts constraints on the surface vehicle. The position of the segments in the image if regarded over time will vary with the velocity of the USV. With a high velocity and low algorithm refresh rate, the difference between images will be large. An illustration of how far a vehicle travels at different speeds during one update of a 1 Hz algorithm is shown in Figure 7.4.

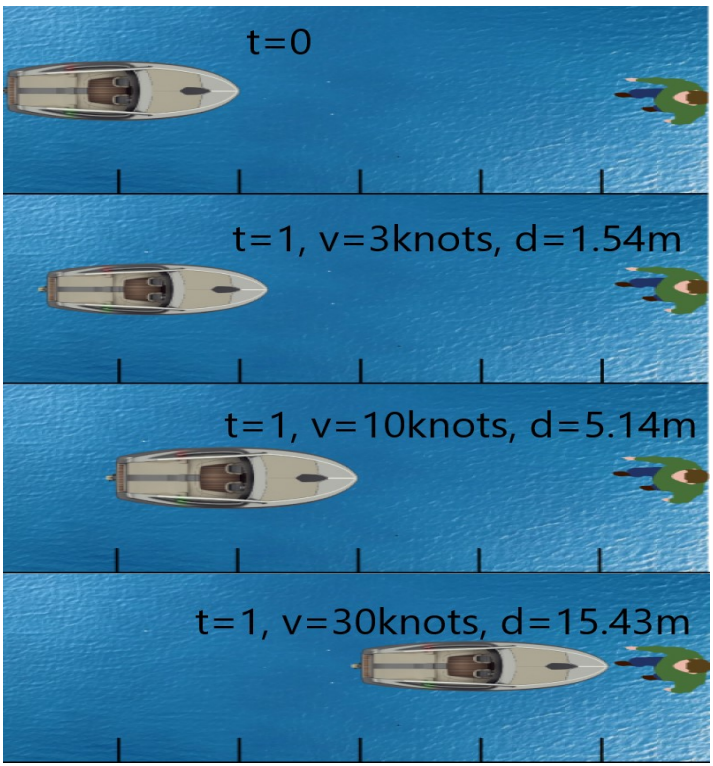


Figure 7.4: Illustration of the distance travelled by boats travelling at 3, 10 and 30 knots respectively in the time frame between predictions of our algorithm. In the images, t represents the current time step, v represents the velocity, d is the distance travelled and each tick on the bar below the images represents 5 meters.

The algorithm's refresh rate in its current state poses a problem to its usefulness. This is due to it currently processing images at a frequency of roughly 1 Hz. Because of the spatial nature of the features used to match segments, scene changes attainable at 1 Hz may cause the features of a segment to become too dissimilar between frames. Thus, unless improvements can be made to the algorithm's performance, it is necessary to limit the velocity at which a vehicle is allowed to rely on the system. During testing, roughly 70% of the computation time was used by the per region logic, which was responsible for combining the reconstructions and segmented regions. This should be highly parallellizable, implying that a higher refresh rate is possible to attain.

7.2 Algorithm components

Neural network considerations and issues

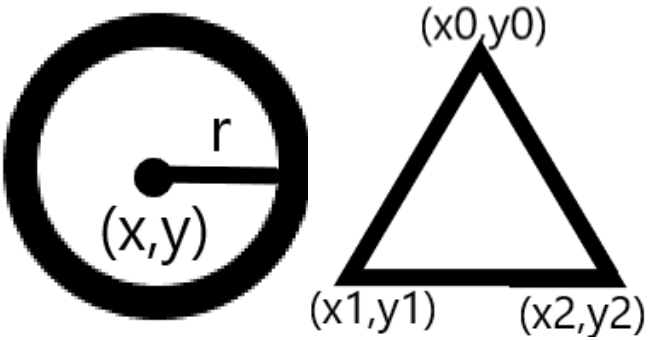
Choosing β Fine tuning a β-VAE can be difficult. The loss of a β-VAE depends on two factors that vary independently: the reconstruction loss and the Kullback-Leibler divergence.

The reconstruction loss varies with the input size. Given that the input image size is incremented, the reconstruction loss will increase. This is due to the error being calculated as the euclidean norm of the original image and its reconstruction.

The Kullback-Leibler divergence varies with the latent size of the code. An increase in the number of latent variables in the code will cause the Kullback-Leibler divergence to increase, since the difference between the distribution of each latent variable and a distribution $\mathcal{N}(0,1)$ is always greater than 0. This poses a tricky problem in choosing β as it depends on both the code size and the input size.

Network code capacity When utilizing a VAE as is done in this project, it is important to limit the reconstructive capacity. This can in part be controlled by manipulating the code size of the VAE. It impacts how much information can be used during the decoding, and as such, it is easy to think that a VAE with a larger code size is a better network. However, depending on the task, the ideal network might be one which only encodes the most important features by being forced to work with a small code size. If the network is given more variables to work with, it will find a representation for the easier parts of the data while being allowed to map harder parts more directly to the reconstruction. This means that the additional variables in the code could be used to represent previously unseen data. An example of this problem is presented in Figure 7.5.

In Figure 6.2e, reconstructions by networks with differing code sizes are illustrated. While the neural network using the smallest code size of 8 produces reconstructions that are hard to interpret, the VAE with the largest code size has a too great reconstructive ability for this task. During training the network has never seen e.g., boats and should therefore perform poorly at reconstructing them. However,



(a) (b)

Figure 7.5: Illustration of the code size required to reconstruct a circle and a triangle. The circle in Figure 7.5a, requires a parameter for x, y and r while the triangle in Figure 7.5b, needs a minimum of 6, x_0, x_1, x_2, y_0, y_1 and y_2. Given an autoencoder trained to reconstruct images of circles, by increasing the code size from 3 to 9, it would give the autoencoder 6 free parameters which are not necessary for reconstructing a circle. If an image of a triangle would be used as an input to the autoencoder, the other 6 parameters could be used to find a representation of the triangle.

in the reconstructed image containing a large boat, it is nevertheless slightly reconstructed, which would have interfered with it being detected as an anomaly.

For our algorithm, a key component is the network's inability to reconstruct objects unseen during training. Thus, it is necessary to limit the network's code capacity such that the network is poor at reconstructing the anomalies yet satisfyingly good at representing the non-anomalous components of the scene. Adjusting the z dimensions changes the sensitivity of the network, with a smaller z leading to a higher sensitivity, which thus needs to be adjusted according to the task.

Tiling Less parameters are needed when using a smaller input. When using tiling, the image would be divided up into smaller segments before being passed to the network. With the smaller input dimension, the number of parameters of the network also decreases. This is beneficial as the system is limited by the amount of available memory.

By splitting the image, the system is allowed to act on a more local stage. This makes it easier for the algorithm to reconstruct details in the image. The trade off is that cutting the image into smaller segments removes contextual information, i.e the information from neighbouring tiles. Additionally tiling becomes disastrous when anomalous objects are found in the seams between the tiles as in those cases the anomalous objects found in the original image are split.

Ultimately, tiling was not used in our algorithm. This was due to the benefits

of lower memory requirements were not needed after changes were made to the network structure and because of the loss of contextual information, which led to bad reconstructions at the seams.

Perceptual loss Perceptual loss builds on transfer learning from a pretrained network. This is beneficial as a network having been trained on big data sets of sizes not available for the own task can be utilized. For a convolutional network, this may for example allow information on how to interpret objects to flow from the pretrained network.

The network used for transfer learning also poses limitations. When it comes to images, this primarily means that the input images need to be of a specific size. This is due to the pretrained network having a specific input size. For the ImageNet dataset, the pretrained networks' input sizes are usually between 224x224 to 512x512. In our algorithm, the EfficientNet-B7 network was used, in part due to its large 600x600 input size. However, our algorithm which reconstructs images of size 1024x1024 will still have its reconstruction downsampled before a loss is calculated, resulting in a loss in information.

Superpixel segmentation considerations

Having to function in real time imposes limitations on the available superpixel segmentation methods. Several methods were tested during development. The most interesting method was the Bayesian Adaptive Superpixel Segmentation, BASS [Uziel et al., 2019]. This segmentation method did not meet the systems performance requirements as it segmented one image in approximately 30 seconds. Instead the QuickShift method described by Fulkerson was used, which could perform the segmentation process in less than a second. If a faster version of BASS was available it would be interesting to use. A qualitative comparison between BASS and QuickShift can be seen in Figure 7.6.

Using BASS would add adaptability to the system. When using the QuickShift algorithm it is necessary to determine the maximum distance that a pixel can be from its cluster centroid. As such, an object which is larger than the given maximum distance will be segmented into more than one segment. When evaluating segments over time this can be an issue as the segments which represent large objects will vary. The adaptability of BASS would allow the algorithm to determine the number of segments to split the image into, potentially resulting in a solution to the problem.

IOU was implemented to circumvent segment variability issues. As objects get larger, QuickShift will require more segments in order to encapsulate them. Since the matching method used to match segments over time depends on features extracted from the segments it causes concerns as the features will vary due to the object requiring more than one segment. The idea with IOU was to combat this behaviour as it would give the system an opportunity to label an active segment as anomalous not only if it resembles an active segment from previous time steps but also based on the segment overlap in relation to segments found in previous

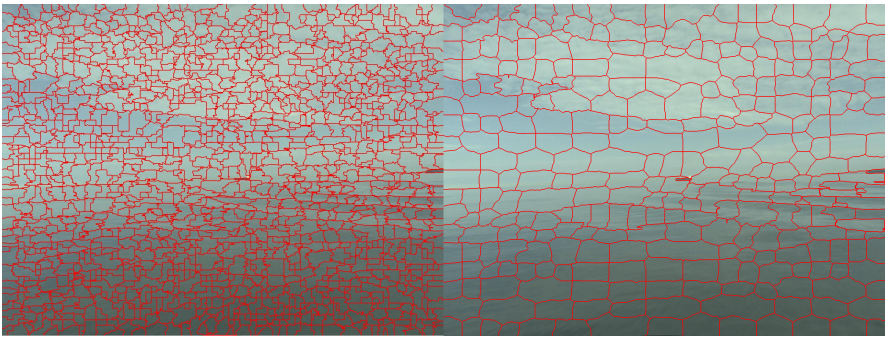


Figure 7.6: A qualitative comparison between the segmentation algorithms QuickShift and BASS as they are tasked with segmenting Figure 3.16. The left image has been segmented by QuickShift while the right image has been segmented by BASS. While the segmentation of the ship in the background is of similar quality, BASS is able to segment the image using far less segments due to the adaptability of the algorithm.

time steps. The approach was quickly discarded as it was inherently poor at moving targets.

Segment joining can accidentally join different physical objects. By joining neighbouring segments that are classified as anomalous some of the issues related to QuickShift segmenting large objects as multiple segments can be avoided. This poses a problem in cases where two or more segments of different anomalous objects are neighbouring each other. In that case, the algorithm will join all those segments together, which results in the segment matching failing due to the feature consistency requirements.

Stereo vision and temporal matching

It is necessary to find a suitable solution to the correspondence problem. Since the method used to match corresponding points is the feature matching method, there are some concerns regarding the robustness of the depth estimations. This is due to the features being hand crafted, which is also true for the temporal matching addition to the algorithm. For a production level algorithm, it would be wise to apply more sophisticated methods such as Markov random field minimization, feature extraction using a deep neural network or a classic feature extraction method such as SURF [Bay et al., 2006].

Segment joining makes distance estimation less reliable. The logic controlling what segments can be joined is non-discriminatory and will join all neighbouring anomalous segments. Thus, it is possible that the algorithm joins segments from different objects. This can have catastrophic effects on the depth estimation since the two objects rarely will be at a similar distance, causing the error of the distance

estimation to increase dramatically. In the same manner, this affects the temporal matching algorithm, as an incorrect segment joining will cause an object to change shape.

The temporal matching component of the algorithm effectively reduces the false positives, but is not without issues. One problem with this technique is how it affects the algorithm's otherwise rather pure perception of what an anomaly is, as discussed earlier. Another problem with temporal matching is its dependence on the anomaly's location within the image. When setting up this component, both worst-case refresh rates of the algorithm and maximum object movement speed across the image needs to be considered as anomalies may otherwise get discarded.

7.3 Ethics

Algorithms can have multiple areas of use. Once a product has been produced and left the hands of its creator, it is up to the users to determine how to use it. In case of the algorithm that has been presented, the authors' intentions are that the algorithm is to be used solely for the purpose of avoiding dangerous situations by locating objects in image data. However, nothing stops the algorithm from being used to find things to target instead.

An anomaly detection algorithm is discriminatory in nature. The objective of an anomaly detection algorithm is to find patterns that deviate from the norm. But determining what the norm is is not as easy as one might expect. Patterns which are completely normal but have not been represented in the training data of the algorithm will be seen as anomalous. If the same data and purpose stated in this thesis would be used then this could mean that the algorithm would become useless during winter as it would not have trained for those conditions.

In an environment filled with people, training the algorithm using biased data could cause the algorithm to behave egregiously. Imagine the algorithm applied to the problem of finding suspicious individuals at an airport. Depending on the data used to train the algorithm, the norm of what a non-suspicious person looks like will be different. If the data is not thoroughly balanced based on these differences, it could lead to the algorithm not evaluating an individual's suspiciousness based on the proper features. In a worst case scenario, this could lead to the algorithm being discriminative toward people having some feature.

It is important that the results from the algorithm is interpreted as intended. The algorithm is intended to locate anomalies using only image data. This means that an anomaly detection from the algorithm implies that there is an object at that location. On the other hand, a lack of a detection does not imply that the area is safe. The difference between the two might sound small but they carry large implications related to how the algorithm should be used. The system should be used as a component of a larger collision avoidance architecture and therefore it is important that the user understands its purpose.

8

Future work

In this thesis, an image-based anomaly detection algorithm has been covered and tested. In our tests, some strengths are seen but also weaknesses and thus, possibilities for improvements.

The options for evaluating the performance of an algorithm on this task is currently limited. While data sets such as MODD2 exist, they are not a good enough fit for the problem. Therefore, a data set should be gathered and annotated to ensure that algorithms with a similar purpose as the one presented in this paper can be assessed. During the creation of the data set, clear rules as to what constitutes anomalies need to be determined. These rules might entail conditions such as: minimum size of anomalies and what object types that are not deemed anomalous e.g small or otherwise insignificant objects.

Further research is needed to find the best segmentation method. In the current version, the usefulness of the algorithm is mostly limited by the segmentation. The processing of the segmented regions is responsible for the majority of the processing time per image and as such it is a prime candidate to research to improve the algorithm's performance.

More work is required to optimize the hyper parameters fed to the network and segmentation method. The hyper parameters have all been selected manually and are as such a result of the ability of the authors to determine what settings perform better. This is mainly due to lacking a suitable data set to evaluate against. As such it would be advisable to annotate a data set using bounding boxes to evaluate the algorithm's settings' performance in a more robust manner.

Investigating methods to reduce the algorithm's dependence on color. The color-based reconstruction error has its weaknesses, with the background and object having to be in somewhat stark contrast to each other. If this could be avoided, e.g,. by using another metric than pixel intensity for how well a region is reconstructed, the algorithm could see its usefulness extended to new areas.

In its proposed state, the algorithm matches segments over time by evaluating the euclidean norm over a set of hand crafted features. However, this method is not robust. A better approach could be to e.g., train a neural network such as an autoencoder to learn to reconstruct segments found in the images in order to extract

useful features on their e.g., shapes and contents. This could make for a more robust method of matching the segments over time as well as matching between the two cameras' perspectives.

Image-based anomaly detection has applications outside the maritime domain. Currently the most limiting factor of the algorithm is its processing time. In an environment such as on a USV where it is necessary to make split second decisions, the algorithm might not be usable. If trained on other data, the algorithm could be applied to similar problems in other domains such as airport security or parking surveillance to detect e.g., lost luggage or illegal parking.

Bibliography

Abdelhameed, I., S. Muhammed, and H. A. Ali (2014). "Automatic Quick-Shift segmentation for color images". *IJCSI International Journal of Computer Science Issues* **11**:1.

Badrinarayanan, V., A. Kendall, and R. Cipolla (2015). "SegNet: a deep convolutional encoder-decoder architecture for image segmentation." *CoRR* **abs/1511.00561**. URL: http://dblp.uni-trier.de/db/journals/corr/corr1511.html#BadrinarayananK15.

Bay, H., T. Tuytelaars, and L. Van Gool (2006). "SURF: speeded up robust features". In: Leonardis, A. et al. (Eds.). *Computer Vision – ECCV 2006*. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 404–417. ISBN: 978-3-540-33833-8.

Beauchemin, S. and J. Barron (1995). "The computation of optical flow." *ACM Computing Surveys (CSUR)* **27**, pp. 433–466. DOI: 10.1145/212094.212141.

Bengio, Y., A. Courville, and P. Vincent (2012). *Representation learning: a review and new perspectives*. eprint: arXiv:1206.5538.

Bergstra, J. and Y. Bengio (2012). "Random search for hyper-parameter optimization". *Journal of Machine Learning Research* **13**:10, pp. 281–305. URL: http://jmlr.org/papers/v13/bergstra12a.html.

Bora, D. J., A. K. Gupta, and F. A. Khan (2015). *Comparing the performance of L*A*B* and HSV color spaces with respect to color image segmentation*. eprint: arXiv:1506.01472.

Bovcon, B., R. Mandeljc, J. Perš, and M. Kristan (2018a). "Stereo obstacle detection for unmanned surface vehicles by IMU-assisted semantic segmentation". *Robotics and Autonomous Systems* **104**, pp. 1–13. ISSN: 0921-8890. DOI: 10.1016/j.robot.2018.02.017. URL: http://dx.doi.org/10.1016/j.robot.2018.02.017.

Bovcon, B., J. Muhovič, J. Perš, and M. Kristan (2018b). "Stereo obstacle detection for unmanned surface vehicles by IMU-assisted semantic segmentation". *Robotics and Autonomous Systems* **104**, pp. 1–13.

Boykov, Y., O. Veksler, and R. Zabih (1999). "Fast approximate energy minimization via graph cuts". *IEEE Transactions on Pattern Analysis and Machine Intelligence* **23**, p. 2001.

Brinkmann, R. (2008). *The Art and Science of Digital Compositing*. 2nd ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. ISBN: 0123706386.

Chebbout, S. and H. F. Merouani (2012). "Comparative study of clustering based colour image segmentation techniques". In: *2012 Eighth International Conference on Signal Image Technology and Internet Based Systems*. IEEE. DOI: 10.1109/sitis.2012.126. URL: https://doi.org/10.1109/sitis.2012.126.

Chollet, F. (2017). *Deep Learning with Python*. Manning. ISBN: 9781617294433.

Clearing the mine threat (2017). https://saabgroup.com/globalassets/corporate/media-news-press-stories/naval-press-tour-2017/presentation-mcm-underwater-systems-mcm-nov-2017_2.pdf. Accessed: 2020-09-01.

Dairi, A., F. Harrou, M. Senouci, and Y. Sun (2018). "Unsupervised obstacle detection in driving environments using deep-learning-based stereovision". *Robotics and Autonomous Systems* **100**, pp. 287–301. DOI: 10.1016/j.robot.2017.11.014. URL: https://doi.org/10.1016/j.robot.2017.11.014.

Deng, J., W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei (2009). "Imagenet: a large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee, pp. 248–255.

Elkins, L., D. Sellers, and W. R. Monach (2010). "The autonomous maritime navigation (AMN) project: field tests, autonomous and cooperative behaviors, data fusion, sensors, and vehicles". *Journal of Field Robotics* **27**:6, pp. 790–818. DOI: 10.1002/rob.20367. URL: https://doi.org/10.1002/rob.20367.

Erfani, S. M., S. Rajasegarar, S. Karunasekera, and C. Leckie (2016). "High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning". *Pattern Recognition* **58**, pp. 121–134. DOI: 10.1016/j.patcog.2016.03.028. URL: https://doi.org/10.1016/j.patcog.2016.03.028.

Eum, H., J. Bae, C. Yoon, and E. Kim (2015). "Ship detection using edge-based segmentation and histogram of oriented gradient with ship size ratio". *The International Journal of Fuzzy Logic and Intelligent Systems* **15**:4, pp. 251–259. DOI: 10.5391/ijfis.2015.15.4.251. URL: https://doi.org/10.5391/ijfis.2015.15.4.251.

FA1105 product page (2017). https://www.axis.com/products/axis-fa1105. Accessed: 2020-08-13.

FA54 product page (2016). https://www.axis.com/products/axis-fa54. Accessed: 2020-08-13.

Fulkerson, B. and S. Soatto (2010). "Really quick shift: image segmentation on a GPU". In: DOI: 10.1007/978-3-642-35740-4_27.

Golson, J. (2016). *Tesla driver killed in crash with autopilot active, NHTSA investigating*. URL: https://www.theverge.com/2016/6/30/12072408/tesla-autopilot-car-crash-death-autonomous-model-s.

Goodfellow, I. J., Y. Bengio, and A. Courville (2016). *Deep Learning*. http://www.deeplearningbook.org. MIT Press, Cambridge, MA, USA.

He, K., X. Zhang, S. Ren, and J. Sun (2016). "Deep residual learning for image recognition". *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. DOI: 10.1109/cvpr.2016.90. URL: http://dx.doi.org/10.1109/cvpr.2016.90.

Hirschmuller, H. (2008). "Stereo processing by semiglobal matching and mutual information". *IEEE Trans. Pattern Anal. Mach. Intell.* **30**:2, pp. 328–341. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2007.1166. URL: https://doi.org/10.1109/TPAMI.2007.1166.

Hu, W.-C., C.-Y. Yang, and D.-Y. Huang (2011). "Robust real-time ship detection and tracking for visual surveillance of cage aquaculture". *Journal of Visual Communication and Image Representation* **22**:6, pp. 543–556. DOI: 10.1016/j.jvcir.2011.03.009. URL: https://doi.org/10.1016/j.jvcir.2011.03.009.

Kim, H. R., D. E. Angelaki, and G. C. DeAngelis (2016). "The neural basis of depth perception from motion parallax". *Philosophical Transactions of the Royal Society B: Biological Sciences* **371**:1697, p. 20150256. DOI: 10.1098/rstb.2015.0256. URL: https://doi.org/10.1098/rstb.2015.0256.

Krizhevsky, A., I. Sutskever, and G. E. Hinton (2012). "ImageNet classification with deep convolutional neural networks". In: Pereira, F. et al. (Eds.). *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

Lim, B., T. Woo, and H. Kim (2017). "Integration of vehicle detection and distance estimation using stereo vision for real-time AEB system". In: pp. 211–216. DOI: 10.5220/0006296702110216.

Lim, H. S. M. and A. Taeihagh (2019). "Algorithmic decision-making in AVs: understanding ethical and technical concerns for smart cities". DOI: 10.3390/su11205791. eprint: arXiv:1910.13122.

Mahadevan, V., W. Li, V. Bhalodia, and N. Vasconcelos (2010). "Anomaly detection in crowded scenes". In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1975–1981.

Mathieu, E., T. Rainforth, N. Siddharth, and Y. W. Teh (2018). *Disentangling disentanglement in variational autoencoders*. arXiv: 1812.02833 [stat.ML].

Nair, V. and G. E. Hinton (2010). "Rectified linear units improve restricted boltzmann machines". In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML'10. Omnipress, Haifa, Israel, pp. 807–814. ISBN: 9781605589077.

Pihlgren, G. G., F. Sandin, and M. Liwicki (2020). *Improving image autoencoder embeddings with perceptual loss*. eprint: arXiv:2001.03444.

Redmon, J. and A. Farhadi (2017). "Yolo9000: better, faster, stronger". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ren, S., K. He, R. Girshick, and J. Sun (2015). "Faster R-CNN: towards real-time object detection with region proposal networks". In: Cortes, C. et al. (Eds.). *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc., pp. 91–99. URL: http://papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region-proposal-networks.pdf.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams (1986). "Learning representations by back-propagating errors". *Nature* **323**:6088, pp. 533–536. DOI: 10.1038/323533a0. URL: https://doi.org/10.1038/323533a0.

Szeliski, R., R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother (2008). "A comparative study of energy minimization methods for Markov random fields with smoothness-based priors". *IEEE Transactions on Pattern Analysis and Machine Intelligence* **30**:6, 1068–1080. ISSN: 01628828. DOI: 10.1109/TPAMI.2007.70844. URL: http://vision.middlebury.edu/MRF..

Tan, M. and Q. V. Le (2019). *EfficientNet: rethinking model scaling for convolutional neural networks*. cite arxiv:1905.11946Comment: Published in ICML 2019. URL: http://arxiv.org/abs/1905.11946.

United States Department of Defense (2002). URL: https://archive.defense.gov/Transcripts/Transcript.aspx?TranscriptID=2636.

Uziel, R., M. Ronen, and O. Freifeld (2019). "Bayesian adaptive superpixel segmentation". In: *The IEEE International Conference on Computer Vision (ICCV)*.

Wang, S., T. Chen, S. Chen, C. Rudolph, S. Nepal, and M. Grobler (2020). *OIAD: one-for-all image anomaly detection with disentanglement learning*. eprint: arXiv:2001.06640.

Xie, Q., M.-T. Luong, E. Hovy, and Q. V. Le (2020). "Self-training with noisy student improves imagenet classification". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10687–10698.

Yu, C., J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang (2018). *BiSeNet: bilateral segmentation network for real-time semantic segmentation*. eprint: arXiv:1808.00897.

Zhang, X., H. Wang, and W. Cheng (2017a). "Vessel detection and classification fusing radar and vision data". In: *2017 Seventh International Conference on Information Science and Technology (ICIST)*. IEEE. DOI: 10.1109/icist.2017.7926806. URL: https://doi.org/10.1109/icist.2017.7926806.

Zhang, Y., Q.-Z. Li, and F.-N. Zang (2017b). "Ship detection for visual maritime surveillance from non-stationary platforms". *Ocean Engineering* **141**, pp. 53–63. DOI: 10.1016/j.oceaneng.2017.06.022. URL: https://doi.org/10.1016/j.oceaneng.2017.06.022.

Zhang, Z. (2000). "A flexible new technique for camera calibration". *IEEE Trans. Pattern Anal. Mach. Intell.* **22**:11, pp. 1330–1334. ISSN: 0162-8828. DOI: 10.1109/34.888718. URL: https://doi.org/10.1109/34.888718.

Zimek, A. and E. Schubert (2018). "Outlier detection". In: Liu, L. et al. (Eds.). *Encyclopedia of Database Systems, Second Edition*. Springer. DOI: 10.1007/978-1-4614-8265-9_80719. URL: https://doi.org/10.1007/978-1-4614-8265-9%5C_80719.

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
MASTER'S THESIS

Date of issue
December 2020

Document Number
TFRT-6121

Author(s)
Johan Ahlqvist
André Skoog

Supervisor
Mikael Lindberg, Axis Communications, Sweden
Jens-Olof Lindh, SAAB Kockums, Sweden
Mikael Nilsson, Dept. of Mathematics, Lund University, Sweden
Johan Grönqvist, Dept. of Automatic Control, Lund University, Sweden
Anders Robertsson, Dept. of Automatic Control, Lund University, Sweden
Karl-Erik Årzén, Dept. of Automatic Control, Lund University, Sweden (examiner)

Title and subtitle
Image-based anomaly detection using β -Variational Autoencoder for surface vehicle collision avoidance

Abstract

Unmanned vehicles need robust systems to ensure the safety of the vehicle and its environment. Being able to find and avoid perilous situations is paramount to such a system. In this paper we suggest an unsupervised image-based anomaly detection algorithm using a variational autoencoder and a superpixel segmentation algorithm, which is adapted to the maritime obstacle detection task. The algorithm locates potentially hazardous objects and calculates the distance to them by measuring the images' reconstruction error over segmented regions. The algorithm's results on the public MODD2 dataset shows that it has difficulties finding small objects and that it cannot compete with the current state-of-the-art supervised segmentation algorithms on the same dataset, with an F1 score of 26.6% compared to 82.7%. Although further research and optimization is required to utilize the algorithm in a production level product, the results indicate that the algorithm is worth investigating further as it is able to detect many of the objects in our testing videos and due to it having applications in several areas.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
1-68

Recipient's notes

Security classification