Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Eirik Leikvoll
Lars Martin Hodne

Adaptation of Visual SLAM Front-Ends to Challenges of Underwater Vision

Master's thesis in Informatics
Supervisor: Rudolf Mester
Co-supervisor: Annette Stahl
July 2022

NTNU
Norwegian University of
Science and Technology

Eirik Leikvoll
Lars Martin Hodne

Adaptation of Visual SLAM Front-Ends to Challenges of Underwater Vision

Master's thesis in Informatics
Supervisor: Rudolf Mester
Co-supervisor: Annette Stahl
July 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Summary

In this report we consider marine snow, one of the numerous challenges faced by Simultaneous Localisation and Mapping (SLAM) systems in underwater environments. SLAM is used within the area of autonomy to map an agent's surroundings while locating the agent within the map.

Marine snow is a broad term denoting many kinds of floating, underwater particles, both organic and inorganic, with sizes varying from hundreds of microns up to ten centimetres. It is present throughout the open ocean at all depths and, as its name suggests, has a similar appearance to snow. Feature-based SLAM which uses sparse points from images to determine pose is prone to detect points on marine snow. This can severely impact its motion estimates, or even cause it to fail completely, because the marine snow moves independently of the agent, making any point placed on marine snow undesirable noise. Therefore, we explore the potential of Machine Learning (ML) to mitigate the motion noise from marine snow and develop two classifiers called P-CLAS and D-CLAS. P-CLAS classifies keypoints based on the small image patch which surrounds the keypoint coordinates, while D-CLAS classifies based on the keypoint descriptor which is commonly used in feature based SLAM.

To train the classifiers, we collected a set of videos and performed keypoint detection to extract the training keypoints. Each video was carefully selected such that all of its keypoints would be either exclusively 'clean' or 'marine snow'. Furthermore, we extended this data with our novel synthetic approach, in which images from the 'marine snow' videos were combined with background images free of snow through our snow extraction and superimposing pipeline. We also incorporate our methods into the SLAM framework pySLAM, to evaluate their impact and include an ablation study to quantify the effect of some of our design choices.

In our conclusion, we state that both P-CLAS and D-CLAS have similar effects on pySLAM, enabling tracking and mapping in heavy marine snow conditions where the framework would normally either fail to initialise or lose tracking shortly after initialisation. The quantitative results showed that both classifiers, despite their simplicity, are highly capable of modelling our training datasets, and also highlighted that the choice of descriptor for D-CLAS is of particular importance. However, the lack of a diverse, non-synthetic dataset with manual labels may have impacted the usefulness of the quantitative results, because the main tendencies of the qualitative results, *e.g.* P-CLAS consistently outperforming D-CLAS, were not present in real world sequences.

When qualitatively evaluating the networks' classification outputs, we observed that D-CLAS was more robust on a wider array of underwater sequences than P-CLAS. While P-CLAS would on occasion mistake certain surfaces for marine

snow or incorrectly classify marine snow on textured backgrounds, D-CLAS was less affected. Training P-CLAS on our synthetic dataset which includes more textured backgrounds alleviated some of these issues. However, both classifiers may need gains in computational efficiency for use in real-time SLAM, this is particularly the case for P-CLAS.

Oppsummering

I denne rapporten tar vi for oss marin snø, en av mange utfordringer som SLAM-systemer møter i undervannsomgivelser. SLAM brukes innenfor området autonomi for å kartlegge en agents omgivelser, samtidig som agenten lokaliseres innenfor kartet.

Marin snø er et bredt begrep som betegner mange typer flytende undervannspartikler, både organiske og uorganiske, med størrelser som varierer fra hundrevis av mikrometer opp til ti centimeter. Marin snø finnes i hele det åpne havet, på alle dyp, og som navnet antyder, har det et utseende som ligner på snø. Indirekte SLAM, som bruker nøkkelpunkter fra bilder for å bestemme posisjon og orientering, er utsatt for å bruke punkter på marin snø. Dette kan ha en alvorlig innvirkning på bevegelsesestimatene, eller til og med få systemet til å svikte fullstendig, fordi den marine snøen beveger seg uavhengig av agenten, og dermed gjør ethvert punkt plassert på marin snø til uønsket støy. Derfor utforsker vi potensialet til Maskinlæring (ML) for å motvirke bevegelsesstøyen fra marin snø, og utvikler to nevrale nett kalt P-CLAS og D-CLAS. P-CLAS klassifiserer nøkkelpunkter basert på det lille bildeområdet som omgir koordinatene til nøkkelpunktet, mens D-CLAS klassifiserer basert på den beskrivene vektoren til nøkkelpunktet som er vanlig i indirekte SLAM.

For å trene nettene, samlet vi et sett med videoer og utførte nøkkelpunktdeteksjon for å lage nøkkelpunkter til trening. Hver video ble nøye utvalgt slik at alle dens nøkkelpunkter enten var i klassen "trygg" eller "marin snø". Videre utvidet vi disse dataene med vår nye syntetiske metode, der bilder fra "marin snø" videoer ble kombinert med bakgrunnsbilder uten marin snø gjennom vår snøutvinning og bilde-kombinerings metode. Vi inkorporerer også nettverkene i SLAM-rammeverket pySLAM for å evaluere deres virkning, og inkluderer en ablasjonsstudie for å kvantifisere effekten av noen av designvalgene våre.

I vår konklusjon slår vi fast at både P-CLAS og D-CLAS har lignende effekt på pySLAM-rammeverket, og muliggjør sporing og kartlegging i krevende marine snøforhold der rammeverket normalt ikke kunne initialisere eller avsluttet sporing kort tid etter initialisering. De kvantitative resultatene viste at begge metodene, til tross for sine enkle arkitekturer, er godt egnet til å modellere treningsdatasettene våre, og fremhevet også at valget av beskrivende vektor for D-CLAS er av spesiell betydning. Imidlertid kan mangelen på et variert, ikke-syntetisk datasett med manuelt merket data ha redusert nytten av de kvantitative resultatene, ettersom hovedtendensene i de kvalitative resultatene, for eksempel at P-CLAS overgår D-CLAS, ikke var tilstede i virkelige sekvenser.

Når vi gjennomførte kvalitativ evaluaring av nettverkenes klassifiseringer, observerte vi at D-CLAS var mer robust på et bredere spekter av undervannssekvenser

enn P-CLAS. Mens P-CLAS av og til forvekslet enkelte overflater med marin snø eller feilklassifiserte marin snø fremfor teksturerte bakgrunner, var D-CLAS mindre påvirket. Å trene P-CLAS på vårt syntetiske datasett som inkluderer mer teksturerte bakgrunner, reduserte forekomsten av disse problemene. Imidlertid kan begge nettverkene ha behov for forbedret beregningseffektivitet for bruk i sanntids SLAM, dette er spesielt tilfellet for P-CLAS.

Preface

This project benefited significantly from the comments and suggestions from our supervisor Rudolf Mester, co-supervisor Annette Stahl, and Andreas Langeland Teigen.

We want to thank the co-authors of our conference paper for their help and for pushing us to pursue a publication at CVPR despite our reservations. A special thanks to Mauhing Yip and Andreas Langeland Teigen for joining us on our trip to CVPR and helping us prepare for our presentation. We are incredibly grateful to Rudolf Mester and Annette Stahl for providing the means to financially support the substantial travel costs to CVPR.

The support of family and friends throughout the master's project, and our degrees as a whole have been tremendously valued.

Table of Contents

List of Tables

List of Figures

List of Acronyms

3D Three-Dimensional.

ACWMF Adaptive Center Weighted Median Filter.
AROS Autonomous Robots for Ocean Sustainability.
AUV Autonomous Underwater Vehicle.

CNN Convolutional Neural Network.

DoG Difference of Gaussians.
DVL Doppler Velocity Log.

FCNN Fully Connected Neural Network.
FN False Negative.
FP False Positive.

GAN Generative Adversarial Network.

IMU Inertial Measurement Unit.

LRU Least Recently Used.

ML Machine Learning.

NN Neural Network.

NOAA National Oceanic and Atmospheric Administration.

OER Open Educational Resources.

ORB Oriented, Fast and Rotated Brief.

RANSAC Random Sample Consensus.

ReLU Rectified Linear Unit.

SfM Structure from Motion.

SGD Stochastic Gradient Descent.

SIFT Scale Invariant Feature Transform.

SLAM Simultaneous Localisation and Mapping.

SURF Sped Up Robust Features.

TN True Negative.

TNR True Negative Rate.

TP True Positive.

TPR True Positive Rate.

VO Visual Odometry.

VSLAM Visual SLAM.

Chapter 1

Introduction

From its origins in the 1980s, Simultaneous Localisation and Mapping (SLAM) has reached broad adoption within the area of autonomous navigation. SLAM is commonly used for navigation in robot vacuums, drones, cars, martian rovers, robotic bipeds and quadrupeds. Our work as part of the Autonomous Robots for Ocean Sustainability (AROS) research group at NTNU is to extend the reach of autonomy to the underwater domain. Our research which builds upon our 2021 pre-project is described in detail throughout this report.

1.1 Motivation

In the underwater domain[1], the drive for autonomy is especially strong because the need for a human operator can severely limit the capabilities of the underwater vehicle, either because of the tether needed to remotely operate the vehicle quite literally puts it on a leash; or the on-board pilot seat which necessitates frequent surfacings, demands a skilled and costly operator, and makes for a bulkier vehicle, unfit for tight spaces.

Unfortunately, underwater autonomy is subject to numerous difficulties which are not found to the same degree in overwater environments. For example, the image-based Visual SLAM (VSLAM) methods which have been highly successful in overwater autonomy are less effective underwater due to the turbidity of water, low-texture surroundings, lack of sunlight, dynamic illumination, marine snow, shallow water caustics, and marine wildlife ([Köser and Frese 2020]), among other things. Furthermore, GPS, which is the main mode of obtaining position in overwater scenarios is not available underwater, and water is opaque to infrared light which makes IR-based methods of estimating depth unusable.

[1] The description of the underwater domain and its challenges takes significant inspiration from Section 1.1 of our 2021 pre-project report [Hodne and Leikvoll 2021].

In this project, we focus on the issue of marine snow, which interferes with the very foundation of feature-based SLAM, that is its keypoints. Feature-based SLAM uses intermediary features from keypoint detection and matching to find correspondences between images. With accurate correspondences, it can estimate both the points' and camera's Three-Dimensional (3D) position as well as the relative pose, meaning the difference in position and orientation between the two image frames.

Due to its appearance, marine snow is frequently detected by the keypoint detectors used in SLAM. This is problematic because the 3D position of the snow is not constant between frames—the snow moves with the ocean currents. Therefore, any correspondences which include marine snow will constitute *motion noise* when the relative pose between two camera frames is estimated.

Depending on the abundance of such keypoints, the SLAM system may need additional computation time to remove keypoint correspondences which include marine snow or could fail completely if their numbers are too significant. Furthermore, removing marine snow correspondences can be more demanding than removing other forms of noise. While random mismatched correspondences have little correlation with both each other and the true motion hypothesis, marine snow correspondences are moved by the same ocean current, and therefore presents a form of motion noise which consistent with the other marine snow correspondences. Consequently, marine snow keypoints can produce a conflicting motion hypothesis which the SLAM system must ignore.

Therefore, we are motivated to develop systems which can make feature-based SLAM immune to marine snow motion noise. These systems should be compatible with existing SLAM solutions.

1.2 Aim of Study

During the thesis, we consider the following research questions:

- RQ1: How can the effect of marine snow on keypoint detection, matching, and real-time SLAM be mitigated?
  - RQ1.1: Can Machine Learning (ML) methods mitigate the impact of marine snow on underwater keypoint detection, matching, and real-time SLAM?
- RQ2: How can marine snow be digitally synthesised for use in machine learning?
  - RQ2.1: What are the characteristics of marine snow?

1.3 Contributions

We consider the meaningful contributions of this thesis to be as follows:

- We authored a peer-reviewed paper on our work. The article, titled *Detecting and Suppressing Marine Snow for Underwater Visual SLAM*, was presented at the CVPR 2022 Image Matching workshop and published to IEEE Xplore, CVF Open Access, and the CVPR conference proceedings. The paper can be found at CVF Open Access and Appendix A.
- We developed two classifiers for marine snow, P-CLAS and D-CLAS. While P-CLAS works on the image area around the detected keypoint, D-CLAS works on the binary keypoint descriptors from the ORB detector and descriptor [Rublee et al. 2011]. P-CLAS and D-CLAS are designed to run in piggy-back mode on top of any keypoint detector to limit processing to image regions that are actually candidates for use as keypoints.
- We provide a novel method to create realistic marine snow datasets. This original method extracts marine snow from images with flat backgrounds. The resulting 'marine snow dataset' is used to superimpose marine snow onto 'clean' images and video sequences. The data is publicly available[2], and as far as we know, it is the first dataset of its kind.
- We extend VAROS [Zwilgmeyer et al. 2021], an underwater pose-estimation dataset, with superimposed marine snow to create a new benchmark dataset, complete with marine snow motion noise, and VAROS' accurate ground-truths.
- We compare the performance of the descriptor-based classifier D-CLAS to the patch-based P-CLAS, and perform an ablation study to evaluate key design choices of our datasets and P-CLAS. We further compare different descriptors for use with D-CLAS, and offer qualitative results on a diverse selection of underwater videos.
- We implement both classifiers into a SLAM framework and evaluate its results on both synthetic and real-world sequences.
- We provide documented repositories of our project files and code.

1.4 Structure

In Chapter 2 Background Theory, we continue this thesis with an introduction to important underwater phenomena, and ML and SLAM concepts. Chapter 3 Literature Review presents our extensive literature review on marine snow suppression, synthesis, and modelling, as well as tangential areas such as keypoint detection and description, and rain and snow suppression. In Chapter 4 Design,

[2] https://zenodo.org/record/6424752

the design process is presented alongside the prototypes we evaluated and implemented. Chapter 5 Implementation provides the implementation details required to replicate our work. In Chapter 6 Experiments we describe our approach to evaluate the classifiers, while Chapter 7 Results presents our results and immediate analysis. Chapter 8 Discussion extends on our analysis by connecting results from the different experiments and considering what the consequences of these results are for our methods and research questions. In Chapter 9 Future Work, we give our ideas on how to extend our work and suggest some promising approaches which we did not pursue. Finally, in Chapter 10 Conclusion we summarise our work and offer our concluding remarks.

Chapter 2

Background Theory

In this chapter, basic terms and methods used throughout the thesis are presented and explained. The main topics are Machine Learning (ML), Neural Networks (NNs), keypoints, Visual Odometry (VO), Simultaneous Localisation and Mapping (SLAM), and marine conditions.

2.1 Machine Learning

There are many definitions of the term Machine Learning (ML). A frequently encountered definition is offered by Mitchell:

"*A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E.*" [Mitchell 1997]

This broad definition highlights the breadth of ML as a topic. Consequently, it is no surprise that ML is split into sub-domains, which each have sub-domains of their own and so on, *e.g.* supervised learning and its sub-domain self-supervised learning. Besides supervised learning, the most common topics in machine learning are reinforcement learning and unsupervised learning. However, in this project, we employ supervised machine learning.

2.1.1 Supervised Learning

Supervised learning deals with input-label pairs. The label describes the desired output for the particular input. The ML software is meant to model the input/output relation mathematically (e.g. with a neural network, probability distribution, or decision tree). This is achieved through a process colloquially called *training*, in

which the selected supervised ML method is provided both the input and its labels, such that the method can adapt its parameters to the data. In this case, training constitutes the experience, E, coined by Mitchel, while the task, T, is to compute the correct output for a given input. To be useful, the trained model must typically extend beyond the provided training examples, such that a previously *unseen* input still produces a correct or nearly correct output. This is important to ensure that the trained model makes trustworthy predictions when the label is unknown.

2.1.2 Datasets and Dataset Splits

Datasets are used in supervised machine learning to store labeled data samples which can be used to optimise network parameters during training, or to evaluate performance afterwards. To make sure the evaluation is fair, data used in training should not be used during evaluation. This is because the ML model may *overfit* to the data, meaning its parameters are optimised so thoroughly to the training data that it is unable to accurately model anything which deviates from what is known from training.

To make sure that a model can *generalise* to unseen data, *i.e.* it has not begun overfitting to the training samples, we divide the dataset into partitions, known as dataset splits.

The training split is the largest split of the dataset and contains all of the data used to directly optimise model parameters during training. It is the only data on which a model is trained. This data should encompass the variety of the problem space and is ideally well balanced in its representation of the possible labels, to make sure that it is representative of the data on which the model will be evaluated.

The validation dataset is used to assess progress during training and avoid overfitting. If performance on the validation data starts to decrease while the performance on the training data improves, overfitting may have begun. It is customary to save the model parameters which perform best on the validation data. Stored model parameters are often called *checkpoints* since they are made periodically during training.

The test split is used to generate unbiased metrics on data that the model has not been optimised for. Since model parameters are selected based on their performance on the validation dataset, these checkpoints will be tuned towards any biases in the validation dataset. Therefore, the actual performance of a model will be more accurately assessed on the test dataset, on which the model has not been favoured due to its superiority over other checkpoints. To adhere to this principle, the test dataset is rarely touched until the end of a project, by which time the final model(s) have been made.

2.1.3 Neural Networks

Neural Networks are a group of supervised ML methods that use linear algebra and non-linear functions to compute an output vector based on a vector input. Depending on the task T, this numerical output may correspond to a class label, a price, an action, and much more.

The fundamental building block of neural networks is the artificial neuron, which has an input vector and an output value called its *activation*. The artificial neuron can be summarised as in Equation 2.1.

$$y_i = \phi\left(\sum_j w_{ij} x_j\right) \tag{2.1}$$

In this equation, the activation y_i of a neuron, i, given its inputs, x_j, is given by the sum of the individually weighted components of the input vector, where w_{ij} represents the weight of neuron i with respect to its input j. An activation function ϕ, also called the transfer function, calculates the activation of the neuron from the sum of its weighted inputs. The activation function, is explained further in Section 2.1.3.

The artificial neuron is inspired by the neurons in biological brains [Fasel 2003], a model of which is seen in Figure 2.1. Biological neurons connect with others through dendrites which are analogous to the input vector of an artificial neuron. The dendrites channel impulses from other neurons into the soma. The signals can be either exciting or inhibiting which corresponds to the positive or negative sign of the input components of an artificial neuron. The soma effectively summarises the signals, and when their combined electrical potential exceeds a threshold, the neuron will activate along its axon, which transmits this activated signal to other neurons through its synapses [Fasel 2003].

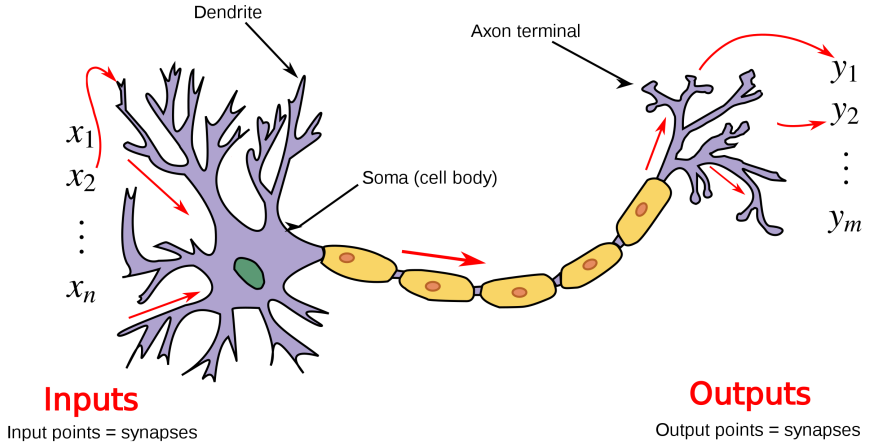


Figure 2.1: A basic sketch of the biological neuron and its parallels to the artificial neuron. Adapted from en.wikipedia.org/wiki/Artificial_neuron#/media/File:Neuron3.svg

A crucial component of both biological and artificial neurons is the ability to adjust the contribution of each dendrite/input, such that the effect of a given input on the summation is aligned with the purpose of that particular neuron. In neural networks, the weights, w_{ij}, are adjusted during training through *gradient descent* and *back-propagation* [Ruder 2016]. By using a *loss function* to quantify the error of the output given by the network, gradient descent can calculate the derivative of the loss function with respect to its inputs. Since the loss function's inputs depend on the network's weights, the gradient can be used to adjust the weights of the network by using the error between the desired activation in each layer and the actual activation. The process which computes the error is known as *back-propagation*, because it starts at the output layer, and proceeds backwards through the network, one layer at a time.

Since the gradient is highly dependent on both the specific samples used as inputs and the network's weights, the adjustment to the weights is done in small increments, which are proportional to the *learning rate* scalar. Parameters such as the learning rate are often called hyperparameters to distinguish them from the network's weights which are also called parameters. Unlike the parameters, hyperparameters are not optimised during training.

In neural networks, the neurons are grouped together in layers. We usually have an input layer, the output layer, and between them the hidden layers. The input layer does no computation and has no weights, it only represents characteristics of the input such as its dimensionality. In contrast, the hidden and output layers all have weights associated with them depending on the type of network.

From the basic building block of the artificial neuron, we can create incredibly complex and varied neural networks. In the early days of neural networks, single layer, or even single neuron networks were the main topics of research. However, multi-layer architectures eventually became the norm as back-propagation enabled the training of deep networks. Typically, Neural Networks use the output of a layer of neurons as input for the next layer. For the purposes of this thesis, we will describe two of the most common types of network architectures, the Fully Connected Neural Network (FCNN) and Convolutional Neural Network (CNN).

Fully Connected Neural Networks

An FCNN is a network where every neuron in a layer is connected with weights to every single neuron in both the preceding layer and the succeeding layer. Consequently, the total number of weights in a fully connected network is a sum of the products of the number of neurons in a layer neurons multiplied by the number of neurons in the previous layer.

The activation, a, of an FCNN layer, l, with n neurons, given an input from layer $l-1$ with m neurons and an m-dimensional vector of activations x, is

$$a = \phi\left(W^l x\right) \tag{2.2}$$

Where a is an n-dimensional output vector, W is an $n \times m$ matrix with the weights of each of the n neurons in layer l, and ϕ is the activation function. Using a matrix product is desirable because this is more efficient for modern computer hardware than calculating the weighted sum in Equation 2.1 one neuron at a time, otherwise Equations 2.1 and 2.2 are equivalent.

The main benefit of an FCNN is its lack of assumptions about the input. Simply add a large enough input layer, and an FCNN will work with any input vector. FCNNs are well suited for statistically independent inputs, where the elements of the input vector are not significantly correlated with each other. The number of layers and their number of neurons are typical design choices for FCNNs.

Convolutional Neural Networks

CNNs are a class of neural networks often used to process visual information, such as videos and images. They are based on the mathematical operation convolution, whose *kernels* are the parameters that are optimised during training. Like FCNNs, CNNs typically contain multiple layers, but instead of vector inputs and weights, CNNs can use matrices. In a CNN, each layer in the network contains a number of kernels (also called filters). When performing a convolution, the kernels are applied at multiple locations in the input which are determined by the convolution's stride, s. At each location in a convolution, the input values which belong to the current location of the kernel are multiplied with the corresponding position in the kernel. Then, all the products from this operation are summed and placed in the output, as seen in Figure 2.2.

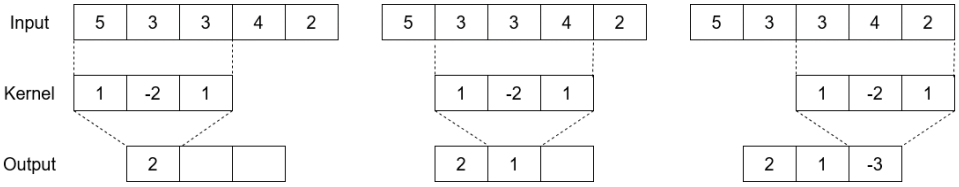


Figure 2.2: An example of a 1D convolution with no padding, stride of 1, and a kernel size of 3

Each position where the kernel is applied, matches one position in the output, meaning when the kernel is moved with stride s, we only move one position in the output. In a multi-dimensional convolution, we can have different strides in the different dimensions, denoted s_{axis}. When the kernel reaches the end of an axis, n, in the input matrix, we return the kernel to the start of the axis and move with stride s_m in the next axis, m. With images, 2D convolutions are used, while for videos 3D convolutions are often used to account for temporal information. In

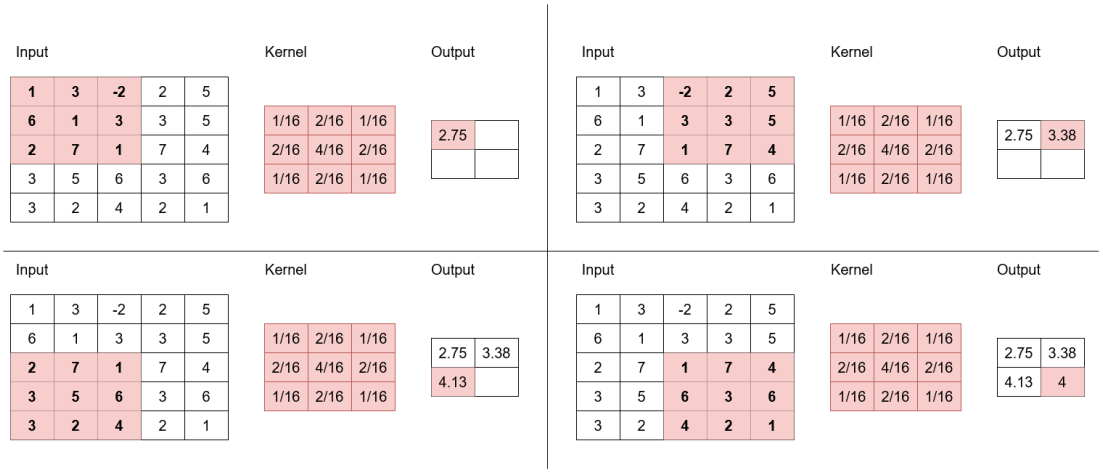


Figure 2.3: An example of a 2D convolution with no padding, a stride of 2, and a Gaussian kernel with size 3×3. The output is rounded to two decimals.

Figure 2.3, a 2D convolution is illustrated with a 5×5 input matrix, 3×3 kernel, and stride $s_x = s_y = 2$.

CNNs are considered more suited for images than FCNNs for multiple reasons. First of all, it is typically unnecessary for a single neuron to connect to every pixel in an image, which would be the case for a FCNN and lead to impractically wide layers. Second of all, CNNs have inherent weight-sharing which is not found in FCNNs. Weight sharing means CNNs do not need to learn weights for each position in the image. Instead, they rely on their moving kernels which will be applied across the image because of the stride, sharing the weights across the image. Additionally, in contrast with FCNNs which must vectorise the input image, CNNs considers the image as a matrix input and, as a consequence, maintains the spatial relations in the image. This is beneficial because a neuron can "focus" on adjacent pixels far more easily, since these will be grouped together in a matrix, unlike a vectorised image, where adjacent pixels are separated by a fixed length.

A layer, l, in a CNN consists of c_l kernels, where each kernel computes a different channel in the layer's output. Each kernel has c_{l-1} channels, where c_{l-1} is the number of channels in the previous layer. A common design feature in a CNN is that c_l increases as l increases, meaning more kernels are used to compute the output matrix. Simultaneously, the height and width of the input matrices typically decrease in a CNN as the elements of the matrix encode more and more abstract features. A high number of channels can be beneficial when a network has to learn a lot of different abstract concepts, because each filter typically recognises one kind of pattern within the image, and the number of pattern combinations increase with the depth of the network. For example, if the network has to classify 1000 different objects, each channel can activate based on the presence of some abstract feature related to the various objects, *e.g.* the contour of an animal or ve-

hicle. The reduction in height and width is sometimes required to accommodate an increased number of filters without introducing significant computational requirements.

Max-Pooling

A common method of rapidly decreasing the height and width of a CNN-layer's output is Max-Pooling. Like the kernels of a CNN, max-pooling uses sliding windows with a stride and given window size, however, unlike convolutional kernels, a 2×2 window with stride 2 is the most common configuration, and featured in the example in Figure 2.4.

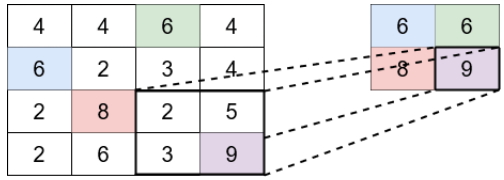


Figure 2.4: An illustration of max pooling

As its name suggests, max-pooling aggregates the maximum value at each location of the moving window in the input. The intuition behind max-pooling is that for a trained network, high values will correspond to a point of interest in the associated region of the image. Therefore, max-pooling can keep the information relating to these points of interest, while significantly reducing the dimensionality of the output. However, the fact that only the maximum value is kept can be considered a weakness, because the other values will receive a gradient of 0, and therefore have no influence on the training step associated with that particular input.

Batch Normalisation

Batch normalisation [Ioffe and Szegedy 2015] is a method to fix the mean and variance of the input at each layer of a neural network. This is done to avoid *internal covariate shift*, which is a phenomenon seen during training where the distribution (mean and variance) of a layer will change as the parameters of the preceding layers are modified with back-propagation. The change in distribution necessitates that the current layer repeatedly adapts its parameters until a more stable distribution appears, thereby prolonging training.

For a batch, B, containing m inputs with mean μ_B and variance σ_B^2, a d-dimensional sample, $x_i \in B$, has its elements, x_i^k, normalised according to Equation 2.3:

$$\hat{x}_i^k = \frac{x_i^k - \mu_B^k}{\sqrt{(\sigma_B^k)^2 + \epsilon}} \tag{2.3}$$

Where ϵ is a small constant added for numerical stability, and $k \in [1, \ldots, d]$ denotes the components of x_i. Ignoring ϵ, this ensures 0 mean and unit variance. Since many functions, *e.g.* sigmoid, cannot be modelled accurately with this mean and variance, batch normalisation includes two learnable parameters for each element of the input, γ^k and β^k, which transform the normalised input according to Equation 2.4:

$$y_i^k = \gamma^k \hat{x}_i^k + \beta^k \tag{2.4}$$

This normalisation scheme has been associated with faster convergence, improved generalisation, and more stable learning at higher learning rates. However, the mechanism with which it does so is contested [Santurkar et al. 2018].

Optimisers for Neural Networks

The specifics of how the error term from gradient descent is applied to a network's weights during back-propagation is determined by the optimiser. First of all it must be understood that theoretically, gradient descent should be performed on the entire dataset at once to obtain the correct gradient. Since doing so in practice is generally impossible, gradient descent is typically done on small batches selected from the training data. This is the idea behind Stochastic Gradient Descent (SGD) [Robbins and Monro 1951], which uses randomly selected batches of data.

Since SGD operates on a subset of the dataset, the gradients it gives will not be exactly equal to the actual gradient of the full dataset. Consequently, SGD will to some extent adjust the weights incorrectly. The use of a learning rate to control the step size when the weights are adjusted is an important way in which this issue is controlled. However, if the learning rate is too high, SGD may never converge, and if it is too low convergence will be very slow.

The Adam optimiser [Kingma and Ba 2014] uses a technique called momentum to reduce the impact of the small error in SGD's gradients. Adam maintains running averages of the gradients and their second moments. These are used in conjunction with the current gradient when updating the network's weights. First, consider SGD's update rule in Equation 2.5 for the network weights, w, to optimise a function, $L(w)$, based on a batch with n samples, and a learning rate η:

$$w := w - \frac{\eta}{n} \sum_{i=1}^{n} \nabla L(w) \tag{2.5}$$

Adam includes two more parameters, m_w^t, and v_w^t, which, at step t, are the running averages of the gradients and the second moment of the gradients, respectively. These are updated according to Equations 2.6 and 2.7, where β_1 and β_2 are forgetting factors which determine how many steps the running average covers.

$$m_w^{t+1} := \beta_1 m_w^t + (1 - \beta_1)\nabla_w L^t \tag{2.6}$$

$$v_w^{t+1} := \beta_2 v_w^t + (1 - \beta_2)(\nabla_w L^t)^2 \tag{2.7}$$

The update rule for the weights, w, is written in Equation 2.8.

$$w^{t+1} := w^t - \eta \frac{m_w^{t+1}}{\sqrt{v_w^{t+1}} + \epsilon} \tag{2.8}$$

Here, ϵ is a small constant added for numerical stability. It should be stated that the original paper presents Adam with a bias-correction step which accounts for the fact that the running averages are initialised to 0. It varies from framework to framework whether or not they use the biased or bias-corrected version. Bias-corrected Adam uses the values $\hat{m}_w$ and $\hat{v}_w$ in the update rule, instead of m_w^{t+1} and v_w^{t+1}. These are computed with the formulas in Equations 2.9 and 2.10:

$$\hat{m}_w = \frac{m_w^{t+1}}{1 - \beta_1^t} \tag{2.9}$$

$$\hat{v}_w = \frac{v_w^{t+1}}{1 - \beta_2^t} \tag{2.10}$$

Activation Functions

Activation functions are functions used on the neuron activations in a neural network to introduce non-linearity to the computation, such that the networks can be used for problems that are not linearly separable. Without such activation functions, each multi-layer network would have a single-layer equivalent which will be limited by its inability to model non-linearly separable problems like the basic XOR function. Therefore, such architectures are often avoided in favour of deeper, multi-layer networks with non-linearity offered by the activation function—ϕ in the case of Equations 2.1 and 2.2. Furthermore, the universal approximation theorem states that for any continuous function of a bounded set of variables, for any given level of accuracy, there exists a neural network that can approximate the function. This was first proven for the case of a two-layer network using the sigmoid activation function [Cybenko 1989]. The sigmoid function is written in Equation 2.11:

$$a = \frac{1}{1 + e^{-x}} \tag{2.11}$$

Of course, different activation functions have different uses, benefits, and drawbacks. For example, sigmoid is a good choice if the output is a one-dimensional probability distribution since its output is always between 0 and 1. A weakness of the sigmoid function is that its derivative can be computationally expensive

to compute compared to other alternatives. Additionally, it suffers from *vanishing gradients*, as large positive or negative numbers will give a very small gradient. This means that gradient descent will take longer to optimise the weight of a sigmoid layer because the small gradient makes the adjustment of each weight similarly small.

Another alternative is the Rectified Linear Unit (ReLU) function in Equation 2.12:

$$a = \max\big(0, x\big) \tag{2.12}$$

ReLU has a much simpler derivative, being 0 if x is negative and 1 otherwise. This can be thought of as passing the entire error of gradient descent through to the next layer, which often leads to ReLU networks learning faster than a sigmoid equivalent. However, ReLU still suffers from vanishing gradients, as any negative input value will give a gradient of 0. Other variants of the ReLU function exist to handle the problem of vanishing gradients, *e.g.* leaky ReLU whose gradient is 0.1 for negative numbers.

2.1.4 Loss Functions

So far we have not expanded on the performance measure, P, from Mitchell's definition of ML. In the case of neural networks, one ubiquitous performance measure is the loss function, L. It is ubiquitous because it is the objective that is optimised by gradient descent during the training of neural networks. There are loss functions that fulfil numerous different purposes, including ones for binary classification. Binary classification problems are problems where the network makes one of two classifications per sample, *e.g.* a positive or negative label. The binary cross-entropy loss function is a specialised case of cross-entropy loss. Cross entropy loss stems from information theory and is a way of quantifying the difference between two probability distributions. For a machine learning classifier, it can be written as in Equation 2.13:

$$\mathrm{L} = -\frac{1}{N}\sum_{i}^{N} y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i) \tag{2.13}$$

Here, N is the number of samples, y_i is the ground truth of sample i, and is typically either 0 or 1. $\hat{y}_i$ is the model's prediction for a sample i. This assumes that the output $\hat{y}_i$ is between 0 and 1 and corresponds to a probability estimate of whether the input belongs to the positive class. This allows the cross-entropy loss function to calculate the distance between the ground truth probability distribution, and the output. We also notice that the closer the predictions are to the ground truths, the closer L comes to 0. Since the function is differentiable, gradient descent will be able to incrementally adjust the weights of the neural networks to tune its output to the ground truth.

2.1.5 Binary Metrics

Binary classification metrics are another form of performance measure which are used to evaluate models for binary classification. In such tasks, the confusion matrix is arguably the basis for most metrics as it contains all of the information about a binary classification result. This includes False Positives (FPs), True Positives (TPs), False Negatives (FNs), and True Negatives (TNs).

To illustrate these concepts, consider a dataset in which all items are labelled 1, and a classifier that labels these samples as ”is 1” and ”is not 1”. Should this classifier correctly label a sample as ”is 1” this is considered a true positive since this correctly matches the ground truth label. Should it instead make an ”is not 1” classification we consider this a false negative, since the classifier has made a negative classification when it should have been positive. For a sample with label 0, the negative classification ”is not 1” will be correct and therefore a true negative. Conversely, the positive ”is 1” label would represent a false positive. We summarise this example with an overview in Table 2.1.

Classification	1	0
Ground Truth 1	TP	FN
Ground Truth 0	FP	TN

Table 2.1: An illustration of the possible binary classification designations

A confusion matrix will look very similar to Figure 2.1, but instead of the TP, FP, TN, and FN labels, it will include the number of predictions that belong in each of these quadrants.

By using the values in each quadrant of the confusion matrix, we can calculate more concise metrics which evaluate a binary classifier’s performance. Perhaps the simplest to understand is accuracy, which describes the share of predictions which were labeled correctly. Mathematically, we write:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

However, accuracy may not always be the best performance measure. For instance, if the number of samples belonging to one label is far outnumbered by the other label, or if the classifier is able to classify one label very well, but not the other, this metric may appear misleading. For example, consider a classifier which predicts whether it will snow in Paris on a given day. The classifier could get a very high accuracy by always predicting no.

The aforementioned weather forecasting classifier would do significantly worse on recall. The metric recall, also known as True Positive Rate (TPR), only focuses on samples with a positive ground-truth label, in the sense that it computes the

share of positive samples which are labelled positive by the classifier. Mathematically, this reads:

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Notice how the number of true negatives, *i.e.* no snow in Paris predictions, does not impact this measure. In a similar fashion, the True Negative Rate (TNR) represents the share of negative samples which are classified with a negative label:

$$\text{TNR} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

Since blindly labelling everything with a positive prediction will give an ideal TPR of 1, TPR is often used in conjunction with precision, which specifies how many of the samples classified as positive, are in fact positive. This has the formula:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Since precision and TPR (recall), go so well together, some metrics use a combination of them. Perhaps the most well known being the F1-score, or more generally, the F beta score which is as follows:

$$F_\beta = (1 + \beta^2)\frac{\text{precision} * \text{recall}}{\beta^2 * \text{precision} + \text{recall}}$$

In this metric, β is a parameter that weighs precision and recall. β greater than 1 values recall, while a value lower than 1 values precision. The F1-score uses $\beta = 1$, thereby giving the harmonic mean between precision and recall.

2.2 Keypoint Detection, Description, and Matching

Image keypoints and their uses were described in Section 2.2 of our pre-project report [Hodne and Leikvoll 2021]. This is amended with further discussion below.

Keypoints are coordinates in an image which locate prominent image patches, distinct enough to be matched across multiple images.

Keypoint detection and matching is one of several approaches for obtaining feature correspondences between images. Many keypoint detectors exist that look for different heuristics in the image, *e.g.* based on outliers in the Difference of Gaussians (DoG) of an image. Typically, features such as corners, edges, and blobs are targeted, but many keypoint detectors specifically use a subset of these.

One desirable trait for keypoint detectors is that their keypoints should be present regardless of image transformations, *e.g.* if we detect a keypoint in an image, we can rotate, scale and translate the image and still be able to detect the same keypoint in the transformed image. More robust keypoint detectors detect similar keypoints despite viewpoint and illumination changes. Keypoint detectors that are immune to such conditions can be said to be invariant to scale, rotation and illumination.

In order to pair the same keypoint across different image transformations, we need to match them. Therefore, keypoints are often accompanied by a descriptor that characterises the image patch at the keypoint's coordinates. Similar to the detector, the descriptors should also be invariant of changes in the image, for example, to support the matching of a feature which appears large in one image but not the other (scale invariance).

Traditionally, handcrafted descriptors have been used to match keypoints across images. However, in recent years learned descriptors have become increasingly common. Handcrafted descriptors are created by human experts, and are based on their informed opinion as to what best describes a feature. This runs the risk of there being flaws in their assumptions. On the other hand, learned feature descriptors avoid the problems of handcrafted descriptors by learning what qualities are useful in a descriptor, based on a dataset of images combined with an objective function. Learned descriptors are also able to do away with the notion of image patches and can learn what information from the image is needed when describing a point with high matchability.

When matching a keypoint from one image, we often seek to find the most similar descriptor among the keypoints in the other image. The matching step is traditionally done using nearest neighbours search on the descriptor vectors. For each descriptor in one image, the descriptor with the smallest L2-distance away in the second image is considered a match. Matching can also be done in 2D pixel coordinate space, where each keypoint is simply matched with the keypoint closest to its own pixel coordinates. This way of matching can be problematic as it does not account for large camera movements and will often give incorrect matches.

Problems can arise from nearest neighbour matching, such as multiple descriptors being so close that it seems arbitrary to consider one a match and not the others. One way to solve this could be to find mutually nearest neighbours, however, this doubles the computational effort and empirically does not greatly improve matching[Hartmann, Havlena, and Schindler 2014]. Matching speed can also be a problem, because nearest neighbours matching can be quite slow when the number of keypoints increase, as its time complexity is $\mathcal{O}(n^2)$.

2.3 Visual Odometry and VSLAM

This section is largely based on our work in Section 2.1 of our 2021 pre-project report [Hodne and Leikvoll 2021].

Visual Odometry (VO) methods are a group of computer vision algorithms which estimate the camera poses—meaning location and orientation—which relate a sequence of images. While VO only estimates pose, Simultaneous Localisation and Mapping (SLAM) incorporates mapping of the surrounding environment. Mapping is done concurrently with the localisation of the agent within the map, similar to the relative pose estimation in VO. Loop closing is a common distinction between SLAM and VO, which is used to adapt the estimated trajectory such that areas which are visited multiple times maintain the same position in each visit. This is useful because it can eliminate most of the accumulated error between the first and second visits.

The most common type of SLAM, known as feature-based SLAM, estimates pose based on displacement vectors from keypoint correspondences between images. If a sufficient number of such point-to-point correspondences are available, and if no severe outliers (meaning incorrect correspondences) are amongst them, the relative pose between the two images can be estimated. Furthermore, given the relative pose, the 3D position of the image points can be estimated using *triangulation*. Collecting these triangulated points into a 3D point cloud gives a sparse *'map'* of the environment which represents (coarsely) the geometry of the regarded scene.

Correspondence outliers, which can lead to incorrect estimates, appear from two main mechanisms. First are incorrect correspondences where keypoints which do not represent the same spatial location are matched, thus producing misleading displacement vectors. Secondly, even if keypoints are matched correctly, keypoints on dynamic objects will generate displacement vectors unfit for pose estimation because the motion of the moving object will change the displacement vector such that it longer correlates with the motion of the camera.

2.3.1 Direct and Indirect SLAM

Image correspondence estimation—the basis upon which motion estimation, as well as triangulation, is built—is divided into direct and indirect (feature-based) methods. These methods are used by Visual SLAM (VSLAM) systems which perform SLAM using only image sensors. Direct methods optimise the photometric error to obtain displacement vectors for pose estimation, meaning that transformations which can match similar-looking regions between the image pairs are generated. These methods are called direct because they use the image patches themselves to find correspondences, not intermediary features such as descriptors. On the other hand, feature-based methods perform an additional step where features are extracted from images which are then matched based on location, appearance, or both. With accurate correspondences, feature-based tracking can be done using only 5 matches if the intrinsic parameters of the camera are known ([Nister 2004]). The values in the intrinsic camera matrix are usually limited to the focal length and projective centre of the camera.

In the underwater domain, an interesting dilemma appears; since the terrain often

is low texture, it can be difficult for feature-based methods to find reliable features for localisation. However, direct methods which are better suited for low texture environments are negatively affected by the non-static lighting, which makes appearance-based matching less effective since a region may be far more shaded in one image compared to another.

2.3.2 Keypoint Rejection

In feature-based SLAM, keypoint rejection is a process intended to remove outlier keypoints prior to pose estimation, such that they bear no effect on the estimate. Traditionally, this is done using *consensus-based methods*, such as Random Sample Consensus (RANSAC) [Fischler and Bolles 1981]. Consensus-based keypoint rejection operates on matched keypoints, and bases itself on the assumption that most of the identified correspondences will be accurate and support a similar motion hypothesis, *i.e*. there is a general consensus in the data as to what the relative pose is. The outliers on the other hand will be outnumbered, and conflict with the consensus. Furthermore, in the case of outliers caused by incorrect matching, the outliers are likely to conflict with each other.

In more specific terms, consensus-based methods create subsets of hypothetical inliers to calculate motion hypotheses with high consensus among the data. When a small set of *hypothetical inliers* are selected (randomly in the case of RANSAC), a motion hypothesis is made from the subset. This hypothesis is tested on the remaining data to collect a *consensus set* of data points which, within some margin, match the hypothesis. If the consensus set is sufficiently large, a refined hypothesis may be generated on the complete consensus set. Of the sufficiently large consensus sets, the one whose motion hypothesis has the lowest loss with regards to a loss function will be considered the actual inlier set.

The main strength of consensus-based methods is their flexibility, not only do they work with any feature-based SLAM system, they are generally compatible with any estimation task in which outliers must not influence the estimate. However, as the number of outliers increases, these methods become increasingly computationally demanding. Moreover, in pose estimation, computation is wasted on matching keypoints which should be removed regardless. For example, keypoints placed on dynamic objects must be processed by the matching algorithm before the search for outliers, despite the fact that dynamic objects are bound to produce outlier displacement vectors. Furthermore, if the outliers can model a consensus set of their own, the algorithm may give the wrong output, *e.g*. if almost all keypoints are placed on a moving vehicle which covers most of the image.

Due to the aforementioned weaknesses, some methods propose rejecting the keypoints themselves, before the matching step. In such methods, descriptors, image patches or hand-crafted features belonging to a keypoint are used with a classification method to separate trusted and distrusted keypoints.

There also exists less costly heuristic approaches to reject keypoint matches. Lowe's

ratio test is perhaps the most well-known [Lowe 2004]. Based on the nearest neighbour matching algorithm, the ratio test discards a keypoint if its best match is not sufficiently better than the second-best match. The intuition behind this heuristic relies on an assumption that there will only be one good match for a keypoint, while the remaining matches are no more than noise. Consequently, if the best match is not meaningfully nearer than a noisy match, it is likely noisy as well. Therefore, the keypoint has no reliable matches and should be ignored.

2.4 Marine Conditions

Underwater odometry is subject to a number of challenges and phenomena which are not present to the same degree in other environments in which visual odometry is pursued. This section seeks to characterise these phenomena and establish their prevalence in different regions of the water column and the world.

2.4.1 Marine Snow

Marine snow is a term which entails a broad range of particles underwater, both organic and inorganic [Alldredge and Silver 1988]. Marine snow can be made organically from the ground up, or by repeated collisions and aggregation of smaller particles. These colliding particles, though mostly organic, may consist of plankton, clay, faeces, and other detritus. Their sizes can range from a few microns to many centimetres, again an indication of how broad the term is. When marine snow is discussed in this paper, it is implied that we are concerned with particles large enough to be present in recorded underwater footage used for VO and SLAM. This is more in line with another source [Lampitt 2001], which classifies marine snow as particles above 500 microns.

Marine snow's presence has been described as ubiquitous throughout the pelagic zones of the world's oceans, meaning regions away from the coast, in which wildlife can swim freely in any direction [Alldredge and Silver 1988]. For organisms in the aphotic zone, where less than 1% of the sunlight penetrates, marine snow acts as a significant source of nutrition as it sinks to greater depths, usually with a velocity of about 100–500m per day.

While marine snow is present throughout the water column, it is most heavily concentrated in the sunlit euphotic zone due to its elevated levels of marine snow production [Lampitt 2001]. Concentrations of marine snow are also known to vary seasonally, and can be especially high on the seabed by continental slopes, meaning the region *between* the deep ocean and relatively shallow continental shelf extending out from land.

The appearance of marine snow on video is determined by a complex set of factors, though the primary mechanism which renders marine snow visible is light backscattering [Boffety and Galland 2012]. In this context, backscattering is the diffuse reflection of waves of light back into the direction they came from. Since

Figure 2.5: Some of the different appearances of marine snow

Autonomous Underwater Vehicles (AUVs) tend to light up the area in front of their camera, the light will backscatter from marine snow particles back towards the camera, and thereby affect video recordings with bright white spots. The amount of backscattering will be determined by the strength of the light source and its wavelength(s), the amount of light absorbed or scattered by the water itself, as well as the qualities of the marine snow particles, like size, shape, and material makeup [Boffety and Galland 2012]. Finally, camera settings such as focal length and aperture can have a significant effect on how sharp or blurry the particles appear. In Figure 2.5, a diverse selection of marine snow samples is given to present the breadth of this term. The image patches come from the same image and are at the same scale to make size comparable across patches.

2.4.2 Other Notable Marine Conditions

The following list of challenges was compiled in Section 2.4 of the preceding master's pre-project[Hodne and Leikvoll 2021].

Underwater environments pose numerous challenges for feature extraction and pose estimation. A 2020 paper describes how wavelength-dependent attenuation and light-scattering can limit visibility to only a few meters in regions with little or no natural light [Köser and Frese 2020]. And since the only light source is often fixed to the robot itself, illumination shifts are both large and common. Due to the turbidity of water, even in well-lit regions, distant objects will seem to disappear in a haze. Additionally, marine snow and underwater wildlife can produce noisy keypoint correspondences. These attributes of underwater environments are on display in Figure 2.6. With the aforementioned challenges in mind, it may be unsurprising that some authors are unaware of any purely visual SLAM method used for live control of an underwater robot [Köser and Frese 2020].

Additionally, GPS which can find an agent's absolute position is not available underwater. And although Inertial Measurement Units (IMUs) can measure *relative* motion with high frequency, due to accumulating inaccuracies, IMU-based data should not be used on its own for SLAM [Teixeira et al. 2020]. This inevitable error accumulation is also discussed in an aforementioned paper [Köser and Frese 2020], which notes that only depth can be known absolutely using pressure sensors. RGBD cameras with infra-red depth sensors do not work underwater since

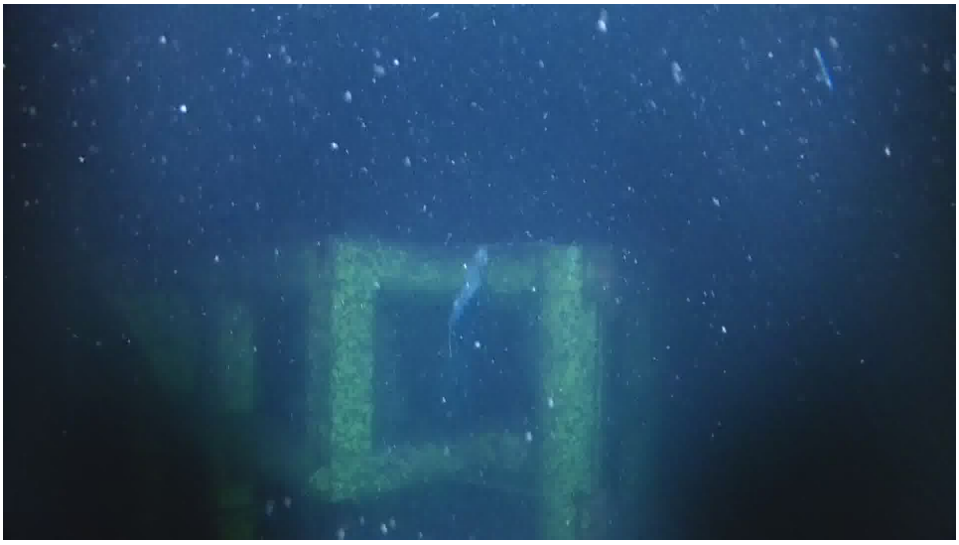


Figure 2.6: Screen capture from an underwater Eelume robot. The image displays vignetting, heavy marine snow, marine wildlife and water turbidity.

water is opaque to infrared light. However, sonar is a viable alternative for depth data. For robots that operate close to the ground, a Doppler Velocity Log (DVL) can be used to measure the Doppler shift of a signal aimed at the ground. This shift can then give an accurate measurement of ground speed. Lastly, magnetic compasses can measure some components of the earth's magnetic field to aid in localisation.

There is a lack of ground truths for training machine learning systems on SLAM tasks underwater, partly because it is so difficult to know a robot's position in such environments. While overwater scenarios can rely on GPS to generate ground truths, underwater systems must look to other methods, e.g., buoys with know positions or offline, global bundle adjustment which minimises the reprojection error between point correspondences to create more precise ground truths.

2.5 Semantic Segmentation

Semantic Segmentation is a computer vision task where each pixel in an image is assigned a label from a set of pre-existing classes. Pixels of the same class typically share something which relates them, *e.g.* that they collectively form a car within the image. However, it is important to point out that it is rarely the individual pixel value itself that decide its label; instead, accurate segmentation usually evaluates collections of pixels when applying a label, e.g., when assigning a group of pixels the car label.

Chapter 3

Literature Review

In this chapter, we present a review of the literature which can inform our work to make feature-based SLAM systems indifferent to marine snow. To this end, we present research on snow and marine snow detection; snow and marine snow synthesis for data generation; and keypoint rejection methods.

3.1 Literature Search Method

To deliver a repeatable and transparent literature search, we include our literature search method. We used Google Scholar to identify a preliminary list of candidate papers with the following search terms:

- Marine Snow
- (Rain or Snow or Marine snow) and (Removal or rejection or detection)
- (Rain or Snow or Marine snow) and (Synthesis or modelling or simulation)
- (keypoint or interest point) and (rejection or removal)
- Small object segmentation

From this preliminary list of papers, we extended our literature selection by including papers referenced in the initial batch of papers. In the upcoming literature review, we present research from this collection which passed our selection criteria:

- The methods and research processes used are described in detail or referenced
- Evaluation is based on appropriate scientific methods, and the claimed results are supported by the data

- The paper is relevant to our research on marine snow

3.2 Marine Snow Suppression

Motivated by our goal of eliminating keypoints on marine snow, we divide marine snow suppression into parallel methods and serial methods. Parallel methods are those where a majority of the computation can be done in parallel with keypoint detectors, e.g., a snow detector which segments snow while the keypoint detector runs in parallel. Serial systems are methods which must be run before keypoint detection because their output is needed as input for detection, *or* must begin after keypoint detection because the detection is needed as input (snow removal methods and keypoint classifiers, respectively).

Similarities between Snow, Marine Snow, and Rain

Rain and snow suppression are two areas of research which have benefited each other, and often appear together in articles [Zheng et al. 2013; Ding et al. 2016; Xu et al. 2012; Voronin et al. 2019]. This is despite some differences which have been highlighted by the authors of DesnowNet [Y.-F. Liu et al. 2018]. They point out that while rain typically falls in predictable paths, snowfall, particularly slow-moving, can have local discrepancies in movement, snow (and marine snow) also has a larger variation in size, shape, and velocity compared to rain, even within one image. In this respect, marine snow is arguably more similar to snowfall than rain is. Therefore, it is natural to consider methods from snow and rain suppression, and especially snow detection and removal methods since rain is less likely to match the white appearance of snow and marine snow. Consequently, we include literature from the field of snow detection, removal, and synthesis within this chapter.

3.2.1 Model-based Methods for Marine Snow Suppression

Model-based methods are characterised by handcrafted processes to detect and possibly remove marine snow. Directly addressing marine snow removal has not been a topic of research for particularly long, with all non-trivial solutions coming from the last decade [Farhadifard, Radolko, and U. v. Lukas 2017; Boguslaw Cyganek and Gongola 2018]. While some approaches existed earlier, these methods were primarily image enhancement oriented and simply modelled marine snow as a form of additive noise, either salt and pepper or Gaussian [Arnold-Bos, Malkasse, and Kervern 2005; Shanmugasundaram, Sukumaran, and Shanmugavadivu 2013; Cho and A. Kim 2017]. While these methods can be effective against very small marine snow particles, denoising methods for marine snow removal fail to address particles which occupy more than a couple of pixels, somewhat short of the largest particles which can cover a few thousand pixels if lighting conditions are correct [Boguslaw Cyganek and Gongola 2018]. On the topic of salt and pepper noise for marine snow modelling, [Boffety and Galland 2012]

argues that this model disregards properties of water such as light absorption and scattering, as well as the varying size and shape of marine snow.

We believe that the earliest method specifically made for marine snow removal is found in Banerjee *et al*. [Banerjee et al. 2014]. This claim is supported by other sources [Farhadifard, Radolko, and U. v. Lukas 2017; Boguslaw Cyganek and Gongola 2018]. Banerjee *et al*. present a basic approach which does snow removal using median filtering and implicit snow detection based on the luminance channel of a YCbCr image representation. The image is traversed with a 7×7 window, and locations which have high luminance centre and high luminance variance are selected for marine snow removal. The "high luminance" threshold is selected based on the mean and standard deviation of the local patch. Snow removal uses the median filter to replace the window-centre value with the median value of the window patch. An immediate issue of this approach is the fixed filter size, which does not model the highly diverse sizes of marine snow particles. This is addressed by Farhadifard *et al*.—the authors of one of the corroborating sources mentioned earlier [Radolko, Farhadifard, and U. F. v. Lukas 2016]. They extend the method with multi-scale filters, however, further details are not given.

A year later, Farhadifard *et al*. published a median filter approach which does account for the varying size of marine snow [Farhadifard, Radolko, and U. v. Lukas 2017]. Like the original filter-based method of [Banerjee et al. 2014], the dissimilarity of the patch-centre value to the patch-mean is used as a selection metric. To identify additional outliers within a patch, the patch is represented in RGB colour space, and an outlier detection step selects all pixels which are closer to the pixel centre than a threshold based on a weighted standard-deviation value. As final criteria, high-saturation patches are considered false detections and consequently removed due to the typically grayscale appearance of marine snow. Before the detected pixels are changed to the patch median value, a voting step is conducted to eliminate another source of false positives, namely the edges of certain objects (presumably white). The voting scheme requires that a pixel is an outlier in terms of luminance in 80% of the patches the pixel appears in. The number of patches scales quadratically with the filter size.

For multi-scale detection, the algorithm is used with filter sizes up to 19x19 for an HD image[1]. This has the significant disadvantage that the possibility for shared computation is very limited across different scales, thus impacting computational efficiency. Furthermore, the 19x19 filter size demands a great deal of computation for the voting step, where the mean and standard deviation for 361 19x19 patches must be calculated for each candidate pixel, of which there can easily be thousands. With RGB images this effort is conducted thrice.

A later paper [Boguslaw Cyganek and Gongola 2018] highlights and addresses one shared shortcoming of the aforementioned methods, namely their implicit dismissal of the temporal information present in video sequences. Allegedly, this is the first Spatio-temporal marine snow removal method. From three input

[1] It is not clear if this is HD (1280x720) or Full HD (1920x1080)

frames, the method detects and removes snow in the centre frame. The method begins with a colour saturation check using RGB distance, reminiscent of [Farhadifard, Radolko, and U. v. Lukas 2017], and once more due to the grayscale appearance of marine snow. Then, the average RGB distance of the 32x32 image patch around this centre pixel in the other two frames is determined. The centre-frame pixel is passed on to the next processing step if the RGB distance of the centre pixel in the centre frame is approximately equal to or lower than the lowest average value. To limit the detection to fast-moving particles, the next step examines the change in luminance between frames. The candidate pixel is passed to the next step if it has a sufficiently greater intensity in at least one colour channel compared to the corresponding channel in both of the other frames. This step is done with filter sizes of 1 through 7. The final detection step uses clustering to verify that the size of connected patches of candidate pixels is above a minimum area. Finally, median filtering is done similarly to previous methods, however using the median pixel intensities of the surrounding frames, instead of the centre frame.

A weakness of median-filter-based methods is their somewhat crude approach to snow removal. Since they replace the detected snow pixels with the median value, the replaced patches will be highly uniform and featureless. Consequently, any background gradients disappear and edges appear smeared. This helps motivate the use of learning-based methods which can detect and remove snow based on information gathered during the training process. Notably, Sato et al. state that they are unaware of any deep learning based marine snow removal methods [Sato, Ueda, and Tanaka 2021]. However, neural networks have been used in an intermediate marine snow detection step before filter-based removal [Koziarski and Bogusław Cyganek 2019].

When searching the literature on rain and snow removal for filter-based methods, median filtering is less prevalent. A snow removal method similar to Cyganek's method [Boguslaw Cyganek and Gongola 2018] is provided in a 1999 paper [Hase, Miyake, and Yoneda 1999], however without any snow detection and using a filter size of 1. Their method replaces pixel values with the corresponding median pixel value in an odd number of prior frames. In another paper, [H. H. Li, S. Liu, and Piao 2016], a hardware implementation of snow removal is made on an FPGA-chip using median filtering. This was motivated by its effectiveness against salt and pepper noise and computational efficiency with a 3×3 filter. Finally, [S.-C. Huang et al. 2020] use the Adaptive Center Weighted Median Filter (ACWMF) [T. Chen and Wu 2001] presented at the start of the millennium to create an initial snow-free candidate. This estimate is refined using a particle swarm optimisation method which minimizes a patch-wise error term. Notably, the ACWMF method is more advanced than the median-filtering currently used underwater and is less prone to blurring edges when filtering.

Significantly more common in rain and snow removal is the guided image filter which is considered an edge-preserving filter [He, Sun, and Tang 2013]. The guidance filter filters an image based on a guidance image, often the input image itself. This input has the effect of blurring the image, or in other words, extracting the

low-frequency information in the image which yields the high-frequency information through subtraction with the original image. In multi-guided methods, these high and low-frequency components are used in further guided filtering steps, with max, min, $+$, and $-$ operators to recover a deblurred, rain- or snow-free image. Other methods combine the guided filter with the l0 guided filter for image recovery.

The guided filter has been used in snow removal [Ramaiah et al. 2021], and rain and snow removal with the guided l0 filter [Ding et al. 2016]. A very similar guided l0 method is used for rain removal [Gautam and Raj 2018]. Multi-guided methods have also been used for rain and snow removal [Zheng et al. 2013; Xu et al. 2012], with one paper using it with anisotropic gradients to preserve edges since some smoothing of the edges still occurs [Voronin et al. 2019]. For our purposes, one shortcoming of these methods compared to the aforementioned median filtering methods is that they do not provide snow detection. The high-frequency component extracted from the image will preserve information about any snow in the image, but it can not isolate this data from other high-frequency components to form a snow detection output.

3.2.2 Learning-based Methods for Marine Snow Suppression

Learning-based methods typically follow one of two methods, either using Generative Adversarial Networks (GANs) to learn a style transfer function between snowy and snowless images [Z. Li et al. 2019; Jaw, S.-C. Huang, and Kuo 2021], or what we call composite methods which combine multiple networks and algorithms to remove the snow in multiple steps. Since it is not possible to extract snow detections from the GAN approach, we choose to focus on the less opaque composite methods.

The 2021 paper [Y. Wang et al. 2021] uses a dual-channel neural network for colour correction, dehazing, and marine snow removal. First, images are separated into their high and low-frequency components, using the guided filter [He, Sun, and Tang 2013]. Each of these components are then fed through separate channels of the neural network. A *snow removal network* operates on the high-frequency component of the input, intending to remove marine snow and other high-frequency noise in the image. An *enhancement network* operates on the low-frequency component with the intention of dehazing and colour correcting the image. Afterwards, the output of both networks are combined and fed through a *refinement network*, which generates the restored output. Unfortunately, the provided examples of restored images are so small it is difficult to assess the quality of their method.

Koziarski et. al [Koziarski and Bogusław Cyganek 2019] consider the temporal nature of marine snow by utilising a 3D CNN. Their architecture first detects snow using a combination of 3D and 2D convolutions, before using adaptive median filtering to remove the snow. The authors introduce *temporal summation*, where the 3D feature maps are summed along the original temporal dimension, as a method to transform the 3D feature maps into 2D feature maps. They suspect this

summation method is sufficient to aggregate the temporal dimension, as they only use three frames at a time as input. Their 2D and 3D convolutional layers consists of 64 3×3 and $3\times3\times3$ filters, respectively. Adaptive median filtering uses the snow detections from the CNN to remove the snow, but unlike [Boguslaw Cyganek and Gongola 2018], the temporal information is not used directly to reconstruct the information obscured by the snowflakes.

In another paper [P. Li et al. 2019], the authors perform multi-scale detection and removal of regular snow using stacked, densely connected CNN. In a densely connected CNN, the inputs of each layer are the concatenated output matrices of all previous layers, as opposed to traditional architectures in which only the output of the previous layer is used. Their system is divided into three main parts, thereby the *stacked* nomenclature. First, feature maps are calculated at three different scales using a multi-scale CNN. At each scale, three different kernels of sizes 3x3, 5x5 and 7x7 are used. Next, the feature maps are concatenated and fed through a snow detection densely connected convolutional network. The snow detection module consists of a 40-layer modified DenseNet [G. Huang et al. 2016]. Finally, to remove the snow, the output from the snow detection module is concatenated with the feature maps from the multi-scale convolution network and passed through yet another densely connected convolutional network. The snow detection and snow removal architectures are identical. For a loss function, they utilise a weighted mean square error. During training, the snow detector module is first trained independently to convergence, before both the detector and removal are trained simultaneously.

DesnowNet [Y.-F. Liu et al. 2018] incorporates two modules—the translucency module and the residual generation module—which are used to perform above-water snow removal. The residual generation module recovers areas completely covered by opaque snow, while the translucency module recovers areas that are obscured by translucent snow. Inception-v4[Szegedy et al. 2016] is used to extract multi-scale features for the modules. Instead of summing up the multi-scale features, the authors introduce a concatenation function that is learned which intends to preserve spatial information to a higher degree than summing.

3.3 Marine Snow Synthesis

Our interest in marine snow synthesis stems from a data scarcity issue and the tremendous effort required to manually label marine snow in images. By synthesising snow–and possibly other parts of the sequence—data scarcity is removed entirely and ground truths have near-perfect accuracy. However, new challenges regarding realism in synthesis must be considered. In our view, a number of qualities determine the realism of synthesised marine snow, including its motion (when applicable), translucency, sharpness, colour, shape, size, and spatial distribution.

As alluded to earlier, synthetic datasets for marine snow may be entirely synthetic where both the marine snow and surrounding environment are simulated graphi-

cally using Three-Dimensional (3D)-models and graphics software to generate an entire sequence from scratch. Alternatively, the marine snow may be superimposed onto existing images, in which a snowless image is used as a background with simulated snow added on top. In such methods, a 3D model is typically not available. With these distinctions in mind, we choose to structure this section into superimposing methods and 3D-environment-based methods. Before looking at these methods, we describe formal models of marine snow which can be of interest for both paradigms.

3.3.1 Models of Marine Snow

A marine snow model for image simulation was presented back in 2012 [Boffety and Galland 2012]. Its stated goal was to go beyond the simple salt and pepper noise model of marine snow and create a simulation suitable for the evaluation of image restoration algorithms. For simplicity, they model marine snow as a white Lambertian scatterer, meaning particles which emit white light with radiance (power per unit area) being independent of the direction the particle is being observed from. The reflectivity of the particles is determined by a Gaussian spatial profile. This effectively models the marine snow with a thick centre, which progressively gets thinner towards the edges. The work is especially rigorous in its consideration of back-scattering, in which it models both the light absorption and light scattering characteristics of water. However, the model has some weaknesses. First is the limitation posed by the spatial reflectivity profile, which only covers one highly symmetric shape of marine snow, and therefore omits many of the irregularities which commonly define the contour of marine snow. The Gaussian profile, while adequate for smaller particles, also lacks realism for larger snowflakes, which have an unnaturally wide, fuzzy border. However, the authors note that adding different spatial profiles would introduce new shapes to the simulation. Lastly, the colour of the marine snow seems inconsistent across the examples offered in the paper, with the hue seemingly being overly determined by the simulation background.

Sato et al. [Sato, Ueda, and Tanaka 2021] identified and described two different types of marine snow, the highland type and the volcanic type. The highland type appears as a solid ellipsoid in the centre encompassed by blurred edges, thereby giving a smooth transition between snow particles and the surrounding water. They describe the apparent pixel intensities as an elliptic conical frustum, *i.e.* sliced conical cones, with a rough surface. The volcanic type appears similarly, however, the solid centre has a sunken crater-like appearance. This gives a halo-like effect between the blurred edge and the solid centre. The authors describe the pixel intensities in a similar fashion, but with an overshot top edge of the frustum. One weakness of this model is that especially the volcanic type appears very blurred and out of focus, similar to the aforementioned model. Another weakness is the low amount of diversity in the snowflakes modelled. Only two types are described, both elliptical in shape. Furthermore, the paper has not been peer-reviewed. However, this is not to say that the work is of diminished value, as it is

the only existing, publicly available marine snow dataset which we know of. Additionally, the highland and volcanic marine snow, while specialised, do appear in select underwater footage in certain conditions.

3.3.2 Superimposing Methods

Superimposing methods have become common for expanding datasets with multiple types of above-water and underwater conditions like snow, fog, and rain. This research has been motivated by the limited range of conditions present in popular datasets. By using superimposing methods on existing data, a large number of degraded images can be generated with relative ease for use in training and validation. Furthermore, for a number of image enhancement tasks, the presence of the original non-degraded image offers precise ground truths for supervised learning.

Sato et al.[Sato, Ueda, and Tanaka 2021] modelled two different types of marine snow, the highland type and the volcanic crater type. In their work, they modelled the highland type by taking two congruent ellipses of the same rotation and translation, but different scales. The inner ellipse is then filled with pixels of value $c + \epsilon$, where c is a constant value that determines the transparency of the marine snow and ϵ is a small noise factor. The pixel values between the outer and inner ellipses are interpolated between $0 + \epsilon$ and $c + \epsilon$. When modelling the volcanic crater type, the pixel values are calculated in the same way as in the highland type, before a third ellipse is added inside the inner ellipse. The pixel values are calculated in the same way, before a new term g is added on top, raising the pixel values on the middle ellipsis with a smooth transition between the outer and inner ellipses. To superimpose the snow onto an image, they select random pixels in the image, where the ellipses are added with a random size and rotation. Before the snow particle is added, its location in the image is replaced by a blurred version of the original, such that the region behind the edge of the marine snow appears translucent, not transparent.

Alongside DesnowNet [Y.-F. Liu et al. 2018], the authors introduce the Snow100K dataset for (above water) snow removal and detection. Their dataset was generated using 5800 snow masks made in Photoshop. This method may be similar to others using Photoshop for snow synthesis [W.-T. Chen et al. 2021], which in turn is based on a Photoshop tutorial, but this is not clear. Each snow mask featured snow in three different sizes: small, medium and large. Within each size, trajectory, shape, density, and transparency of the particles can be randomised. For further dataset augmentation, random cropping and brightness adjustment are applied before superimposing with a weighted sum approach, whereby the mask is used as both the weight and snow. To offer varying difficulties for evaluation, easy, medium and difficult datasets are made by restricting the superimposing to either the small masks, one small and one medium mask, or one of each mask size per image.

3.3.3 3D-environment Based Methods

With 3D-environment-based methods, everything from the geometry of the environment, its illumination, snow, and physical conditions like ocean currents must be modelled and rendered in 3D-software. Using this method gives the authors full control of the sequences they synthesise, however with an additional need for 3D simulation experts, high-performance computers, and possibly expertise in physics and oceanography. Because of this, these methods have become less common than other methods and we have found almost no literature describing 3D-modelled marine snow. However, it is still worth investigating existing underwater 3D simulations which currently do not incorporate marine snow, because using an existing environment and adding snow to it is far less work than creating a 3D environment from scratch.

Hildebrandt and Kirchner [Hildebrandt and Kirchner 2010] model an underwater scene consisting of a plain seabed with no flora, fauna or human-made objects. They simulate a camera with 4 attached lights which almost homogeneously illuminate the scene. The camera moves through a 4.5 kilometer long path consisting mostly of u-turns and straight lines. Since their simulation does not model underwater lighting conditions such as turbidity and absorption, their system is only suitable for a camera pointing straight down at the seabed at a relatively close distance. They generated two datasets: one with marine snow, and the other without. Their marine snow model is only presented in short terms, though it is based on a particle emitter system for simulated 3D environments. When the simulation is rendered, 5 layers of marine snow can be included in the simulation. Each layer contains 100000 marine snow particles with varying speed, size, and random movement. Our best guess is that the 5 layers model marine snow at different distances from the camera. They state that approximately 1200 particles will be visible from each layer in any given frame of the simulation.

While only one picture example is given, the marine snow appears slightly more realistic than the model-based approaches from the previous section. However, the particles' diversity remains limited, as only small, mostly round marine snow is visible in the example. Additionally, we can not evaluate the movement of the marine snow based on the image, as this is not described in the paper. Lastly, it seems unnecessary to model 500000 particles when only 6000 remain visible, though we are unable to quantify the added cost of this.

Zwilgmeyer et al. [Zwilgmeyer et al. 2021] have created a framework for generating underwater images with physically accurate lighting and movement. Since position and geometry are known absolutely, precise ground truths are available for multiple tasks. The dataset includes relative poses, depth maps, simulated IMU data, and sequences without underwater effects. Their environment consists of a sea floor with fine-grained sand, rock, textures, and man-made objects to imitate the operating environment in the north sea. However, the dataset does not include any marine snow or other dynamic objects.

TartanAir [W. Wang et al. 2020] is a diverse dataset for VSLAM that contains an

underwater sequence. Their sequences are less focused on realism, and more focused on the diversity and difficulty of sequences. For example, the water is unrealistically clear while there is a large presence of dynamic objects like marine wildlife and floating bubbles. Similarly to Zwilgmeyer et al., TartanAir does not model marine snow, however, the bubbles that float through the water might have a similar impact.

Compared to marine snow simulation, the efforts to synthesise regular snow have been ongoing for far longer. In 2004 [Langer et al. 2004], an early method for real-time visual snow simulation was introduced. To improve real-time performance, they deliberately use a low particle count when simulating snow. To make up for this sparsity, a dynamic texture fills in the gaps based on spectral synthesis methods. This method synthesises snow based on a time-sensitive opacity function. The function gives the density of snow at the pixel coordinates x, y at time t. The function effectively adds sinusoidal noise throughout the image to mimic the appearance of small snow noise. Allegedly, This method offers more realistic results than simply increasing particle counts.

3.3.4 Hybrid Methods

We theorised that an approach combining superimposed snow with a 3D model of an existing sequence could be used to create a hybrid approach. Using 3D-reconstruction methods, a 3D model of the background sequence could be generated and then rendered with marine snow. A 2D projection of this simulated snow could then be used to superimpose snow onto the original sequence to achieve the realistic backdrop of superimposing methods and accurate depth and occlusion of 3D-model-based methods.

We later discovered this approach in use for superimposing overwater snow and fog on driving sequences [Bernuth, Volk, and Bringmann 2019]. Using either stereo images, or single images with depth, a 3D model is generated in OpenGL on a frame-by-frame basis. Using this 3D model, a physics-based simulation of snow is executed in the 3D model and then projected as a 2D snow layer onto the original sequence.

The snow particles are modelled as flat planes in light snow, and three intersecting planes during heavy snow. Snow is superimposed using motion vectors based on a wind and gravity model, alongside the car's motion vector, which is known from the dataset. On top of this, motion blur is simulated using a weighted sum of 30 interframes to model the blurring effect which occurs when cameras record moving objects. These methods lead to a highly realistic distribution of snow in the still images as if the car is driving through it. However, it is not clear if this realism remains when the images are played back as a video sequence.

Similar to [Langer et al. 2004], the authors purposefully limit the volume in which snow particles are simulated, to an area right in front of the camera. Additionally, because far-away particles become indistinguishable from each other, the simula-

tion volume is further reduced since the effects of the consolidated snow can be simulated based on their light attenuation effects. One drawback of this method is that each frame requires its own 3D model. Using Structure from Motion (SfM), all images could be combined to create one 3D model. Additionally, by matching features between frames, SfM can reduce the 3D-reconstruction error to create a more accurate reconstruction.

3.4 Keypoint Detection, Description and Rejection

This section is based on section 3.1.1 in our master's pre-project report [Hodne and Leikvoll 2021]. We include this section to highlight some of the detectors which struggle to find good keypoints in the presence of marine snow, or in the absence of defined corners. The characteristics of related descriptors are also discussed briefly.

The Harris Corner Detector [Harris, Stephens, et al. 1988] is a rotation-invariant, scale *variant*, keypoint detector based on the eigenvalues of the structure tensor. The structure tensor describes the distribution of the gradient within a specified window in an image. Corner detection is done with a five-step process. First, the image is converted to grayscale values to increase the processing speed. Next, the spatial derivative is calculated and used to make the structure tensor. Then, the smallest eigenvalues of the structure tensor are computed. Finally, non-maximum suppression is performed with a 3x3 filter to get the final points. The basic idea of this scheme is that corners will correlate with a large change in intensity if the specified window moves. This contrasts with uniform areas which will have no change in intensity and line segments which will only change intensity when moving across the edge.

This work was later extended to create the Harris-Laplace Detector [Mikolajczyk and Schmid 2001]. Their system begins by calculating a multi-scale representation for the Harris corner detector, before selecting points with a maximum Laplacian over different scales. This way their keypoint detector is robust against scale, rotation and translation.

In 2004, the Scale Invariant Feature Transform (SIFT) detector and descriptor [Lowe 2004] was introduced, which is invariant to image scaling and rotation, while remaining robust to affine distortions, noise, illumination changes, and varying viewpoints. SIFT descriptors are made to be highly distinctive which increases the likelihood of correct matches while reducing the probability of outliers. The efficiency of extracting SIFT descriptors is improved by a cascaded filtering approach which limits computationally expensive tasks to areas which are deemed salient.

SIFT is based on four main steps. First, a DoG method is used at multiple scales over all image locations to identify potential interest points. Measures of an interest point's stability determine if it is selected. Then, orientation is estimated based on the local gradient and the point is transformed relative to orientation,

scale and location, which offers invariance to these transformations. Finally, the local gradients in the keypoints' surrounding image patch are represented using a partial illumination and 3D-viewpoint invariant representation.

Later research introduced Sped Up Robust Features (SURF) [Bay et al. 2008], a novel scale and rotation invariant feature detector and descriptor. SURF takes inspiration from SIFT and has the same four main steps, but is several times faster and more robust. However, despite this increase in efficiency, both SIFT and SURF are considered unfit for real-time SLAM applications.

This issue motivated the FAST [Rosten and Drummond 2006] detector, which was an attempt at a real-time alternative to detectors like SIFT and SURF. To maintain usefulness in matching, FAST prioritises a 3D scene criterion, namely that different views of the same scene should have detections which correspond to the same 3D location. To the authors' surprise, FAST significantly outperformed previous methods on this metric by using a decision tree approach to corner detection. The tree uses pixel intensity information of a 16-pixel circle around a candidate corner p to determine the class of p.

The Oriented, Fast and Rotated Brief (ORB) feature detector and descriptor [Rublee et al. 2011], which is prominently used in the ORB-SLAM systems [Mur-Artal, Montiel, and Tardos 2015; Mur-Artal and Tardos 2017; Campos et al. 2021], aims to deliver a more efficient alternative to SIFT and SURF, just like FAST. ORB is rotation invariant, resistant to noise, and two orders of magnitude faster than SIFT, while showing similar results. ORB uses a modified FAST corner detector to produce keypoints. The detector has been augmented with an orientation operator to determine keypoints' orientation, just like SIFT and SURF do. Furthermore, a Harris corner filter is used to reject edges, and detection is done at different scales to offer multi-scale keypoints.

To generate descriptors, a modified version of BRIEF [Calonder et al. 2010] is used. BRIEF outputs a binary descriptor, based on a decision tree performing binary tests on the keypoint. Similar to SIFT, BRIEF is robust to blur, varying illumination and perspective distortions. However, in-plane rotation is a known weakness which worsens matching performance, even with only a few degrees of rotation. To amend this issue, ORB introduce steered BRIEF, which uses the patch orientation from the augmented FAST detector to rotate the binary test positions into a general frame of reference.

3.4.1 Keypoint Rejection

For snow suppression in SLAM, marine snow removal or detection effectively acts as the second time a frame is processed since the keypoint detector and descriptor have processed the frame themselves. This kind of repeated computation can significantly slow down total processing speeds. Since the keypoint detector highlights which regions are actually considered for matching, an alternative solution to marine snow detection or removal is to filter keypoints themselves, *e.g.*

based on the keypoint descriptor or the small image patch around the detected keypoint. Consequently, in this section, we look at methods using descriptors and image patches for keypoint rejection.

The conference paper [Hartmann, Havlena, and Schindler 2014] uses ML to predict the matchability of a keypoint based only on its SIFT descriptor. In their use case they mainly need to reject keypoints from foliage and other dynamic objects moving about the scene, because these are the main sources of keypoints unfit for matching. They find that using the magnitude of the DoG response from a keypoint detector is a poor way of predicting matchability. Instead, the authors explore a solution based on a random forest classifier, which trains an ensemble of decision trees to predict matchability. Their implementation has 25 decision trees, each with a maximum tree depth of 25. To classify keypoints, the random forest classifier received the SIFT descriptor associated with each keypoint. In scenes with high amounts of foliage or dynamic objects, their method significantly outperforms other methods, retaining 60% of the matches with only 30% of the keypoints.

In a later iteration of the same conference, Papadaki and Hänsch [Papadaki and Hänsch 2020] present a similar method to Hartmann [Hartmann, Havlena, and Schindler 2014]. To decrease the computational cost of keypoint matching, they remove keypoints deemed unmatchable by a random forest classifier. In contrast with Hartmann who classified keypoint descriptors, they use a handcrafted feature representation which can be rapidly generated from the SIFT keypoints. Furthermore, instead of 25 trees with a depth of 25, their classifier is a meagre 5 trees, each 5 levels deep. The handcrafted input representation is intended to go beyond just appearance-based characteristics and includes the keypoint coordinates, keypoint size, and rotation, the SIFT response, octave (scale), and number of dominant orientations (a measure of the ambiguity of the orientation), and the intensity of the green colour channel to identify vegetation. The dataset includes photographs from cities since building facades and vegetation are prone to produce keypoints on repetitive patterns which are difficult to match.

The method offered almost identical results to Hartmann [Hartmann, Havlena, and Schindler 2014] on the test data, however, Hartmann's model had slightly improved TNR, while the proposed method had the edge in recall. However, due to the compact feature representation and smaller architecture, the method improved processing speed by an order of magnitude.

Another conference paper [H. J. Kim, Dunn, and Frahm 2015], uses a Support Vector Machine (SVM) to predict the suitability of an image patch in image retrieval. To increase the discriminative power of the input, they perform classification on bundles of descriptors belonging to the same local image region. This is motivated specifically by a need to match larger image regions like a cityscape, as opposed to smaller localised features.

The paper [Leonardi, Fiori, and Stahl 2020] does keypoint rejection in underwater images to mitigate the effects of lighting artefacts and dynamic phenomena,

such as fishes and caustics. To classify keypoints, they extract their surrounding 257×257 image patch and scale them down to 65×65 pixels. Each patch is is classified by a CNN as either suitable or unsuitable for tracking. Their architecture consists of a shallow network with three convolutional layers, followed by a fully connected layer and a softmax layer. Each of their convolutional layers also does pooling and ReLU activation. Their training is supervised, and their data creation is done through manual labelling of other datasets. Their results showed strong test accuracy, at 96.7%, however, in our opinion, the manual labelling process requires an unsustainable amount of human effort to generate data. Using a custom labelling tool to speed up the labelling process, they labelled a total of 13158 keypoints from 110 images. We worry that this is not enough to generalise to real-world conditions, because a SLAM pipeline can easily use 10000 keypoints per second. It is also not clear if keypoints from one image can appear in multiple splits of the dataset. If this is possible, the results could be skewed by similarities in the keypoints in the training and evaluation splits.

3.5 Semantic Segmentation

Semantic segmentation is the task of separating an image into labelled regions of pixels. This area has been the subject of considerable research which has produced well-performing networks such as Mask R-CNN and U-net with its derivatives. However, since these networks were originally made for large objects, marine snow becomes too small for these networks to address [H.-K. Kim et al. 2019]. Furthermore, the computational demands, number of classes, and layers of abstraction present in these networks are wholly unneeded for our use case. Hence, our literature search focused on the segmentation of small objects with a small number of classes.

Numerous papers in small object segmentation highlight the difficulty of class imbalance and, consequently, the need for appropriate loss functions. Small object segmentation touches on a number of domains, including traffic light segmentation [H.-K. Kim et al. 2019], satellite image segmentation [Segl and Kaufmann 2001; Kampffmeyer, Salberg, and Jenssen 2016], and bird segmentation in windmill parks [Takeki et al. 2016]. Deep CNNs are heavily used among these papers, though unsupervised image segmentation techniques are employed in the 2001 paper [Segl and Kaufmann 2001].

Chapter 4

Design

In this chapter, we present our design decisions and their motivation. Briefly put, the goal of all our designs is to enable SLAM front-ends which are immune to marine snow motion noise.

4.1 Architectures

In the literature review, we divided marine snow suppression architectures into serial and parallel designs. Serial architectures are systems which must run entirely before or after keypoint detection and/or description, while parallel architectures can largely run simultaneously with detection and description. In this section, we present our choice of architecture for marine snow suppression and the various design choices which were made when developing these systems.

4.1.1 Chosen Approach for Marine Snow Suppression

A clear divide exists within the serial methods which must be highlighted, namely whether the method should be used before or after keypoint detection. Before keypoint detection, one could use marine snow removal methods. these methods employ image enhancement techniques to eliminate marine snow (and possibly other unwanted degradations) from the image. Using such methods before keypoint detection will ideally eliminate all marine snow particles in an image that the keypoint detector may erroneously extract, hence sanitising the keypoints.

Alternatively, a serial approach may use keypoint rejection to remove marine snow keypoints. Keypoint rejection methods use information about the extracted keypoints, like their prominent image patch or descriptor, to filter away keypoints that are deemed unreliable. This is illustrated in Figure 4.1.

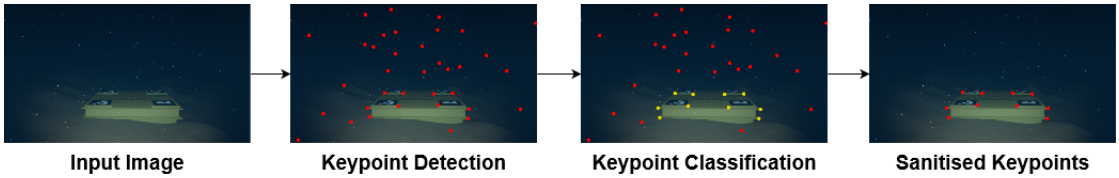


Figure 4.1: A serial pipeline, running keypoint detection first and then keypoint classification either on the descriptors or image patches around the keypoints.

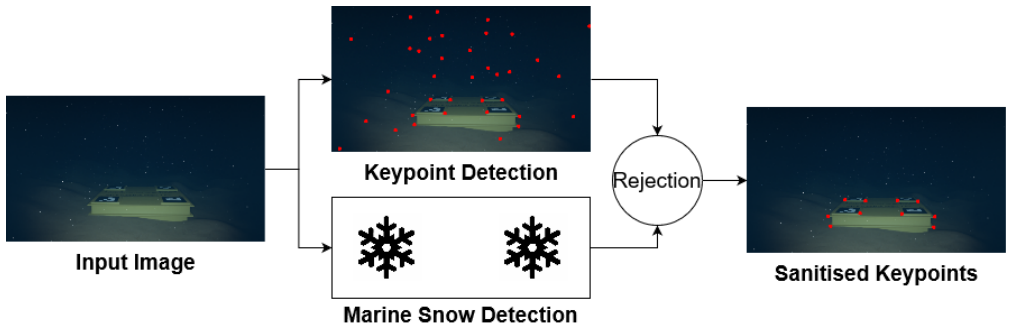


Figure 4.2: A parallel pipeline, running keypoint detection at the same time as snow detection. Some computation has to be done after keypoint detection and snow detection to filter the keypoints.

For parallel snow suppression, we propose a semantic segmentation approach which classifies each individual pixel as snow or not snow. This segmentation mask can be used to reject keypoints placed on snow. This approach is illustrated in Figure 4.2.

We believe both snow removal, segmentation, and keypoint classification are viable methods for snow suppression. However, we argue that keypoint classification using descriptors or image-patches is the most promising choice. Snow removal (and segmentation) performs work on the entire image, irrespective of the distribution of keypoints, meaning computation is wasted in regions where no keypoints are present. We also worry that a snow removal step can introduce artifacts which disrupt the keypoint extraction, description, and matching steps inherent in SLAM. It is also not clear to us how snow removal is preferable to snow segmentation on the full image, since segmentation poses no risk of introducing artifacts and can be run in parallel with keypoint detection, therefore reducing cycle time to some extent.

For the segmentation approach, computation is also wasted in regions where there are no marine snow keypoints. And although it is desireable to parallelise the code as can be done with segmentation, the effect of this is difficult to quantify. Furthermore, to run in parallel necessitates a vacant CPU thread, which with most

modern segmentation networks must be paired with a GPU.

The main benefit of keypoint classification is that it limits computation to locations which are actually candidates for keypoint based pose estimation. Their inputs should also be significantly lower in dimensionality, which may help reduce the complexity and computational demands of the final designs. One weakness compared to the snow removal approach, is that for some environments, keypoint rejection may remove so many keypoints that it is not possible to accurately perform SLAM without at least one more round of keypoint detection.

We have already presented our reservations about the snow removal approach, which we fear will introduce artifacts that disturb feature based SLAM. Furthermore, because the main benefit of the segmentation design is difficult to quantify, and because this method can pose more stringent requirements on hardware, we are left with the serial keypoint classifier scheme which we believe is the most computationally efficient among the three approaches. Keypoint classification did have one weakness compared to snow removal, regarding heavy marine snow conditions where most keypoints may be eliminated. However, since this issue should not apply under most operating conditions, we believe keypoint classification remains as the best approach.

We consider both keypoint classifiers based on descriptors and classifiers based on image patches to be viable candidates in our chosen approach. Hence, we present two designs, one for descriptor classification, the other for patch classification. The motivation behind each scheme is detailed further in the designated sections of each design.

4.1.2 Descriptor-based Classifier

Our reasoning to use descriptors for classification is based on the ability to reuse information extracted during the keypoint description process. While descriptor classification does necessitate a platform dependent implementation, meaning trained models only work with one keypoint descriptor, there are multiple benefits which may outweigh this negative. Descriptors are generally made with characteristics such as invariance to rotation, noise, scale, and illumination changes, all of which seem desirable for classifying marine snow. Therefore, training on descriptors not only entails a significant dimensionality reduction and a subsequent computational efficiency gain, it provides a representation with numerous qualities which the network might otherwise have to learn by itself, *e.g.* scale invariance.

To classify descriptors we considered multiple machine learning methods, for example the random forest approach we encountered in the literature review [Hartmann, Havlena, and Schindler 2014; Papadaki and Hänsch 2020]. Our first steps towards descriptor classification was to evaluate our own FCNN classifier, and scikit's Random Forest Classifier, Support Vector Classifier, K-Nearest Neighbours, Naive Bayes, and Quadratic Discriminant Analysis classifier. We evaluated

each design on a dataset consisting of ORB descriptors. The descriptors had been generated on video sequences in which we knew the label of all keypoints beforehand. Specifically, we used video sequences entirely without marine snow to generate samples in the 'clean' class, and video sequences whose only features were marine snow to generate samples for the 'marine snow' class.

The FCNN implementation surpassed the other methods with relative ease (see Table 7.3), and was consequently positioned as the first descriptor-based classifier, which we named D-CLAS. As our first descriptor-classifier, D-CLAS has a rather conventional design with the ReLU activation function in the hidden layers and the sigmoid function in the output layer. The final sigmoid function ensures that the output is in the range $[0, 1]$ which can be interpreted as a pseudo probability of the input being in the positive 'marine snow' class. The only other design consideration for this initial architecture was to gradually decrease the layer sizes to one neuron in the final layer. The architecture is described in full in Table 5.1.

4.1.3 Patch-based Classifier

The image-patch based classifier—P-CLAS—is motivated by a desire for a keypoint detector and descriptor-independent system, *i.e.*, a system which will work with all keypoint based SLAM pipelines without any additional training. Furthermore, if we consider a patch classifier architecture using a feature extractor + FCNN classifier, we can immediately draw similarities to the descriptor-based classifier. If we think of the flattened output of the feature extractor as a snow-focused version of D-CLAS's descriptors, it seems natural to hypothesise that these features may be more suitable for snow classification than keypoint descriptors, since the feature extractor is specifically optimised for snow detection. In our opinion, this hypothesis is not particularly far-fetched, hence we were motivated to develop this idea further.

In keeping with above example, we modelled our initial architecture following the feature extractor and snow-feature classifier concept. In the first design of P-CLAS, feature extraction was done using a conventional CNN style network with max-pooling and batch normalisation, followed by a FCNN head with sigmoid output for classification. This choice of architecture was made, simply to follow the designs of basic image classification networks.

Patch-extraction was another important topic, since we made the decision of only including images and their keypoint coordinates in the patch-dataset. This was deliberately done to make the patch extraction method and attributes like the size and shape of the patches an integral part of the architecture. Doing otherwise would limit the possibilities of modifying the patches later on, something we wanted to avoid since we believed the input patches could have a significant impact on P-CLAS' performance.

Based on our findings in the literature search on the characteristics of marine snow and marine snow detection methods, we wanted to extract patches at multiple

scales to account for the diversity of marine snow sizes. Consequently, our design includes a patch-extraction method which, given a keypoint, image, and a list of patch scales, can extract patches at each scale. CNNs require that the input matrices are of the same dimensions, hence we use bi-linear interpolation to resize the patches to the same height and width.

Since keypoints can be near the edge of the image where large patches may go outside the bounds of the image, we zero-pad all images after loading them from disk with the padding size equivalent to half of the maximum patch size. We later tested with median-padding in an attempt to make patches on the border more consistent with patches elsewhere in the image, and found that P-CLAS's accuracy on keypoints which included some amount of padding increased from 0.825 to 0.843, mostly due to a decrease in the false negative rate from 0.266 to 0.228. However, as it turns out, the number of keypoints which were actually affected by the padding rule was only 1266, or less than 0.1% of the dataset. Hence, to us, the cost of calculating the median did not seem to be worth the small difference, and our designs only make use of zero-padding.

4.1.4 Keypoint Detection Masks

During the experimental stages of our work, we developed methods to generate keypoint detection masks using the classification results from our classifiers. Keypoint detection masks tell the keypoint detector where it should and should not look for keypoints in an image. By generating our own masks based on the classification results from P-CLAS or D-CLAS, we would be able to direct the keypoint detector to areas with good keypoints, or make the keypoint detector avoid areas with marine snow, or do both simultaneously. Our initial design was a rudimentary one, in which we weighted areas around each keypoint detected in an image. Keypoints labelled clean would increase the weight of their nearby area, while keypoints labelled marine snow decreased the weight. We would then apply a threshold to the weights create a keypoint detection for a second round of keypoint detection, *e.g.* to make up for the reduction of keypoints caused by the removal of snow keypoints. However, we quickly noticed that our approach did not generalise well across sequences. We believe a more complex heuristic is needed to create useful masks, meaning something which considers the distribution of keypoints, and does not just weigh each keypoint individually. We eventually chose to focus on other uses of the classifiers, consequently, the methods for keypoint detection masks remain in the design stage.

4.2 Datasets

We have made four datasets for classification of keypoint patches and descriptors. The first dataset extracted keypoints from images with either exclusively marine snow or no snow at all. By collecting videos with these characteristics, we can label keypoints in bulk based on which category the video belonged to. The

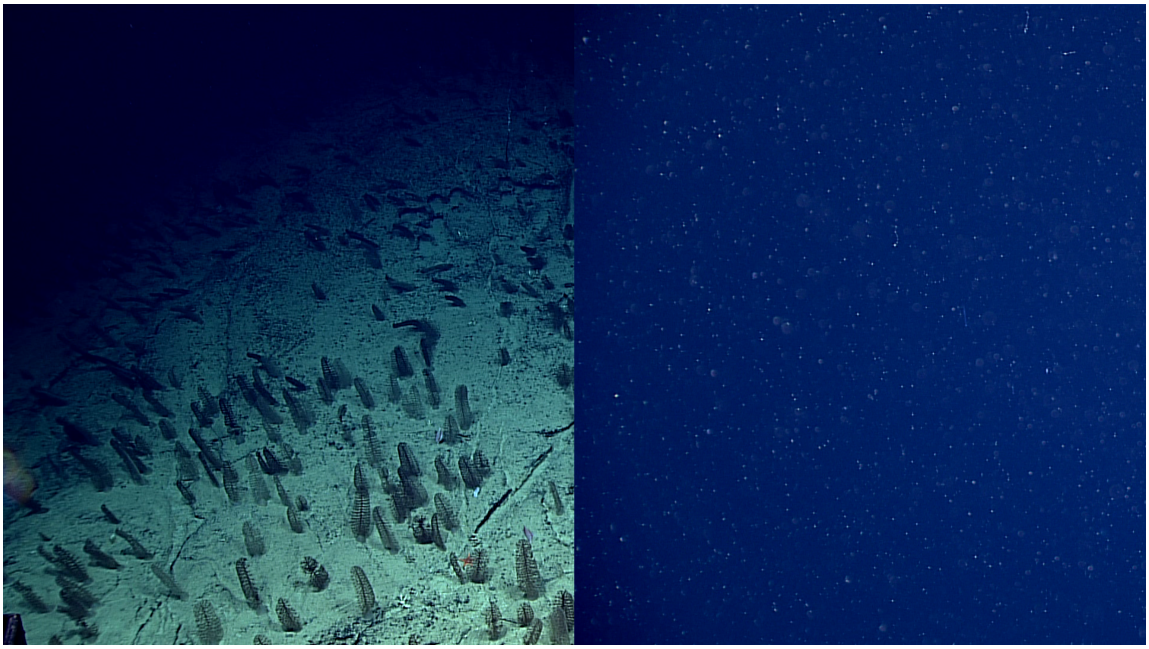


Figure 4.3: Example frame from a 'clean' sequence (left) and a 'marine snow' sequence.

next datasets expanded on this idea by extracting marine snow from the underwater images which exclusively pictured marine snow, and superimposing it onto feature-rich backgrounds.

4.2.1 Unmodified Keypoint Classification Dataset

The unmodified dataset came from an idea to exploit the lack of texture in underwater images which are far away from the ocean floor and man-made objects, and therefore contain no geological or structural features of any kind. In such images, the surrounding water will appear as blue, feature-less background, owing to the light scattering and absorption properties of water. Consequently, any keypoint detected in such an image must come from organic matter such as marine snow and marine wildlife. By running keypoint detection on videos which only depict marine snow on a flat background, we can know for certain that any detection is a marine snow keypoint. Similarly, by collecting underwater sequences near the ocean floor which contain no marine snow at all, we can rapidly generate labelled keypoints with minimal manual effort. This is important because a single image can contain thousands of keypoints which would take a significant amount of time to manually annotate in a sequence with mixed classes.

In Figure 4.3, an image from the 'clean' class is seen on the left, while the image containing only marine snow is on the right (keypoints have not been high-

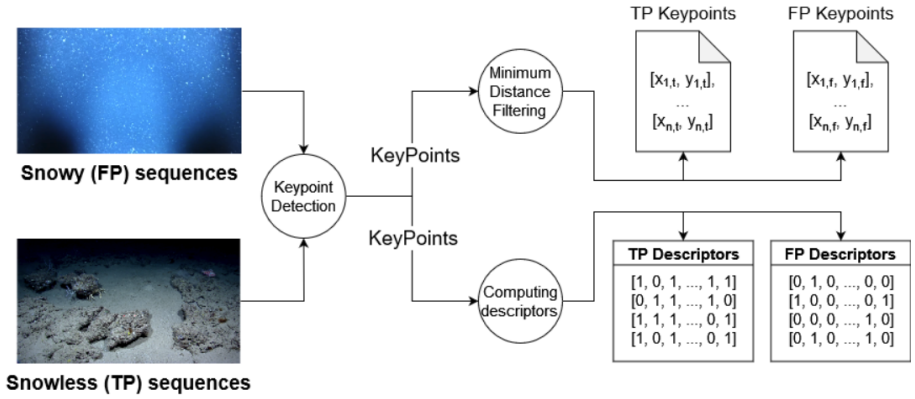


Figure 4.4: Data generation and filtering with pre-labelled Snowy/Snowless sequences

lighted). To find sequences for keypoint detection and description, we used the National Oceanic and Atmospheric Administration (NOAA) Ocean Exploration Video Portal which includes thousands of videos of expeditions conducted by Open Educational Resources (OER) sponsored vessels. We separated these videos into the snowy and snowless collections, and extracted frames at a fixed interval from these sequences to limit the size of the dataset, and avoid nearly identical frames from videos with mostly stationary cameras. With these frames, we conducted keypoint detection and description using ORB, as outlined in Figure 4.4, and stored the keypoint coordinates and descriptors separately.

4.2.2 Superimposed Snow Datasets

One weakness of the unmodified dataset is that snow is always found on very uniform, flat image patches. If we compare this data to the snowless samples, we notice that a model can obtain very high accuracy by only relying on basic attributes such as the uniformity of the background, its colour, or its variance. Expanding the snowy dataset with more diverse backgrounds would significantly increase the visual complexity and diversity of the snowy training data, since their background is no longer featureless and untextured. For example, the new backgrounds can include a combination of man-made objects, geological features, and wildlife. Such backgrounds may be important to train a classifier which generalises to most underwater scenarios, while also being required to ensure thorough evaluation of marine snow classifiers.

To amend the issue of predictable backgrounds, we propose a system which extracts marine snow from the flat 'marine snow' images, and superimposes it onto more interesting backgrounds using a weighted sum. The weighted sum approach was common in our literature review, both in superimposing based synthetic datasets [Sato, Ueda, and Tanaka 2021; Y.-F. Liu et al. 2018], and as a model for image enhancement networks [Y.-F. Liu et al. 2018; P. Li et al. 2019].

An alternative approach to snow extraction is to model synthetic snow such that it can be generated algorithmically and then superimposed onto underwater imagery. This approach can guarantee accurate ground truths for use with keypoint classification, and potentially snow segmentation and snow removal. The challenge of this approach, however, is to create a model which is realistic enough to make classifiers trained on such images generalise to real-world data. Our impression from the literature review is that for small marine snow particles, synthetic approaches can retain high realism. However, these small particles are less likely to be detected by a keypoint detector, so this benefit may be moot.

We believe that a benefit of the snow extraction and superimposing approach is that it can yield snow which is more realistic than fully synthesised snow—especially for larger particles—since it is based on real-world imagery. Furthermore, since snow extraction can be done on videos, this method provides a very straight-forward path to temporally accurate motion of superimposed snow. However, a potential challenge is the level of supervision needed to extract and superimpose snow, since individual sequences may benefit from tailored parameters. Also, while realistic motion can be achieved when extracting snow, some of this realism is lost when superimposing snow onto background videos, since the motion often becomes unlikely or downright impossible when displayed in front of another video. However, it is not clear to us that this lack of realistic motion actually matters for training and evaluation of snow classification, even if a temporal information is used for prediction, because neural networks have no notion of what is realistic or not.

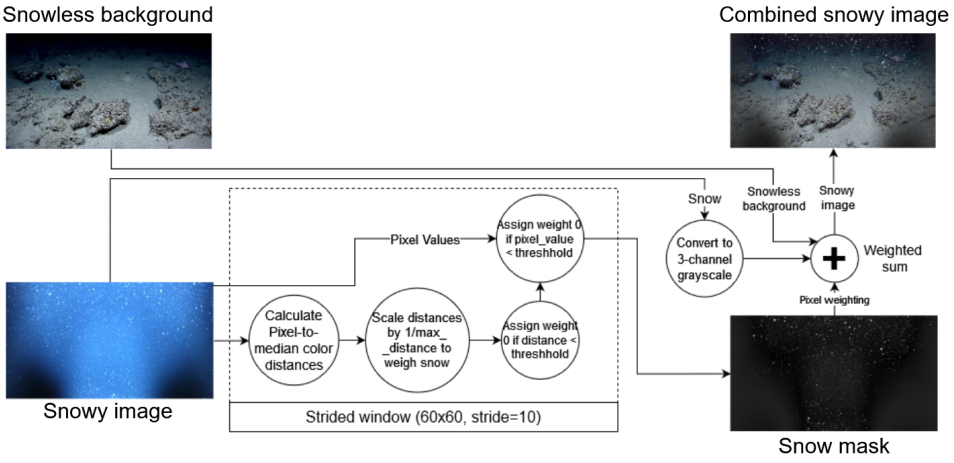


Figure 4.5: Data augmentation pipeline to superimpose snow onto snowless sequences

Our design is rooted in the following five-step process: (1) create an *alpha-mask* on the snowy image based on a minimum distance from the median colour. When combining two images into one, alpha masks provide a pixel-wise weighting of how much information to take from each image. (2) extract snow from the ex-

isting snowy sequences with the distance-based alpha-key, (3) perform keypoint detection on the extracted snow and snow free background image, (4) superimpose the snow onto any snow free background image with summation weighted by the alpha-mask, and (5) use this combined image to extract image patches and descriptors with the existing keypoints.

We divide the superimposed data generation method into two pipelines, one to extract and superimpose snow, and another to detect keypoints and generate descriptors. These are outlined in Figure 4.5 and Figure 4.6, respectively.

To design our snow extraction method, we can actually exploit the textureless background of our snowy images and use a simplified version of the implicit snow detection methods found in the model-based snow removal algorithms from our literature review [Banerjee et al. 2014; Farhadifard, Radolko, and U. v. Lukas 2017; Boguslaw Cyganek and Gongola 2018]. We employ a moving window approach which computes the distance of each pixel to the window median. This distance serves as a weighting between background and snow, where high distances signify snow and low distances signify background. Importantly, this requires that the window size is sufficiently large such that the median colour is not shifted to the colour of any snow particle(s) present in the window. The distances are scaled by the inverse of the maximum distance to give the weighting. Weights are set to zero if the distance is below a threshold in order to remove as much of the background as possible. Similarly, if the pixel corresponding to the weight is particularly dim in the original image we set the weight to zero, as darker areas with high distance to the median colour often correspond to shadows and other illumination artefacts. The moving window traverses the image with a stride s which is significantly smaller than the window size. Because of this, we can use the mean weight of the overlapping windows to reduce the impact of large marine snow particles which shift the median colour away from the background colour.

This method can be extended to multi-scale window sizes by averaging the weights over all scales. However, we argue that prioritising larger windows is important to avoid shifting the median colour when marine snow fills most of the window. Furthermore, the smallest marine snow particles which may not be extracted with an oversized filter are those which are least likely to be detected by a keypoint detector, hence these points are less important to extract.

The final step step is to create a grayscale version of the extracted snow to desaturate the snowy image. This is useful because otherwise the extracted snow retains some of the hue from its original sequence, which can give the snow and unrealistically blue appearance after superimposing. We consider omitting this step to be an optional data-augmentation method.

Unlike the unmodified dataset, superimposed datasets have both true positive and false positive keypoints in each image. Consequently, some adaptations must be made when extracting keypoints, labels, and descriptors. We first tried extracting keypoints on the combined, superimposed image with the idea that we could use the extracted marine snow to determine if a keypoint was detected on

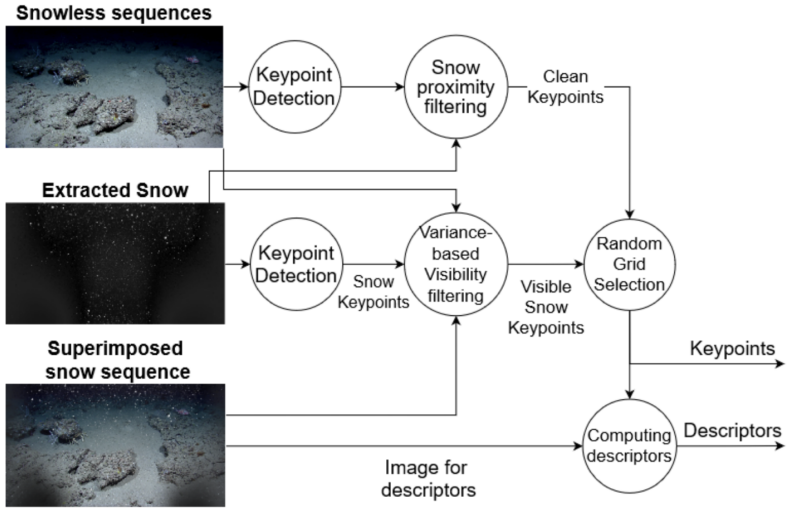


Figure 4.6: Data generation pipeline with superimposed snowy sequences

marine snow or not. However, when visualising the ORB detections, we found that only a small amount of keypoints were actually placed on the marine snow itself, potentially leading to class imbalance issues. Consequently, we explored an approach which detected keypoints separately on the background and extracted snow, since we found that we could generate far more keypoints on marine snow with this method.

Our approach to generate keypoints for superimposed datasets begins by extracting 'marine snow' keypoints on the extracted snow, and 'clean' keypoints on the background image. Then, we perform grid spread filtering (see Section 4.2.4) separately on each class to remove duplicate keypoints in the image. An additional filtering step was needed to exclude snow-keypoints which, while visible in the snow mask, become indistinguishable from the background when superimposed. Such keypoints appear snowless when extracted as image-patches, and are likely ignored as noise when descriptors are generated. Thus, keeping such keypoints is akin to mislabelling the dataset. Consequently, we developed variance based visibility filtering, which compares the variance of an image-patch extracted from the background image with and without superimposed snow, and remove a keypoint if the variance is not sufficiently increased. The patch size is determined by the keypoint octave, which indicates the scale of its prominent image patch.

We later noticed that keypoints in the background image were sometimes covered or nearly covered by the superimposed snow since we detect these keypoints before superimposing. In general, nearby snow is not necessarily a negative, as both training and evaluation data should account for the fact that good keypoints may be in the vicinity of marine snow—after all, this is the case in real underwater scenarios. However, after examining the superimposed data, some problematic

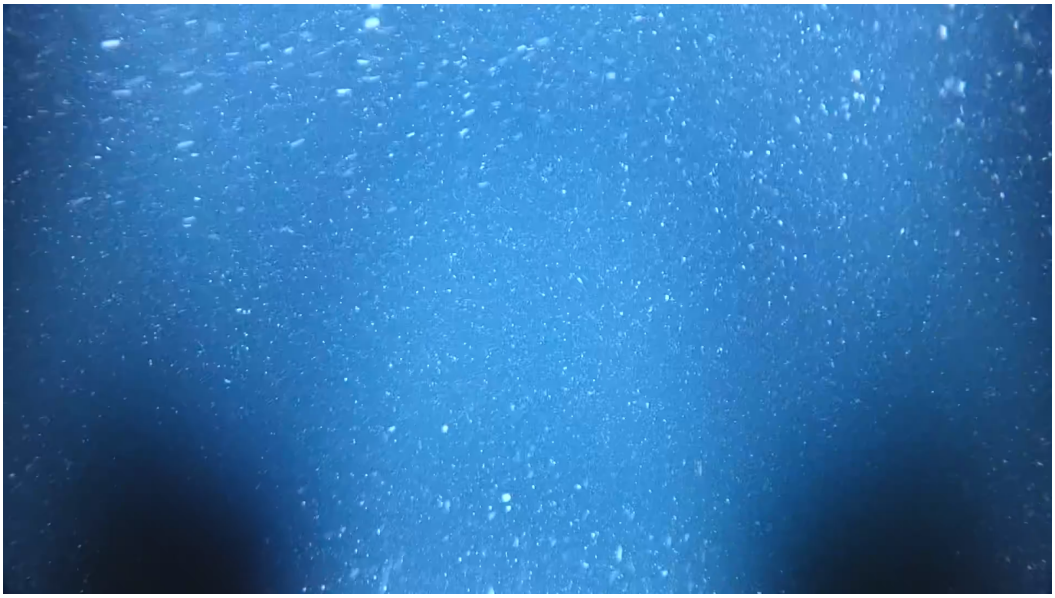


Figure 4.7: An image consisting of snow on a textureless background.

keypoints were present which after superimposing were blatantly mislabelled as 'clean'. Consequently, we implemented an additional filtering step for keypoints from the background image, in which a very small area around the keypoint's location in the superimposed snow is checked for snow which may put a true positive label in question. With patch size 8, we extract information from the superimposed snow and reject the keypoint if the maximum value within this patch is above a threshold. We use a small patch size because we only want to remove keypoints which are significantly affected by nearby snow. Therefore, keypoints which are slightly near marine snow still remain, which we hope will teach the classifiers to operate in conditions with particularly dense snow.

In Figures 4.7, 4.8, and 4.9, we detail three stages of our superimposing approach. Respectively, the figures depict a marine snow image, its extracted snow mask, and a superimposed image combining the extracted snow with an underwater background using the snow mask as weight.

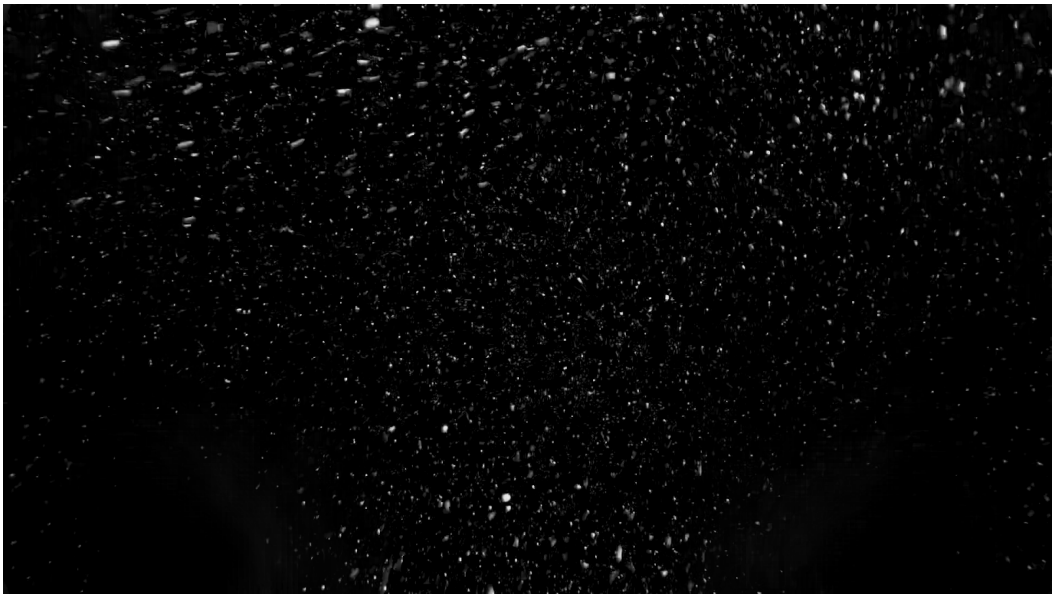


Figure 4.8: A mask of the extracted snow from Fig. 4.7

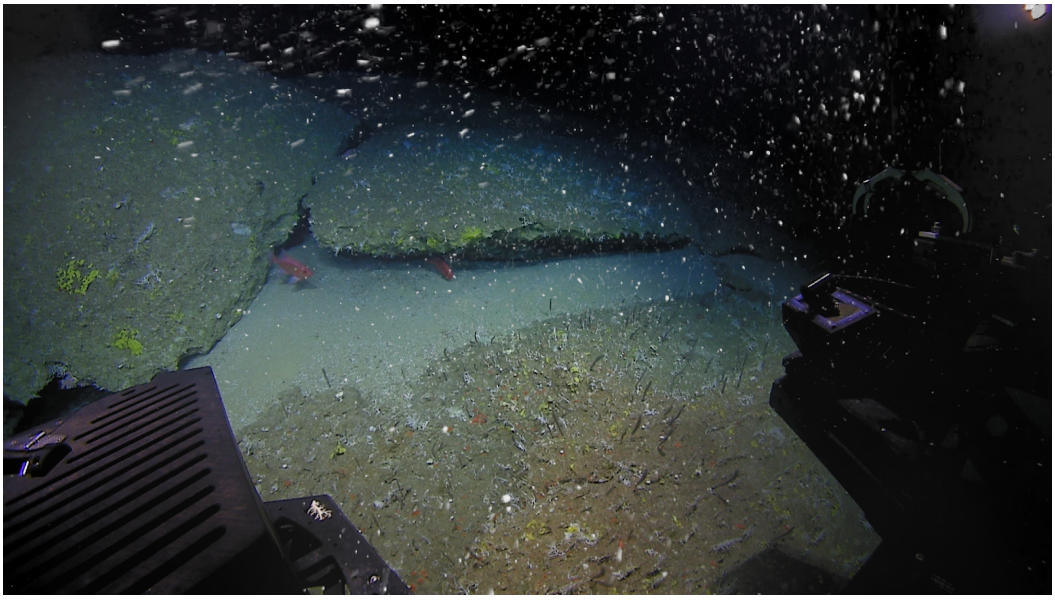


Figure 4.9: The snow from Fig. 4.7 superimposed on a background using the snow mask from Fig. 4.8

4.2.3 Overwater Backgrounds

While the NOAA Video portal was a good source for videos both with and without marine snow, it was relatively time consuming to collect videos from this service due to the sparsity of videos exclusively with marine snow, and especially videos completely without marine snow. This had the effect of reducing the diversity of the environments present in the snowless dataset, an issue which was exacerbated by sequences which came from the same expedition, and others in which the camera barely moved. This lack of diversity can harm the generality of the dataset and increases the chance of overfitting to the data. Another weakness is the lack of man-made objects in the sequences, which comes from the nature of the scientific expeditions in which the sequences were recorded.

Considering the above limitations, and the fact that we had already collected a significant amount of data from NOAA, we searched for other methods of collecting background images for superimposing. It can be argued that a marine snow detector should be able to recognise marine snow regardless of the background which surrounds it. Therefore, we decided to use overwater images to create a second superimposed dataset, separate from our superimposed dataset using underwater backgrounds. This would drastically increase the diversity of the superimposed datasets, and therefore reduce the chance of overfitting. Our hypothesis was that this added diversity would also aid the models' ability to generalise to unseen underwater environments since they would be unable to rely on background information to classify snow.

To maintain some degree of similarity with underwater environments, and to increase the chance that the superimposed snow was actually visible in the combined image, we used images from the Exclusively-Dark-Image-Dataset [Loh and Chan 2019] as backgrounds. While most datasets with dark images are aimed toward city-driving tasks at nighttime, and therefore mainly contain city environments with street lights, this dataset is intended for image classification of dark images. Therefore, it presents a more varied selection of dark environments compared to most night-time datasets. For example, both indoor and outdoor environments are given, city and nature surroundings are present, as well as a range of lighting conditions ranging from near pitch-black to sunrise.

Because the images of this dataset are in all kinds of aspect ratios and generally lower resolution than the images we extract snow from, we upscale them to 1920×1080 resolution to make them fit the superimposed snow. While this will in some cases significantly distort the image ratio, e.g. from a portrait orientation to a landscape orientation. This should not be an issue when extracting keypoints and descriptors, as keypoint detection does not rely on true-to-life scales.

4.2.4 Keypoint Spread and Grid Based Filtering

When extracting keypoint coordinates with ORB, we observed that keypoints tend to cluster together very closely, and even overlap on the exact same coordinates since ORB extracts features independently at eight different scales. To eliminate unintended duplicates in the coordinates, we removed keypoints based on a minimum Manhattan distance threshold.

We later improved this filtering step by selecting keypoints to improve their spread in the image. By increasing the keypoints' spread, we can utilise more of the diversity within the image itself, while reducing the number of duplicate keypoints in the dataset. To spread the keypoints in the image, we employ a grid-based selection scheme reminiscent of the bucketing method sometimes used in SLAM and VO to select correspondences. We start by detecting a set of keypoints, larger than the target amount of keypoints in the final output. We then divide the image into a 10×10 grid and take a random keypoint from a random grid cell until the output set reaches the target size. This filtering is done separately on true positive and false positive keypoints since we do not want to exacerbate class imbalance issues. Additionally, in the superimposed datasets, nearby keypoints from different classes may highlight some of the smaller nuances that aid in classification, meaning nearby keypoints with different classes should be kept.

Failed approach to optimise spread

Inspired by a metric we proposed in the pre-project report in 2021, we attempted to optimise a formula related to point distributions in a grid, but we later realised that the metric did not spread keypoints. Of course, optimising it was also quite computationally demanding. Since we do not need a perfect spread, the heuristic approach above is entirely sufficient for our needs. However, in case it is of interest, we still present the failed approach below.

The optimisation approach is as follows: from the set K of all keypoints, we begin with a randomly selected keypoint and add it to the set of filtered keypoints F. F is iteratively expanded with keypoints from K, by selecting the keypoint whose average distance to the points in F is closest to a pre-set value e. e is computed based on the dimensions of the background image and is the expected distance of two randomly selected points in a $H \times W$ grid [Mathai, Moschopoulos, and Pederzoli 1999]. Mathematically, e is given by this formula, where $d = \sqrt{H^2 + W^2}$:

$$e = \frac{1}{15}\left(\frac{W^3}{H^2} + \frac{H^3}{W^2} + d\left(3 - \frac{W^2}{H^2} - \frac{H^2}{W^2}\right) + \frac{5}{2}\left(\frac{H^2}{W}\log\frac{W+d}{H} + \frac{W^2}{H}\log\frac{H+d}{W}\right)\right)$$

Optimising this metric does not generate "seemingly random" distributions, instead, it creates a ring around the centre. We believe that this measure is simply correlated with spread, which was the source of our misconception.

4.2.5 Snowy-VAROS

Our last dataset is an extension of the existing VAROS dataset [Zwilgmeyer et al. 2021] with superimposed marine snow. We name this dataset Snowy-VAROS. Since the relative pose ground truths of VAROS still apply after superimposing marine snow, the Snowy-VAROS dataset is still useful for quantitative evaluation of pose estimates. Furthermore, comparing results between Snowy-VAROS and the original, snow-free VAROS sequence allows us to evaluate the results of key-point rejection more definitively than most experiments.

4.2.6 3D-Modeled Marine Snow Dataset

In the early stages of the master's project, we experimented with an idea to model snow *within* the 3D-environment used to render the VAROS dataset [Zwilgmeyer et al. 2021]. This approach could obtain ground truths for both marine snow removal, segmentation and detection, in addition to the existing relative pose and depth ground truths of VAROS. This idea is different from the Snowy-VAROS dataset in the sense that it incorporates 3D-rendering of marine snow instead of 2D superimposing.

We collected and analysed examples of marine snowflakes to use as references when modelling our own flakes for the VAROS dataset. In our experiment, the snowflakes were modelled as translucent, irregularly shaped ellipsoids and added to the VAROS scene through a particle emitter. To model the translucency of a specific point on a marine snowflake, we used the angle between the normal vector of the point on the ellipsoid and the vector between the camera and the point. This emulates the effect of marine snow being increasingly translucent closer to the edges. An example render is found in Figure 4.10.

The marine snow emitter was set to only emit particles close to the observer, to eliminate the processing of particles that would not be visible, similar to methods from the literature review [Langer et al. 2004; Bernuth, Volk, and Bringmann 2019]. To emulate ocean currents, all particles were placed in a constant motion field, while each particle had Brownian motion enabled, to simulate the motion introduced by a marine snowflake's unique shape interacting with local currents. Had we continued on with this dataset, the constant motion field would eventually be replaced with one of the many alternatives in Blender which model more complex motion fields that more accurately emulate ocean currents.

Compared to our other methods, this method of simulating marine snow had the benefit that the motion of the marine snow was consistent with the motion of the camera, unlike the superimposing method, where the camera motion of two videos are forcefully combined into one video. However, the limitation of only being able to add snow to the VAROS dataset, the limited amount of unique snowflakes, and the uncertainty about whether the 3D modelled snow would generalise to real snow compelled us to put this method on hold in favour of the superimposing method.

Figure 4.10: A render of snow particles added to VAROS in blender. A particle system emitting spheres of fixed size was used (essentially the most basic implementation).

Chapter 5

Implementation

In this chapter, we provide more detailed descriptions of which frameworks, parameters and architectures we used in our experiments.

5.1 Architectures

Below we present the serial architectures we implemented, one using descriptors (D-CLAS) and one using image patches extracted from keypoint coordinates (P-CLAS). All deep learning architectures were implemented using the PyTorch framework, while other binary classifiers were implemented using scikit learn.

5.1.1 Descriptor-based Classifier

D-CLAS uses a standard FCNN architecture with binary vector input, ReLu activation function in the internal layers and Sigmoid activation in the final layer. The architecture is described in full in Table 5.1.

Layer	L1	L2	L3	L4	L5	L6	L7
Activation Function	ReLu						Sigmoid
Dimensionality	256	196	196	128	64	16	1

Table 5.1: First experimental Neural Network architecture for the descriptor classifier.

D-CLAS was implemented in PyTorch and trained for 1 epoch on a GTX1080 GPU with binary cross-entropy loss and the Adam optimiser with default hyperparameters. When making a classification based on the sigmoid output, we used a threshold of 0.5, meaning outputs greater than 0.5 were considered snow, and clean otherwise. This is the case for the P-CLAS as well.

The other descriptor classifiers which D-CLAS is compared to in Table 7.3, such as random forest, K-nearest neighbours, Gaussian process, linear support vector classifier, Gaussian naive Bayes, and quadratic discriminant analysis were all implemented using the scikit learn framework.

5.1.2 Patch-based Classifier

Our P-CLAS architecture stacks 5 sets of convolutional layers, each followed by batch-normalisation and ReLu. We used 3×3 kernels with a step size, s, of 1 in all convolutional layers. The flattened output of the last convolutional layers is given to a fully connected layer which outputs the classification with a sigmoid activation function. Figure 5.1 illustrates the architecture of P-CLAS, and the number of kernels in each layer.

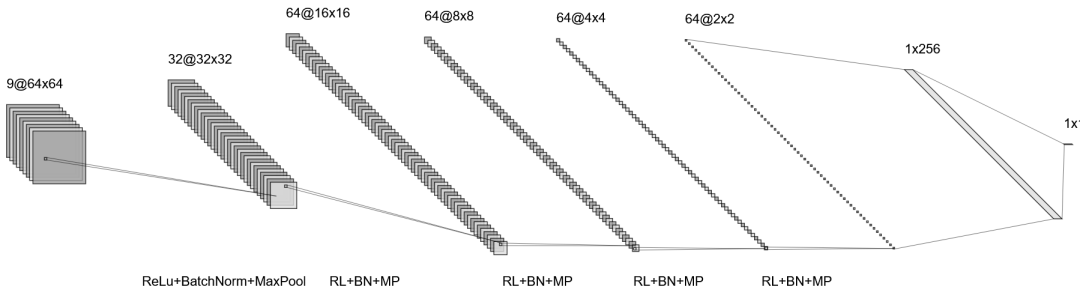


Figure 5.1: The CNN-based P-CLAS architecture

Patch extraction was done with three scales: 64×64, 48×48, and 32×32 pixel squares centred on the keypoint. These were all resized to 64×64 using bi-linear interpolation and stacked in the channel dimension before classification. This architecture was implemented in PyTorch and trained for 1 epoch on an RTX2080 GPU with binary cross-entropy loss and the Adam optimiser with batch size 8 and a learning rate of 1e-4. During training, the image patches are subjected to a rotation data augmentation step, which randomly rotates patches in increments of 90 degrees.

5.2 Datasets

To train and evaluate our architectures, we have made three datasets for binary classification of keypoint coordinates and descriptors. We also made a fourth dataset—Snowy-VAROS—for evaluation of SLAM systems in marine snow conditions. We present the datasets and process to create them here. The keypoint classification datasets and their sizes are listed in Table 5.2.

	Images	Snow KPs	Background KPs
Unmodified	6,008	598,931	1,181,570
Underwater	10,051	1,525,556	2,227,102
Overwater	8,705	1,772,123	2,055,001
Total	24,764	3,896,610	5,463,673

Table 5.2: The keypoint classification datasets and their sizes. Train, val and test splits were made following 80/10/10 splits

5.2.1 Data Gathering

Most of our data consist of videos from NOAA expeditions which were retrieved from the Ocean Exploration Video Portal[1]. The videos, which are listed in Appendix C, were manually screened for the presence of marine snow. The ORB-features from videos without marine snow were collectively labeled as clean keypoints, while keypoints and descriptors from sequences with exclusively marine snow were labeled as snowy keypoints. Since the snowy sequences contain no other features than marine snow, any detection from a keypoint detector must be marine snow, therefore no manual labelling is required. Figure 4.3, shows one frame from the snowy and snowless sequences. Furthermore, we received data from our supervisors at the Autonomous Robots for Ocean Sustainability (AROS) research group, who have collected videos with the Eelume[2] robot in Trondheimsfjorden. These sequences contained marine snow quite distinct from the marine snow in the NOAA sequences, appearing particularly dense and large. However, Eelume had no sequences without marine snow so only exclusively snowy sequences were collected from them.

Because the NOAA videos were interlaced, meaning their frames were blended together in striped sections, we performed deinterlacing in Blender to create videos with progressive frames to match real-time operating conditions underwater. Afterwards, we extracted every 20th frame from the Eelume and NOAA sequences, and split the dataset with an 80/10/10 training, testing, and validation split, while ensuring that each video only appeared in one of these data splits.

5.2.2 Unmodified Dataset

We first developed a dataset for classification of keypoint patches and descriptors using the ORB keypoint detector and descriptor with default settings in OpenCV. This detector is among the most widespread in SLAM, and its binary descriptors are well suited for neural network classifiers. However, to make sure we made the correct choice, we used the same ORB keypoint detections to generate three more descriptors: SIFT, FREAK and VGG120. Since all of these were used with the ORB *detector*, we could compare the abilities of each descriptor in snow classification.

[1]https://www.ncei.noaa.gov/access/ocean-exploration/video/
[2]https://eelume.com/

Figure 5.2: Snow superimposed on Snowy-VAROS. Figure from [Hodne, Leikvoll, et al. 2022].

For the unmodified (**U**) dataset, which uses the collected images as they are with no superimposing, we detected 500 keypoints per image, and used random grid selection to limit the number of keypoints per image to 300. In total, we used 3941 background frames, and 2067 snowy images.

5.2.3 Superimposed Datasets

The superimposed datasets were intended to increase our data's variety and generality. A weakness of our initial U dataset was the trivial characteristics of the image patches, particularly the marine snow class which mainly featured white blobs on blue backgrounds. Consequently, for a patch labelled snowy, we can with high accuracy predict that its background is flat, mostly untextured and featureless besides the marine snow. This is undesirable because it makes the prediction task far easier during training than in the real world.

The underwater (**UW**) superimposed dataset used 3941 background frames, and superimposed 2067 snowy images from Eelume and NOAA, while the dark overwater (**OW**) dataset used 6438 background images, and the same 2067 snowy images fro superimposing. With the UW dataset we superimposed each frame of extracted marine snow onto 4 random backgrounds, as opposed to 5 random backgrounds for the OW dataset. Both datasets used two data augmentation steps: firstly, we superimposed either a grayscale or colour version of the extracted marine snow with 50% probability each. Second, we flipped the extracted snow along the horizontal axis, the vertical axis, both axes, or neither with equal probability.

To create Snowy-VAROS, we used a subsequence of VAROS where the robot travels above a straight pipe (see Fig. 5.2), because the full VAROS dataset, is a very challenging sequence for traditional SLAM systems. However, the pipe subsequence offers better features such as corners and edges for keypoint detection than other parts of VAROS which move primarily above mounds of sand. Additionally, the straight pipe makes it easy to judge the quality of the tracking by the shape of the pipe in the sparse map made by the SLAM pipeline.

5.2.4 Snow Extraction

The snow extraction process was first introduced in our paper *Detecting and Suppressing Marine Snow for Underwater Visual SLAM*, and is further discussed below.

During snow extraction, we use a 60×60 window, P, with a stride of 10. For each window, P, we then calculate the RGB-distance, D, between all pixel values p in P and the median colour of the window, M_P. In order to make a mask of weights, W, between 0 and 1, we scale the distances by the inverse of the maximum distance. To further reduce the effect of the background we use a threshold to set the weight of pixel p, W_p, to 0 if its distance to the median colour, $D(M_P, p)$, is below the threshold value $\tau_D = 30$. This threshold value was determined through experimentation, and 30 was found to be low enough to not affect snowflakes while also being high enough to eliminate much of the background.

Because marine snowflakes usually appear as bright spots, we also use a threshold, $\tau_I = 20$, on the grayscale pixel intensities, I_{GS}. This is needed because shadows often appear in the snowy images and have a high distance from the median colour which we do not want to include in our mask. We use $I_{GS}(p)$ to denote the grayscale intensity of a pixel at location p. The full calculation of the mask weight at location p in P can be seen in Equation 5.2. Equation 5.1 describes the function used to find the distance between two pixels.

$$D(p, q) = |I(p) - I(q)|. \tag{5.1}$$

$$W_p = \begin{cases} 0 & \text{if } I_{GS}(p) < \tau_I \\ 0 & \text{if } D(M_P, p) < \tau_D \\ \dfrac{D(M_P, p)}{\max\limits_q D(M_P, q)} & \text{otherwise.} \end{cases} \tag{5.2}$$

For each pixel, its average weight from all windows is used when extracting snow. This is to ensure that windows with large marine snow particles do not introduce image artefacts after superimposing the snow, due to the shift in median colour caused by the large snow particle. From a snowy image S, background image B, and alpha-key weight W, we superimpose images following Eq. (5.3).

$$I = B \odot (1 - W) + S \odot W. \tag{5.3}$$

5.2.5 Keypoint Filtering

For the superimposed datasets, we detect 1400 ORB keypoints per superimposed image, where 700 clean keypoints were detected in the original background image, and another 700 marine snow keypoints in the extracted snow. To remove samples which become mislabelled after superimposing, we used our variance-based filtering and snow proximity filtering methods.

The variance-based filtering for snow-keypoints used a variance threshold of $\mathcal{E} = 14$, and patch-size twice the size of the ORB keypoints' size parameter. The threshold, $\mathcal{E}$, was selected by sampling keypoints which were filtered. We then evaluated their resemblance to marine snow in the combined image until borderline acceptable cases were kept, and clearly misleading ones were removed. Formally, a keypoint, K_s, from the extracted snow is removed if the inequality in Eq. (5.4) is false, where P_{Si}, and P_{BG} are image patches around K_s in the superimposed image and background image, respectively.

$$\mathsf{Var}\left[P_{Si}\right] > \mathsf{Var}\left[P_{BG}\right] + \mathcal{E} \tag{5.4}$$

The snow proximity filtering used a patch size of 8×8 when checking for snow near keypoints detected in the background image. The patch, P_{ES}, is taken from the extracted snow of the snowy image. If the patch's maximum colour channel intensity was greater than 70 in the extracted snow, as seen in Equation 5.5, the keypoint was removed due to a potentially misleading label as free of snow. Again, this threshold was selected by sampling keypoints which were removed, until a fair filtering was found.

$$\max P_{ES} > 70 \tag{5.5}$$

Finally, after filtering the keypoints to remove misleading samples, we applied random grid selection separately on each set of keypoints to extract up to 250 keypoints from each class.

5.2.6 Dataloaders for Limited Working Memory

It quickly became apparent that the data pipelines we made could generate far more data than what our computers could store in 16GB of working memory. Consequently, we made custom dataloaders which prioritised two competing objectives: minimising reading from disk, and maintaining random ordering of samples in the training dataset.

For the patch-dataset we elected to extract patches within the dataloader during training, as opposed to extracting patches beforehand and saving them as part of the dataset. Consequently, all images must be loaded at least once to extract its patches, meaning memory consumption can rapidly exceed what resources

we had. We made this decision in order to offer a more flexible dataset, where all properties of the extracted patches were determined by the model, not the dataset.

Since one image corresponds to hundreds of keypoints, we made a Least Recently Used (LRU)-cache which only stores a small subset of the dataset's images. One challenge of this approach is that caching is only effective if it has a high hit-rate, meaning when an image is needed it should, with only a few exceptions, be present in the cache. Since all images in the dataset must be available whenever one of its keypoints is used for training, a completely random ordering of the keypoints would lead to frequent misses of the cache; an LRU-cache is based on the assumption that if an item in the cache is removed, it will not be needed for a long time, an assumption which is completely violated with random ordering.

Thus, our objective for the patch-dataset is to order the dataset in a way which maximises the cache hit-rate but maintains some degree of random distribution of images and labels. Conceptually, this can be achieved with an ordering of the dataset, D, in which the keypoints of an image, K_i are located in a subarray, A_i, of D where the cardinality of A_i never exceeds a given size t_i, and $t_i \propto |K_i|$. Increasing t_i effectively increases the randomness of the keypoints since an image's keypoints can be present in a larger section of D. However, this increased spread risks reducing the cache hit-rate if the cache size is not large enough to accommodate a higher value.

Our design follows this example by ordering the dataset before training begins. When ordering the keypoints, keypoints are selected at random from a small, ordered set of images S. When an image in S has no keypoints left, it is removed, and a new, randomly selected image is placed at the rear of S. We use a discrete probability function when selecting which image in S to extract a keypoint from. This distribution decreases linearly, meaning it is biased towards items at the front of S. This is intended to prevent the subarrays, A_i, from clustering together.

Figure 5.3 shows an illustration of a probability distribution similar to ours prior to normalisation. To create a distribution, we specify the slope of the linear function which defines the distribution, and its offset at the last sample which makes sure the probability which corresponds to the last item in S is greater than zero. The function is then sampled at $x = 0,\ 1/|S|,\ \ldots\ ,\ |S|/|S|$ and normalised. This means that we cannot specify t ourselves, but rather define slope, offset and the cardinality of S, and measure t_i afterwards. Testing with approximately 1.000.000 keypoints from 4417 images, $|S| = 100$, and a cache size of 300, we noticed a change in hit-rate from 8% with random ordering to 99.6% with our ordering. We later confirmed that this hit-rate was as good as it could be, meaning each image was only missed once. We also measured the median cardinality of A_i to be 26155, i.e., for 50% of the dataset, $t_i \leq 26155$.

Even with $|S| = 250$, the hit-rate was unchanged at 99.6%, meaning the lru cache never loaded an image twice and works even with the median t now at 64950. However, we stayed with the smaller value of $|S|$ to accommodate datasets with a greater number of keypoints per image than the one we tested.

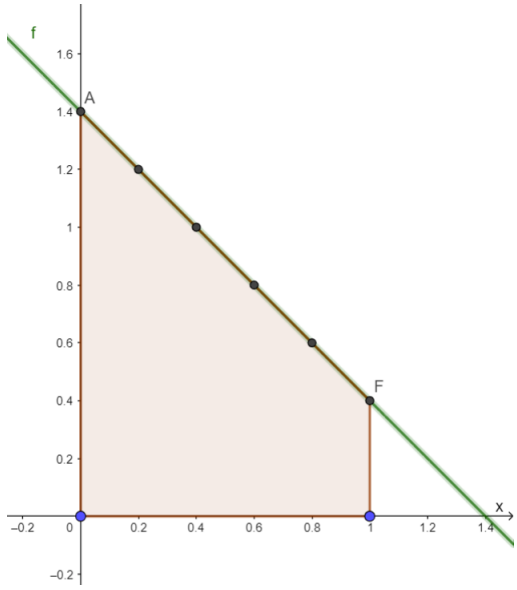


Figure 5.3: Example of a probability mass function used for dataset ordering (after normalisation). This example has a slope of -1.0, an offset of 0.4, and $|S| = 6$

While the descriptor dataset was less memory-intensive than the patch-dataset, the sheer number of samples present in our datasets meant that a conservative approach to memory use was still required. The descriptor datasets consist of two folders—one for each label. Each folder contains .npy files which correspond to the descriptors of an image with the same labels as the other files in the folder. While the patch-dataset needs to load an image for every keypoint, the descriptor-dataset does not need the image at all. Hence, it does not matter if keypoints from one image are spaced far apart in the dataset. Consequently, our only trick to limit memory use was to divide the dataset into multiple subsets, only loading one subset at a time. To ensure random ordering of the data, subsets are generated at random by scrambling the paths of all the .npy-files. After loading the descriptors of a subset, the subset itself is scrambled, because otherwise the subset would be grouped by image and label.

5.2.7 Other Opportunities

We made our datasets not just as a snow detection benchmark, but as a task agnostic dataset that can be used for other tasks as well *e.g.* snow removal and segmentation. Therefore, our snow extraction and superimposing implementation make it possible to store both the extracted snow and the combined snowy image. Alongside the original background image, these can be used to train and evaluate snow removal and snow segmentation methods.

Notably, as part of our published CVPR paper, we have made the documented code for snow extraction and superimposing publicly available with 2067 snowy images, and 3941 snowless images from our NOAA and Eelume videos. These files were published on the Zenodo open data platform[3]. Consequently, the dataset's potential in tasks such as segmentation and snow removal can be unlocked by others.

5.3 SLAM Evaluation Framework

We used a python-based SLAM and VO framework, named pySLAM[4], to evaluate the classifiers in pose estimation and mapping tasks. Pyslam is an open-source platform originally developed for educational purposes. It offers a range of descriptors, detectors, and matchers, and offers built-in mapping, keyframe management, local and global bundle-adjustment, outlier rejection, active matching [Chli and Davison 2008] which predicts the image coordinates of existing keypoints based the inter-frame motion, and motion models to predict pose based on the current velocity—all of which are staple SLAM-techniques.

Using this system allows us to implement the classifiers without worrying about mixing programming languages. This also provides a highly configurable testbed to evaluate how the classifiers complement modern SLAM techniques. One clear downside is that pySLAM and its python implementation would never be used for real-world SLAM, since it is too slow compared to programming languages such as C and C++. However, we argue this is a relatively small problem given the customiseability and simplicity pySLAM offers.

For our purposes we used pySLAM with the ORB detector and descriptor, extracting 2000 keypoints per frame on the real-world sequences, and 3000 keypoints per frame in VAROS and Snowy-VAROS. The snow classifiers were implemented immediately after keypoint description to create a system which can easily swap between the two classification schemes. With that said, for P-CLAS, the most efficient implementation is to classify keypoints before description, and only generate descriptors on keypoints which pass snow classification.

[3] www.zenodo.org/record/6424752
[4] www.github.com/luigifreda/pyslam

Chapter 6

Experiments

Here we present the qualitative and quantitative experiments used to evaluate the classifiers. We conducted experiments to evaluate stand-alone classification performance, and performance in SLAM use cases. We also include an ablation study, to verify the design choices made for the datasets and detectors.

6.1 Criteria for Research Questions

To design experiments and assess their results in relation to the research questions (Section 1.2), we formulate evaluation strategies related to the following topics:

- Quantitative metrics on test datasets
- Qualitative results on real-world sequences
- SLAM performance
- Execution speed

An explanation of each topic and its relevance to the research questions is given below:

Quantitative metrics on test datasets: We base the quantitative results on binary classification metrics obtained from various combinations of our test dataset splits. These results indicate the ability of the models to generalise to the problem space they have been trained on, but do not necessarily indicate real world abilities since most of the data is synthesised.

Qualitative results on real-world sequences: The qualitative evaluation of keypoint classification intends to document performance on real video sequences.

Because such sequences do not have ground truth values, we must rely on visualisation tools and judge the classification performance ourselves. Given a broad selection of video material, these qualitative tests can highlight performance in a large set of possible operating conditions. While these results clearly relate to RQ1 (How can the effect of marine snow on keypoint detection, matching, and real-time SLAM be mitigated?), they are also relevant for RQ2 (How can marine snow be digitally synthesised for use in machine learning?). This is because the contrast in performance between the quantitative and qualitative tests can highlight the ability of the models to generalise from synthetic to real marine snow, potentially indicating shortcomings of the datasets should they be unable to do so.

SLAM performance evaluates how the snow classification impacts pose estimation and mapping, and enables comparisons of SLAM performance with and without marine snow classification. Additionally, it can highlight how existing keypoint rejection methods cope with marine snow of varying severity. These results will be substantial to our answer for RQ1.

Execution speed is important to support real time operation in SLAM pipelines, in accordance with RQ1. Consequently, we evaluate both the keypoints processed per second by each method, and cycle times in pySLAM with and without the classifiers.

6.2 Experimental Setup

In this section we describe the experimental setup of each of our experiments such that they can be reproduced by others.

6.2.1 Quantitative Experiments

For stand-alone performance, we used test splits of our U, OW, and UW datasets and evaluated F1 score, accuracy, True Positive Rate (TPR) and True Negative Rate (TNR). As opposed to testing on the datasets separately, we chose to combine various splits during testing, training, and validation. However, as a baseline we included the unmodified dataset as a standalone dataset to compare results to non-superimposed data. Alongside the U dataset, we created the UW+U dataset, OW+U dataset, and lastly the All-dataset which combined all three datasets. These dataset combinations were also used to train and validate multiple versions of P-CLAS and D-CLAS.

We mainly view the superimposing process as a method of extending datasets, not necessarily generating them from the ground up (although this is certainly possible too). Therefore, the superimposed dataset splits used in our experiments were always paired with the corresponding split from the unmodified dataset. This approach yields the diversity of the superimposed backgrounds, guaranteed realism

of the unmodified data, as well as the untextured backgrounds of the unmodified 'snowy' images, which, while bland, are frequently encountered underwater.

We also did quantitative evaluation on different implementations of D-CLAS which entailed comparing it to some default classifiers in scikit, and training it with different descriptors, which had all been made from the same ORB keypoints. When training on other descriptors, the only architectural difference was to modify the input layer to the appropriate size, and use real numbered input vectors as opposed to binary input vectors for the VGG120 descriptor.

6.2.2 Qualitative Evaluation on Real World Sequences

Qualitative assessments of keypoint classification were performed on six diverse underwater sequences. For each sequence, we used ORB to extract 2000 keypoints, and then classified these, typically on every other frame from the first 20 seconds of the video. We saved the results in separate recordings with the framerate halved, such that we could analyse the results afterwards at a lower playback speed. We denote and summarise the videos as follows:

- V1: Rocky background with dense snow, both brightly lit.
- V2: Large yellow structure, with very large, dense snow.
- V3: Dark brown sand and rocks with moderate snow under dim, yellow lighting.
- V4: Moves from a close-up of a yellow structure. Extremely dense and large marine snow.
- V5: Flat ocean bed in the distance, with a dense cover of small, fast-moving marine snow.
- V6: Mainly black and white colours, and silty ground with out of focus marine snow.

Here, V2, V4, and V6 are Eelume videos, the others are from NOAA. The Eelume videos generally have lower bitrate than the others. The filenames of the publicly available NOAA videos are given below. They can be accessed through the Open Exploration Video Portal.

- V1: EX1202L2_VID_20120322T135148Z_ROVHD_COR_SPO_FSH.mov
- V3: EX1004L3_VID_20100722T212402Z_ROVHD_ROCK_LEDGE_SLOPE.mov
- V5: EX1304L1_VID_20130718T185323Z_ROVHD_SKATE.mov.

6.2.3 Evaluation of SLAM Performance

To evaluate performance in SLAM use-cases, we used our pySLAM implementation with and without keypoint classification on both real-world sequences and synthetic sequences with and without superimposed marine snow. We visualised

the point cloud and relative poses produced by pySLAM, the keypoints it tracked between frames, and the keypoints rejected by its default outlier rejection scheme based on RANSAC [Fischler and Bolles 1981] and ratio testing [Lowe 2004]. All experiments were conducted with an RTX2080 GPU.

To select sequences for our qualitative tests with pySLAM, our main interest was to evaluate performance under differing marine snow conditions, from no marine snow to dense marine snow. This was done to evaluate both when snow classification may be necessary, and when our networks are successful. A second requirement for the test sequences is that the robot moves sufficiently throughout the recording, because SLAM pipelines are more accurate in sequences with meaningful movement between frames. Consequently, we do not use most sequences from our qualitative keypoint classification tests. Specifically, we evaluate our classifiers on the following sequences:

- Sequence with Exclusively Snow - EX1902_VID_20190514T194459Z_ROVHD.mov
- Sequence with Heavy Snow - Eelume sequence, same as Video 6
- Sequence with Light Snow - EX1202L2_VID_20120322T145121Z_ROVHD_CAR.mov
- Sequence with No Snow - EX1603_VID_20160228T210505Z_PTMAN_ROCK_SIP.mov

While the sequence with exclusively snow is not a sequence where VSLAM would be appropriate, it is still a useful tool to evaluate the behaviour of pySLAM with and without snow classification. Moreover, snow classification may be used to determine when exclusively snowy scenes are encountered, and allow the SLAM system to react to this circumstance.

6.2.4 Ablation Study

The ablation study is intended to compare results from our chosen methods, with modified counterparts which remove certain features. From this comparison, it is possible to better understand the effect of the modified subcomponent. In our study we made the following comparisons:

- A1: Compare a single scale P-CLAS with the standard 3 scale P-CLAS.
- A2: Compare P-CLAS trained on grayscale images with the standard RGB implementation.
- A3: Compare P-CLAS and D-CLAS to models trained on datasets without snow proximity filtering and variance-based visibility filtering.

Experiment A1 was conducted with the dataset suite from the original quantitative experiments, and a new P-CLAS implementation which extracts a single 64×64 patch, in contrast with our standard approach which used three patches at 64×64, 48×48, 32×32.

Experiment A2 was conducted with grayscale images made through the OpenCV BGR2GRAY grayscale transform. These models were tested on the same testsets mentioned above.

To evaluate the effect of snow proximity filtering and variance-based visibility in experiment A3, we trained on UW, OW and UW+OW datasets with and without snow proximity filtering and variance-based visibility filtering. we omitted the U dataset since this dataset has no keypoint filtering. The test results are, however, given on testsets with filtering. This is because a difference in performance between the models must be attributed to the more varied, unfiltered dataset. We do not include random grid selection in the ablation study, even though it can be considered a kind of filtering. This is because it does not limit the domain of the dataset, unlike the filtering steps we evaluate.

6.3 Limitations

The main limitation of our experimental setup is imposed by the difficulty of obtaining ground truths for keypoint classification. Consequently, the vast majority of our quantitative results rely on our superimposing approach. This is also true for the qualitative results obtained with pySLAM in Snowy-VAROS, a dataset which was made with superimposing. This limits the confidence we can reasonably have that the quantitative results generalise to real-world scenarios, since the superimposing approach may introduce artefacts which the networks rely on during classification. Similarly, Snowy-VAROS may favour models trained on our superimposed data, because Snowy-VAROS and the training data were made with the same superimposing pipelines.

Our qualitative tests could be influenced by biases we may have when examining the visual results of our methods. However, we have made an effort to share as much relevant video material as possible, and maintain consistent language when characterising those videos. Another weakness of qualitative tests is that it becomes more difficult to tell apart models which perform similarly, and qualitative results are generally more subjective than quantitative results.

Another concern is similarities between our test and training data. While we made sure to limit individual video sequences to one dataset split when we made the datasets, such that frames from one video may not appear in training and testing data, we did not extend this separation to include videos from similar locations and expeditions. This could lead to unreliable results on the quantitative metrics.

Regarding our experiments in pySLAM, we have already discussed how its python implementation makes measurements of cycle times incomparable with SLAM systems for real-world applications, which are typically made in C++. Another important point is that our analysis of pySLAM's behaviour with and without keypoint classification can not be freely generalised to other, more broadly adopted SLAM systems, because we can not rule out the possibility that the behaviour is just a characteristic of pySLAM alone.

Chapter 7

Results

Here, we present the quantitative and qualitative results of our experiments.

7.1 Keypoint Classification Metrics

This section presents the results obtained on the various test splits of our datasets. The first subsection compares the results of P-CLAS and D-CLAS, while the second and third sections compare D-CLAS with similar architectures based on different classifiers and descriptors, respectively.

7.1.1 Comparing P-CLAS and D-CLAS

In Table 7.1 we present the complete results of all P-CLAS and D-CLAS models on all test datasets. Rows denote the classifier and its training data, while columns denote the test dataset and metric. For simplicity, we begin by looking at Table 7.2, which only contains data related to the U dataset, and All-dataset.

		Unmodified				UW + U				OW + U				All datasets			
		F1	Acc	TPR	TNR	F1	Acc	TPR	TNR	F1	Acc	TPR	TNR	F1	Acc	TPR	TNR
Patch	U	**0.999**	**0.999**	**0.999**	**1.0**	0.617	0.792	0.446	0.999	0.47	0.687	0.309	**0.997**	0.402	0.692	0.252	**0.998**
	UW + U	0.968	0.972	0.991	0.957	0.948	0.961	0.931	0.98	0.854	0.861	0.904	0.825	0.877	0.897	0.894	0.9
	OW + U	0.998	0.998	0.996	0.999	0.913	0.94	0.84	**0.999**	0.94	0.948	0.894	0.993	0.91	0.932	0.839	0.996
	All	0.996	0.996	0.996	0.997	**0.975**	**0.982**	**0.955**	0.998	**0.961**	**0.965**	**0.954**	0.975	**0.964**	**0.971**	**0.95**	0.985
Desc	U	0.945	0.951	0.98	0.929	0.778	0.848	0.712	0.929	0.809	0.833	0.784	0.873	0.763	0.82	0.707	0.899
	UW + U	0.944	0.949	0.978	0.927	0.913	0.933	0.931	0.935	0.892	0.898	0.943	0.86	0.893	0.909	0.93	0.895
	OW + U	0.954	0.959	0.977	0.946	0.916	0.937	0.917	0.949	0.917	0.925	0.919	0.93	0.909	0.926	0.906	0.939
	All	0.955	0.961	0.964	0.958	0.935	0.952	0.926	0.967	0.919	0.928	0.912	0.941	0.921	0.936	0.909	0.955

Table 7.1: Binary classification results by the classifiers. Rows denote the network and its training data, while columns denote the test dataset. The Unmodified dataset (U), Underwater superimposed (UW), and Overwater superimposed (OW) were combined and used for testing. We provide F1 scores, accuracy, True Positive Rates (TPR), and True Negative Rates (TNR) for each case.

		Unmodified				All datasets			
		F1	Acc	TPR	TNR	F1	Acc	TPR	TNR
Patch	U	**0.999**	**0.999**	**0.999**	**1.0**	0.402	0.692	0.252	**0.998**
	All	0.996	0.996	0.996	0.997	**0.964**	**0.971**	**0.95**	0.985
Desc	U	0.945	0.951	0.98	0.929	0.763	0.82	0.707	0.899
	All	0.955	0.961	0.964	0.958	0.921	0.936	0.909	0.955

Table 7.2: Condensed binary classification results by the classifiers. Rows denote the network and its training data, while columns denote the test dataset. The Unmodified dataset (U), Underwater superimposed (UW), and Overwater superimposed (OW) were combined and used for testing. We provide F1-scores, accuracy, True Positive Rates (TPR), and True Negative Rates (TNR) for each case.

In Table 7.2, starting in the left-most columns, we find that both P-CLAS and D-CLAS achieve strikingly high scores on all metrics. P-CLAS in particular does so well on the U dataset that it arguably is a source of worry. However, the metrics are accurate and are most likely caused by the highly dissimilar patches of snow labelled samples and clean labelled samples within the U dataset. Perhaps the most notable result is that both P-CLAS and D-CLAS models trained on the All-dataset—*i.e.* 81% synthesised data—still generalise to the unmodified dataset, with the P-CLAS model matching its U-trained counterpart, and the D-CLAS model improving substantially on all metrics except TPR.

Moving over to the second column of Table 7.2, it is tempting to proclaim the superimposing idea a success since the U-trained models perform significantly worse, while the models trained on all datasets have a far more 'reasonable' drop in performance. However, such a conclusion assumes that the models which perform well on superimposed data will generalise to real-world data, something we can only evaluate through qualitative tests.

Extending our gaze to the full binary results in Table 7.1, we will first highlight how both All-trained classifiers are able to outperform the other checkpoints in all test datasets except the U dataset where the results are far more even. This means that models trained on the two superimposed datasets are able to generalise to both domains. Moreover, the tendencies of Table 7.2 remain consistent with the results in table 7.1.

We should also point out how the U-trained P-CLAS model fares significantly worse than its D-CLAS comparable when considering the TPR metric on the superimposed datasets. This can be taken as a sign that the superimposing approach is not sufficiently realistic to be recognisable to this classifier. If this was the case we would expect both classifiers to be significantly affected by lower TPR scores. While the U-trained D-CLAS model clearly struggles with TPR, we are not entirely convinced by this explanation. Our hypothesis is that the P-CLAS model has been so conditioned on white dots on untextured backgrounds from training on the U dataset, that 'marine snow' keypoints that deviate from this in the superimposed datasets become a large source of incorrect classifications. D-CLAS on the other

hand may have some benefits in its ORB-encoding which alleviates some of these issues. In the end, we will rely on qualitative tests to try to find a conclusion to this debate.

As a final note, we found that D-CLAS exceeds speeds of 66000 keypoints per second, compared to 14600 for P-CLAS on a GTX 1080 GPU. This partly comes down to pre-processing which D-CLAS avoids, while P-CLAS requires patch extraction. By adapting the multi-scale approach, these results could come closer together.

7.1.2 Comparing Classifiers for Descriptors

Table 7.3 lists the results of scikit's MLP (a synonym for FCNN), LinearSVC, GaussianNB, QDA, and Deciscion Tree classifiers, alongside our D-CLAS architecture.

	F1	Acc	TPR	TNR
MLP	0.921	0.926	**0.991**	0.876
LinearSVC	0.933	0.940	0.968	0.918
GaussianNB	0.836	0.838	0.946	0.756
QDA	0.852	0.859	0.936	0.799
Decision Tree	0.832	0.847	0.874	0.829
Ours(D-CLAS)	**0.945**	**0.951**	0.980	**0.929**

Table 7.3: Comparison of D-CLAS with five scikit classifiers

These results motivated our initial decision to pursue an FCNN approach to descriptor classification. As we can see, D-CLAS presents a substantial improvement of F1 score and accuracy compared to the other models. It is only beat on the TPR metric where MLP exceeds it. Since MLP is another term for FCNN, we can assert with confidence that the ORB descriptors are particularly well suited for classification with neural networks. It should also be stated that the scikit models were significantly slower to train than D-CLAS.

7.1.3 Comparing Descriptors

Our motivation behind comparing different descriptors was to see how the encodings used by various descriptors affected the descriptor classifier's ability to classify keypoints. Four different frequently used descriptors were tested—ORB, FREAK, SIFT, and VGG120—using default settings in OpenCV to generate the descriptors. In Table 7.4, the dataset and descriptor used for training can be seen by the rows, while the data used for testing can be seen by the columns.

		U				All datasets			
		F1	Acc	TPR	TNR	F1	Acc	TPR	TNR
ORB	U	0.945	0.95	0.984	0.923	0.77	0.821	0.719	0.893
	All	0.953	0.959	0.977	0.945	**0.926**	**0.937**	**0.94**	0.935
FREAK	U	**0.981**	**0.984**	0.992	**0.979**	0.61	0.784	0.463	**0.968**
	All	0.916	0.925	**0.996**	0.876	0.792	0.847	0.796	0.877
SIFT	U	0.935	0.942	0.965	0.925	0.466	0.684	0.33	0.938
	All	0.851	0.855	0.949	0.784	0.8	0.826	0.831	0.822
VGG	U	0.978	0.981	0.988	0.975	0.573	0.738	0.422	0.963
	All	0.928	0.935	0.976	0.903	0.884	0.902	0.902	0.902

Table 7.4: Binary classification results with different descriptors. Rows denote the descriptor and its training data, while columns denote the test dataset. The Unmodified dataset (U), Underwater superimposed (UW), and Overwater superimposed (OW) were combined and used for testing. We provide F1 scores, accuracy, True Positive Rates (TPR), and True Negative Rates (TNR) for each case.

First of all, FREAK does better than the other descriptors on the U dataset by a relatively large margin, but falls behind ORB on the All-testset. Among the descriptors, ORB does the best across all remaining dataset combinations, including ones from the full table (Table B.1) included in the appendices. Interestingly, the U-trained ORB model is far more capable of generalising to the All-testset than the other U-trained architectures.

On the F1 metric, the U-trained FREAK model drops by 0.189 between the U-testset and All-testset, compared to 0.019 for ORB. Notably, on the TNR metric, U-trained FREAK scores better than all other descriptors, even its All-trained twin. In fact, the ORB architectures are the only ones to deviate from the observation that U-trained models have better TPR than All-trained ones. The same is true for all metrics if we limit ourselves to the U testset. We are unable to explain why this is, but perhaps it is connected to why ORB does better overall.

Briefly extending our analysis to the other two descriptors, we find that SIFT does particularly poorly on the All-testset, while VGG achieves the second-best results overall.

7.2 Qualitative Results in Keypoint Classification

In this section, we describe the qualitative results of P-CLAS and D-CLAS on six video sequences.

V1: Rocky background with dense snow, both brightly lit.

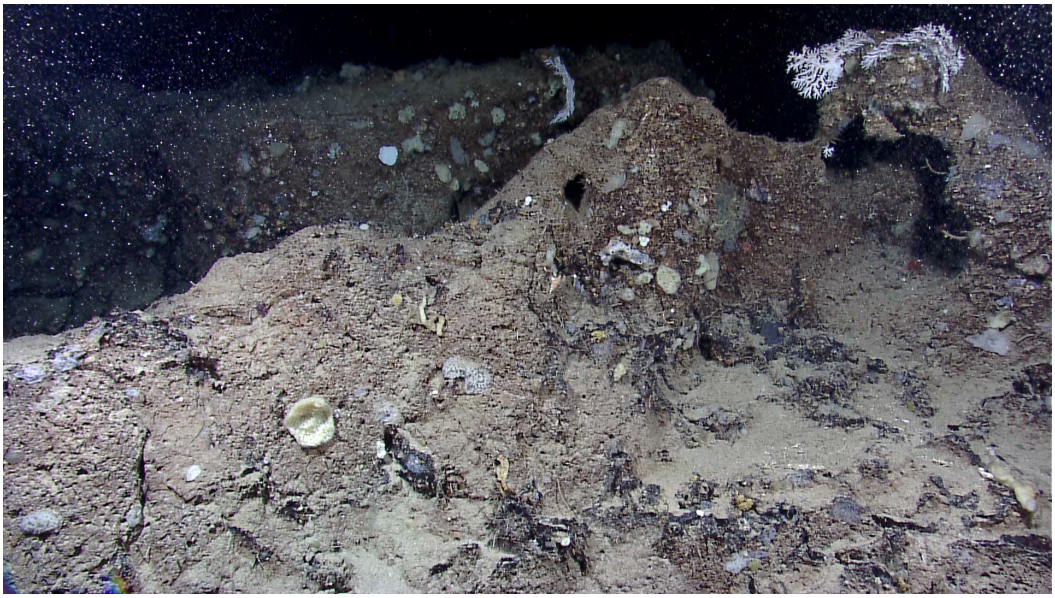


Figure 7.1: A frame from video 1.

Figure 7.1 shows a frame from video 1 which features a series of rocky crests, one prominently in the front of the picture, and one further into the shadows on the top left. The marine snow is particularly visible in front of the shaded rock, which can present a challenge for the classifiers, since the textured background is visible through the shadows. A playlist of the classification results on this video is found at `www.youtube.com/watch?v=-uryM93ozTc&list=PLWEGWR7qQmlVQwQwrdSNvsLCu4EjI82Cg`.

Beginning with U-trained P-CLAS the issue of textured backgrounds is highlighted immediately as it classifies *every* keypoint as clean, despite a significant amount of keypoints being placed in the snowy, upper left region. This is precisely the kind of behaviour we wanted to avoid using our superimposing approach. OW+U-trained P-CLAS improves slightly, but still classifies a majority of the snowy keypoints as clean, while UW+U- and All-trained P-CLAS seem to correctly classify around 50–60% of the marine snow. None of the models had any issue correctly classifying good keypoints, which is unsurprising given the TNR rates observed in the quantitative tests.

The D-CLAS models fare significantly better than P-CLAS on this sequence. Beginning with U-trained D-CLAS, around 70–80% of the keypoints in the challenging upper-left region are classified correctly, though a handful of false positives are introduced on the foremost rock. These results are notable given that this model has not been trained on any superimposed data, and suggests that the ORB

encoding represents keypoints in such a way that marine snow descriptors can remain similar despite differences in their keypoints' backgrounds. This could possibly explain why U-trained D-CLAS did better than U-trained P-CLAS on the All-testset in our quantitative tests.

OW+U-trained D-CLAS performed similarly to the U-trained variant, while the UW+U trained model improved on both TPR rates and the already high TNR. Curiously, the All-trained model did worse than the others, approximately matching the best performance seen by P-CLAS models, with around 50–60% correct keypoints in the upper left corner.

V2: Large yellow structure, with very large dense snow.

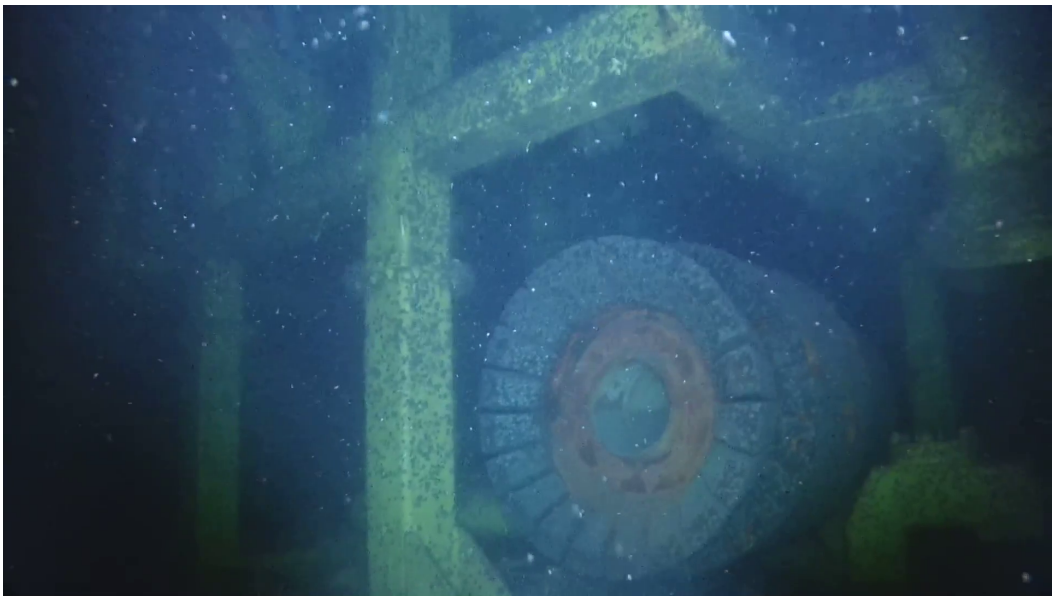


Figure 7.2: A frame from video 2.

Figure 7.2 shows a frame from video 2 which features a large yellow structure supporting a pipe, engulfed by dense marine snow. A playlist of the classification results on this video is found at `www.youtube.com/watch?v=RMyoeGvfmhQ&list=PLWEGWR7qQmlX_Q2KC7BCryX8RekE4FipK`.

U-trained P-CLAS achieves near perfect accuracy on marine snow keypoints in this sequence, only presenting occasional false negatives when the snow moves in front of the yellow beams and the brown, inner section of the pipe. The amount of false negatives was lower than we had expected, given the lack of coloured backgrounds in the U training data. Most marine snow is correctly classified as it traverses the region in front of the pipe. However, at the start of the sequence,

the camera is close enough to the structure to capture a gray, spotted pattern on the outer border of the pipe. All keypoints placed on this pattern were incorrectly classified as marine snow.

While UW+U-trained P-CLAS does slightly better on the spotted pipe section, OW+U model improves markedly, getting around 60–70% of the keypoints on the pipe correctly labelled as clean. All-trained P-CLAS improves this even further, but has a far larger presence of false negatives elsewhere in the sequence not seen with the other models.

U-trained D-CLAS does better than all P-CLAS models on the deceptive spotted pipe, although it has slightly elevated levels of false negatives, though not nearly the extent of All-trained P-CLAS. To our eyes, all other D-CLAS models perform similarly to the aforementioned U-trained architecture, with perhaps modestly better TNR rates.

V3: Dark brown sand and rocks with moderate snow under dim, yellow lighting.

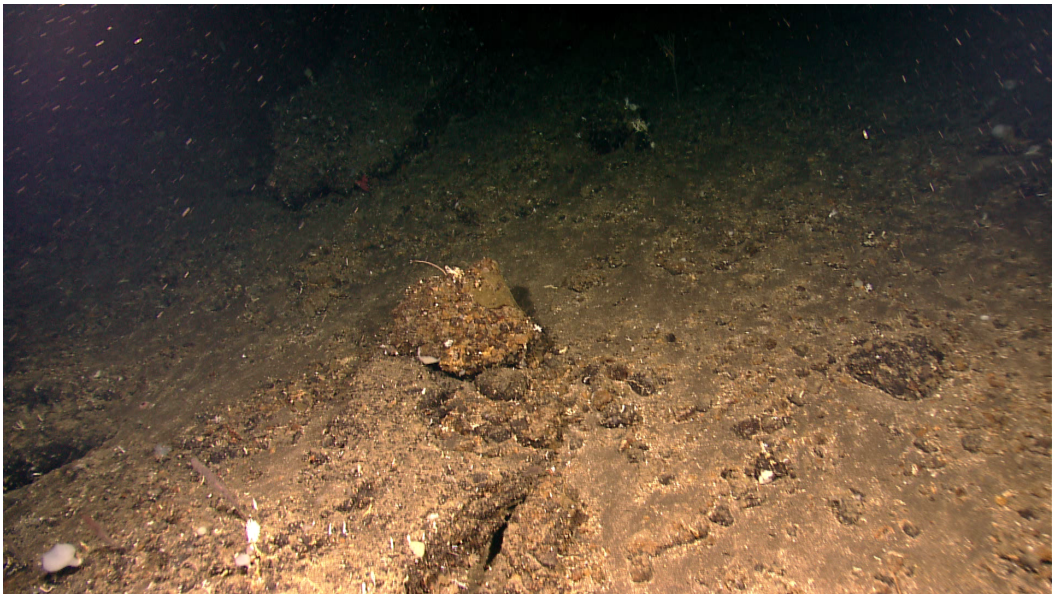


Figure 7.3: A frame from video 3.

Figure 7.3 shows a frame from video 3 which features a brown environment of sand and rock, with low to moderate levels of marine snow. Due to the angle of the robot (which moves relatively far throughout the scene), the marine snow appears bright over the shaded ocean floor which provides a richly textured backdrop for the marine snow keypoints. Notably, compared to the other sequences,

a relatively small amount of the keypoints are actually placed on the snow. A playlist of the classification results on this video is found at www.youtube.com/watch?v=nmNexTjtxkU&list=PLWEGWR7qQmlVyt3NTvr5RcAK6BQZaeOi8.

U-trained P-CLAS sticks to its behaviour from Video 1, where it classifies every keypoint as clean. Overall, performance from P-CLAS is lacking on this sequence, with the main change brought by the other models being an increase in false positives on the ground. While all of the other models occasionally classify the snow correctly, only the UW+U trained model is able to meaningfully improve the snow classification results, achieving between 30–50% correct classifications on the snow as the sequence progresses.

U-trained D-CLAS does significantly better at classifying the snow in the video, easily getting more than 90% correct, however at the cost of a clear increase in false positives on the ground. The models trained on superimposed data improve upon this, of which All-trained D-CLAS does the best. Although it still has worse false positive rates than P-CLAS models.

V4: Moves from a close up of a yellow structure. Extremely dense and large marine snow.

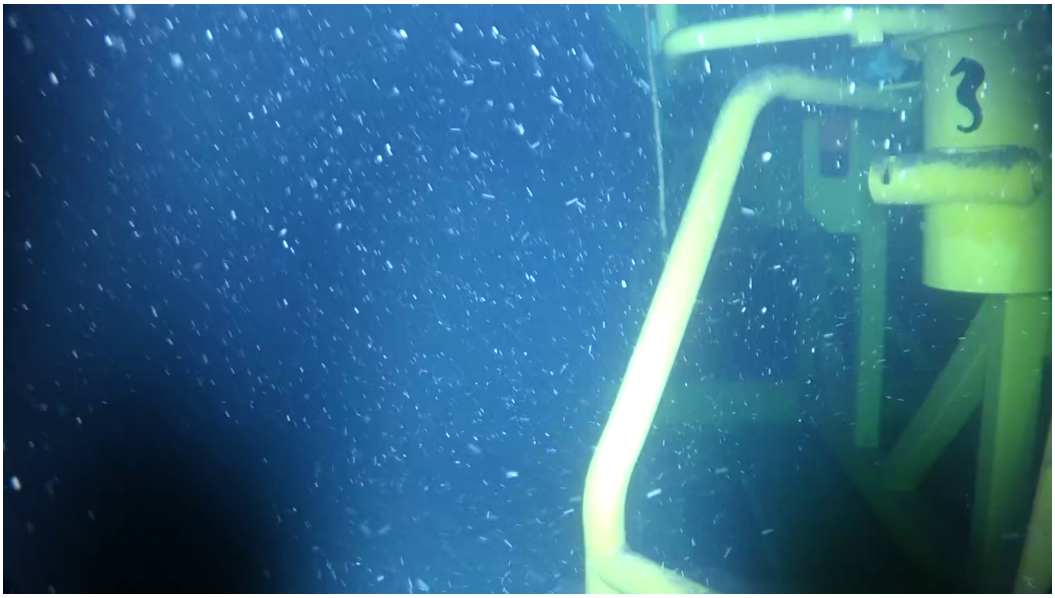


Figure 7.4: A frame from video 4.

Figure 7.4 shows a frame from video 4 which moves from an extreme close up of a yellow charging structure, back to the wide view seen in the figure. The video features extreme amounts of very large marine snow; we have not found

similar conditions in any other sequences. A playlist of the classification results on this video is found at www.youtube.com/watch?v=9lwEYTOttY8&list=PLWEGWR7qQmlVtLx8SNw97ZBv7yR5Ox_2y.

U-trained P-CLAS does very well, probably because most snowy keypoints in this sequence match the flat backgrounds seen in its training data. However, snowy keypoints are often classified incorrectly when they pass the shaded yellow beams in the distance. This is also true for the start of the sequence, where a section of gray marine snow on a darker-gray background is labelled as clean around 90% of the time. OW+U and UW+U trained P-CLAS improve upon the results in the gray region at the start of the sequence, classifying around 50–60% and 70–80% of the keypoints correctly as snow, respectively. However, both models still struggle with marine snow in front of the yellow beams later in the sequence. All-trained P-CLAS does the worst, with significantly higher levels of false-negatives than the other models, even in regions with flat backgrounds.

U-trained D-CLAS maintains similar performance to the best P-CLAS models, perhaps with slightly elevated false negative rates. The same can be said for the other D-CLAS models, although All-trained D-CLAS has higher false negative rates than the rest, almost to the same level as All-trained P-CLAS.

V5: Flat ocean bed in the distance, with a dense cover of small, fast-moving marine snow.

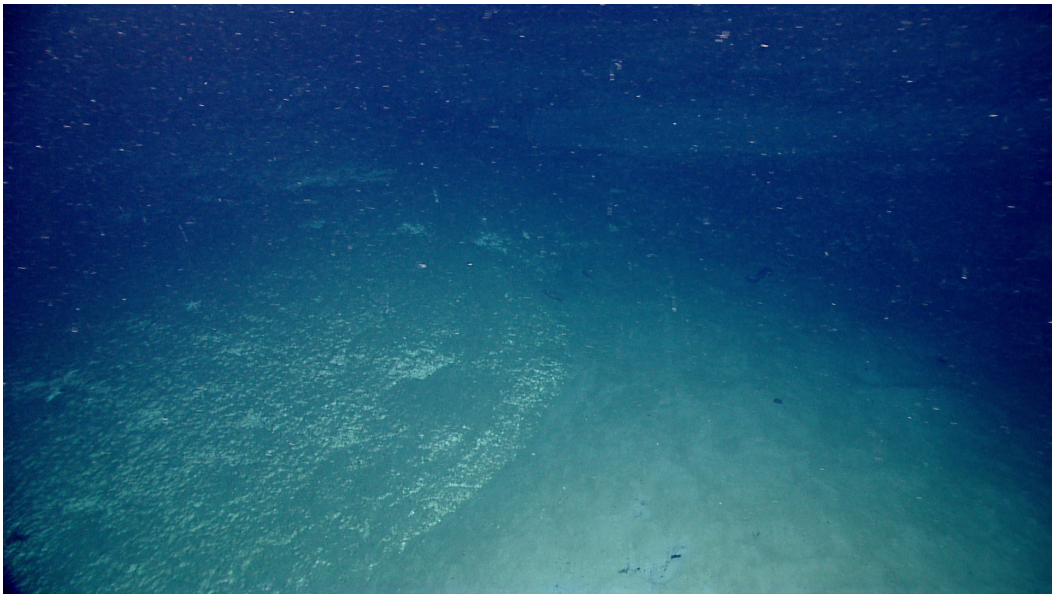


Figure 7.5: A frame from video 5.

Figure 7.5 shows a frame from video 5 which features a flat ocean floor slightly in the distance, behind a dense sheet of small marine snow particles moving quickly from left to right. The results are uploaded here: `www.youtube.com/watch?v=ZQNr05qZagc&list=PLWEGWR7qQmlVV2x0p-57UaySTGVXyIKCE`.

U-trained P-CLAS presents no false positives that we could tell, and a modest amount of false negatives, mainly along the vertical center of the image. UW+U and OW+U models improved TPR rates even further, correctly classifying marine snow well into the bottom half of the image which has a more textured backdrop. This comes at the cost of marginally elevated false positive rates. Meanwhile, the All-trained model has greater levels of false negatives than all other models.

U-trained D-CLAS achieves similar results on the marine snow as the best P-CLAS models, although it has meaningfully elevated levels of false positives on the white texture on the ocean floor. D-CLAS models trained on superimposed datasets improve slightly on this issue, with the All-trained D-CLAS model garnering the lowest false positive rates.

V6: Black and white colours, silty ground with out of focus marine snow.

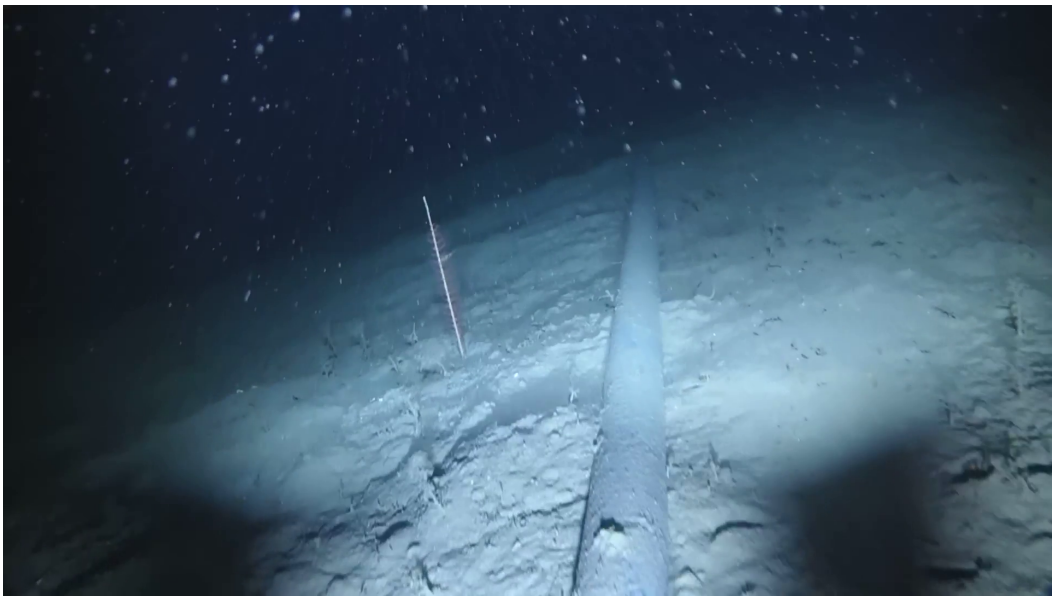


Figure 7.6: A frame from video 6.

Figure 7.6 shows a frame from video 6 which features the robot travelling close to a small gray pipe on the silty ocean floor. The marine snow is almost motionless as the robot moves quickly troughs it. In contrast with previous examples, the ma-

rine snow comes rapidly in and out of focus and illumination as the robot passes. The results are uploaded here: www.youtube.com/watch?v=2fai7oGJCx4&list=PLWEGWR7qQmlWcyxzagFroqY6jc_GA_T3O.

U-trained P-CLAS does reasonably well on this sequence, and on any given frame correctly classifies most snow, except the bottom-most snow which appears in front of the seabed. At one point in the sequence, the robot does a low pass over the terrain, at which point around 20–30% of the keypoints on the ground are classified incorrectly as marine snow. Other than this, false positives are few but consistent. The OW+U trained model has significantly fewer false positives during the low pass, but slightly more false positives once the manoeuvre is over. The UW+U network did worse during the low-pass, while the All-trained model had fewer false positives than the others, but at the same time had significantly higher false negative rates as well.

The U-trained D-CLAS network had near perfect accuracy on the clean keypoints on the ground, and was unaffected by the close encounter with the ocean floor. However, false negative rates were somewhat increased compared to P-CLAS models, with the exception of All-trained P-CLAS. The remaining D-CLAS models all performed similarly to this standard.

7.2.1 Summarising the Results on Videos 1–6

To summarise the results from the above sections, we will first begin to highlight the performance of U-trained models, which were generally able to set competitive baselines for the other models to beat, with the only exceptions being some sequences where U-trained P-CLAS classified *all* keypoint as clean. In such sequences, training P-CLAS on superimposed data improved performance on the incorrectly classified keypoints. Importantly, P-CLAS's performance was far more reliant on superimposed training data than D-CLAS which did well on textured backgrounds, even with its U-trained checkpoint.

Another thing to note was the frequent reduction in performance when training on both superimposed datasets as opposed to only one of them. This was observed with both architectures. Although training on the All-dataset sometimes reduced false positive rates, false negative rates were frequently significantly higher than other models.

D-CLAS was also more competitive with P-CLAS than in the quantitative tests, outperforming P-CLAS on at least two sequences. The fact that U-trained D-CLAS had improved performance on textured backgrounds compared to U-trained P-CLAS seems to indicate that the ORB encoding may process information about the background in a way which allows D-CLAS to generalise more easily between marine snow on textured and untextured backgrounds. Why D-CLAS often has more false positives than P-CLAS is difficult to assert, but it could be worth exploring if ORB's use of grayscale imagery contributes to this difference. Lastly, it should be highlighted how D-CLAS models, on multiple occasions, performed at a similarly high level, regardless of training data.

7.3 Qualitative Results with pySLAM

This section details our qualitative results with pySLAM on a collection of real underwater sequences with differing levels of marine snow density, and on the VAROS and Snowy-VAROS datasets. The experiments were conducted with UW+U checkpoints for both P-CLAS and D-CLAS.

7.3.1 Sequence with Exclusively Snow

Figure 7.7: A frame from the video with exclusively snow

When running pySLAM without our keypoint rejection networks on a sequence with exclusively snow (see Figure 7.7), pySLAM both initialises and attempts to track pose based on the motion of the snow. Tracking continues for 150 frames, until pySLAM eventually shuts down. This tracking duration is relatively long compared to similar sequences we have evaluated. When enabling marine snow rejection, the system never initialises due to a lack of consistent keypoint matches. On average D-CLAS marked 1860 out of 1966 average keypoint detections as marine snow, while P-CLAS removed 1789 out of 1873 keypoints. When running the system without keypoint classification, the average frame timing on this sequence is 0.14 seconds while with D-CLAS it is 0.23 seconds—an increase of 0.09 seconds—and with P-CLAS it is 0.52, or an increase of 0.38 seconds.

Figure 7.8: A frame from the heavy snow sequence

7.3.2 Sequence with Heavy Snow

On the sequence pictured in Figure 7.8 with heavy snow and no marine snow rejection, pySLAM gathers a sufficient amount of keypoints on the ground to initialise tracking and estimate poses. However, a lot of snow correspondences make it through the standard outlier rejection scheme and are added to the map. Additionally, the tracking is highly unstable as can be seen in Fig. 7.9: when repeatedly running pySLAM, the initialisation and tracking can fail multiple times in a row, meaning we have to restart the system with a new random seed to hopefully make it track properly. Since we do not have ground truths for these real-world sequences, we cannot evaluate the correctness of the point clouds in Fig. 7.9 or the point clouds generated with keypoint classification.

When we enable either classifier, pySLAM's initialisation struggles disappear; pySLAM successfully initialises tracking on every run, and very little snow is tracked or added to the map. The marine snow keypoints that the networks fail to remove are usually located on a non flat background or are very large. There was a visible difference in the number of keypoints rejected by P-CLAS and D-CLAS, with P-CLAS removing close to all marine snow keypoints and D-CLAS removing noticeably fewer keypoints. The keypoints that P-CLAS and D-CLAS classified incorrectly were generally removed by pySLAM's inherent outlier rejection scheme.

On average 294 keypoints were filtered using D-CLAS, and 657 using P-CLAS from an average 1455 detections. Based on pySLAM's visualisations, we have

a clear impression that P-CLAS was better than D-CLAS at removing the snow. However, because we do not have ground truths, we can not be sure whether or not this was associated with an increase in false positives.

When running the system without keypoint classification, the average frame timing on this sequence is 0.97 seconds, while with D-CLAS it is 1.19 seconds and with P-CLAS it is 3.62 seconds. Despite classifying a smaller amount of keypoints than on the sequence with exclusively snow, the cycle times have increased significantly with both classifiers. For D-CLAS the increase in cycle times was 0.22 seconds, or 0.13 seconds more than the previous sequence. P-CLAS increased pySLAM's cycle times by 2.65 seconds, or 2.27 seconds more than the earlier sequence.

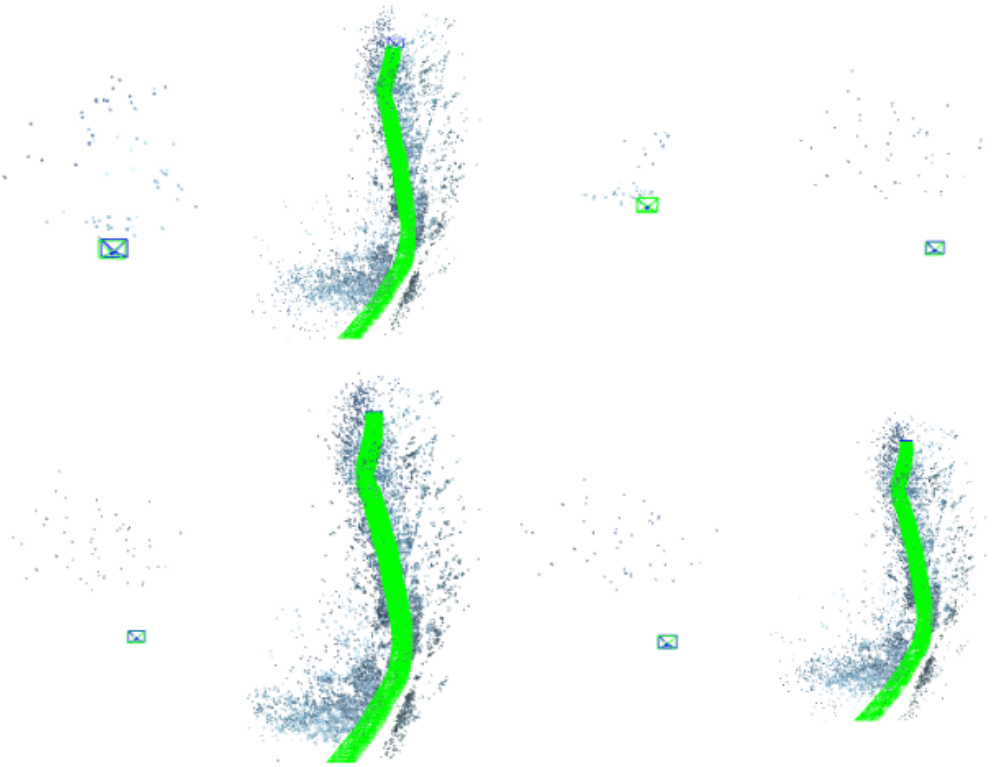


Figure 7.9: Point clouds from 8 consecutive runs of pySLAM on the heavy snow sequence, without keypoint classification. Out of the 8 runs, 5 initialise on the snow and fail to track, while 3 of the runs have better initialisation and are able to continue tracking.

In other sequences with particularly heavy snow, like Video 4 in Section 7.2, the vast majority of keypoints land on snow. Even though our methods are able to remove most of them, there are too few remaining keypoints to initialise tracking

and estimate poses. From the screenshot of Video 4 in Figure 7.4, this issue may be surprising, since the yellow structure appears as a very clear landmark to track on. However, many keypoint detectors, including ORB, are unable to detect keypoints on most of the yellow structure. This is because parts of it are too far away from the camera and become blurry from the water's turbidity, and because the yellow structure has so few defined corners to detect.

7.3.3 Sequence with Light Snow

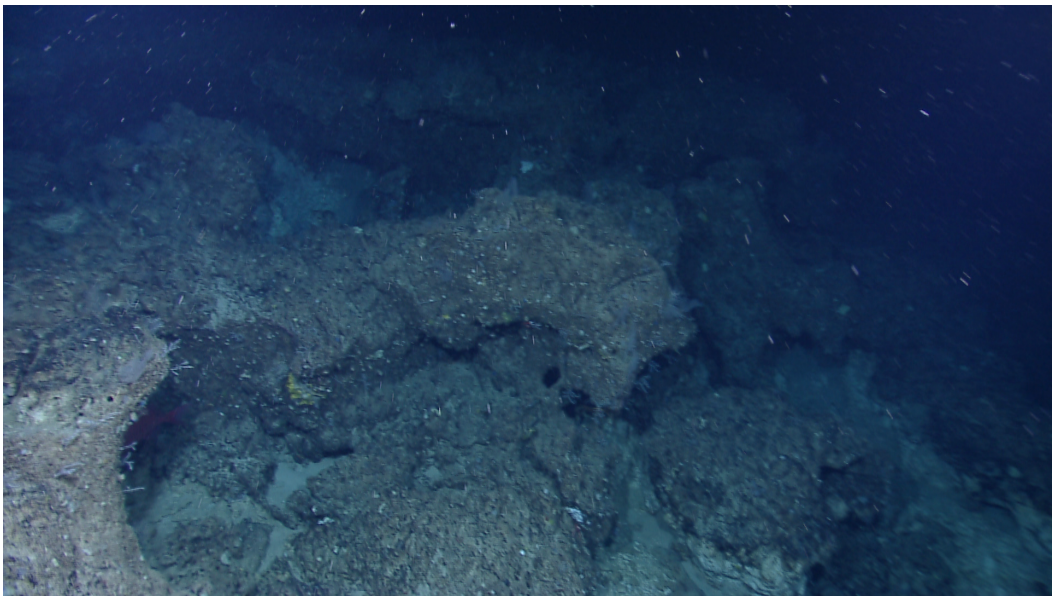


Figure 7.10: A frame from the sequence with light marine snow conditions

On the light snow sequence seen in Figure 7.10, the ORB detector continues to detect keypoints on the the marine snow. However, the outlier rejection schemes implemented in pySLAM are capable of removing these outliers, with seemingly no marine snow being added to the map.

When running with keypoint classifiers, the results are similar. Both P-CLAS and D-CLAS manage to remove keypoints on marine snow without affecting the clean keypoints in any meaningful way, and the generated point clouds exhibit the same general structure. With that said, the classifiers still differ in the amount of keypoints removed; out of 2000 keypoints, P-CLAS eliminated 403 keypoints and D-CLAS eliminated 211 keypoints. Without keypoint classification, the average frame took 2.89 seconds to process, with D-CLAS it took 3.19 seconds, and with P-CLAS it was 6.85 seconds.

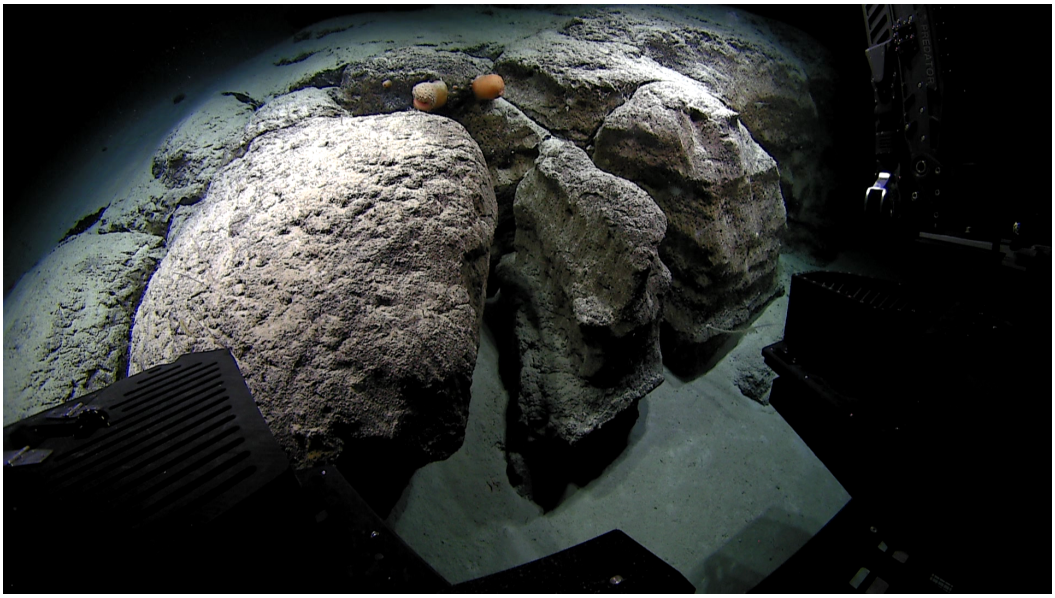


Figure 7.11: A frame from the sequence with no marine snow

7.3.4 Sequence with No Snow

When running pySLAM on the sequence with no snow pictured in Figure 7.11, D-CLAS only marked on average 22 out of 2000 keypoints as marine snow and P-CLAS marked 52 points out of 2000. This had no meaningful impact on the pose estimates or the point cloud generated when compared to running without keypoint classification. When running the system without keypoint classification, the average frame timing on this sequence is 0.94 seconds while with D-CLAS it is 1.25 seconds and with P-CLAS it is 1.83 seconds.

7.3.5 Snowy-VAROS Sequence

Evaluating SLAM performance on real underwater footage has the potential to give the most realistic view of the effect of snow classification. However, with the synthetic VAROS sequence and Snowy-VAROS, we are able to control the difficulty of both the background sequence and snow conditions by selecting which snow to superimpose and onto which section of VAROS. Furthermore, we can compare pySLAM's performance in VAROS and Snowy-VAROS which can highlight the effect of keypoint rejection more definitively than our typical qualitative tests. However, we should be aware that classifiers trained on superimposed images can perform disproportionately better on Snowy-VAROS, because of similarities between the superimposing process of Snowy-VAROS and the training data.

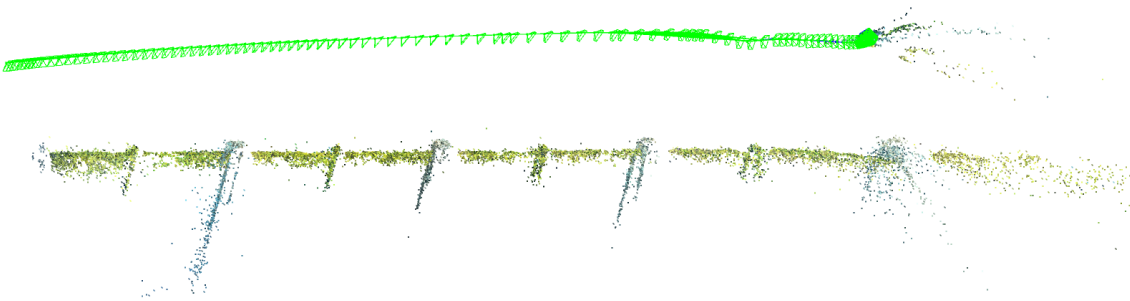


Figure 7.12: A point map from pySLAM in VAROS without snow and no classification. The pipe is properly tracked. Figure from [Hodne, Leikvoll, et al. 2022].

We use a subsequence of VAROS in which the robot travels adjacent to a straight pipe (see Fig. 5.2). This pipe offers more defined features for keypoint detection compared to other sections of VAROS. We use an Eelume sequence from our test data for superimposing. Its density and character of marine snow is comparable to Video 4 of the qualitative test and is well within the heavy snow category.

While VAROS, and by extension Snowy-VAROS, have ground truths which we can use to evaluate the poses given by pySLAM, we did not manage to do so because pySLAM mainly used relative position (without orientation) and made some assumptions on the initial orientation which were difficult to account for. However, by travelling along the pipe section in VAROS, it should be possible to coarsely judge the tracking quality by how accurately the straight pipe is mapped. Consequently, our goal is to match the point cloud in Figure 7.12, which was made with pySLAM on the VAROS dataset without superimposed marine snow, and without snow classification.

When testing with pySLAM only on Snowy-VAROS, a large number of keypoints are detected on the superimposed marine snow, which leads to rapid tracking failure, coupled with inconsistent behaviour between separate runs, as was observed on the earlier heavy snow sequence. During some tests, tracking fails completely, while in other runs tracking is closer to the movement from the source video. Visualising the sparse map from pySLAM in Figure 7.13, we find what looks like a wall of marine snow prominently on the right. Furthermore, the pipe which is straight in the sequence appears bent in the point cloud.

Both P-CLAS and D-CLAS facilitated stable tracking in pySLAM, to the extent that they were hard to tell apart. Although they could not remove all unreliable points, pySLAM could track for far longer while being more consistent between runs. A point cloud from pySLAM with snow classification in Snowy VAROS is seen in Figure 7.14 where the pipe appears straight, with the exception of the very end. This is similar to the behaviour from Figure 7.12 with pySLAM in VAROS without both marine snow and snow classification. From pySLAM's visualisations, we can tell that this behaviour occurs because pySLAM is unable to detect a sufficient amount of good keypoints, irrespective of the presence of snow.

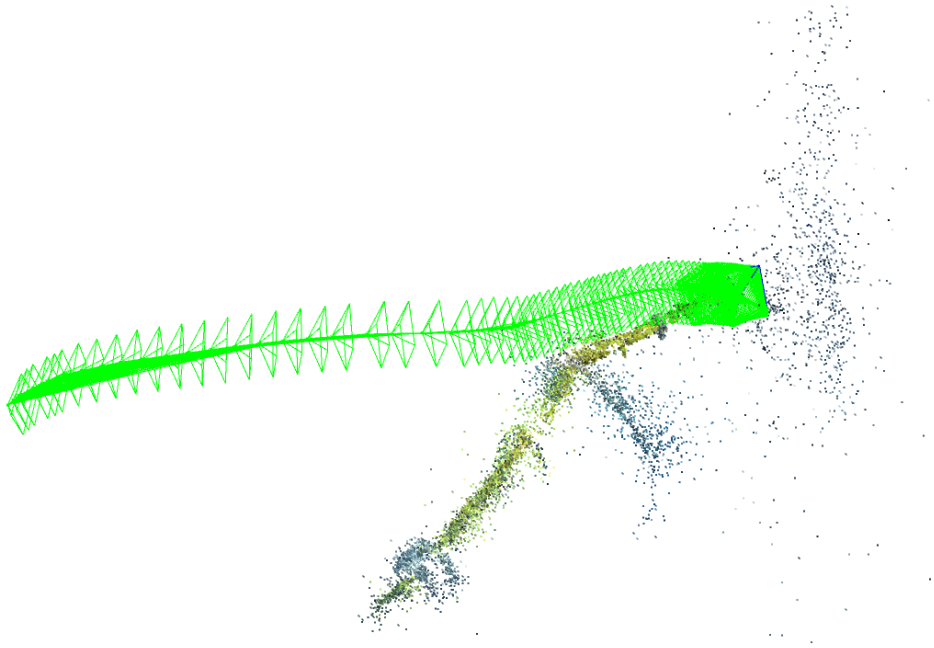


Figure 7.13: A point cloud from pySLAM without keypoint classification in Snowy-VAROS. The tracking fails rapidly in a wall of snow and the straight pipe appears bent, unlike the tracked video. Figure from [Hodne, Leikvoll, et al. 2022].

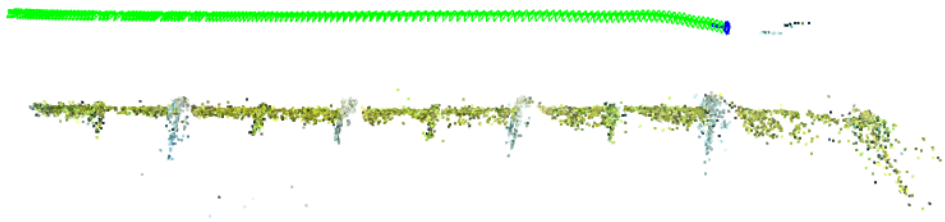


Figure 7.14: A point cloud from pySLAM with keypoint classification in Snowy-VAROS. The pipe is seemingly tracked correctly, and few snow keypoints are included in the map. Figure from [Hodne, Leikvoll, et al. 2022].

On Snowy-VAROS, out of 3000 features, D-CLAS removed on average 1,627 keypoints and P-CLAS removed 1,366 keypoints. For comparison, we ran pySLAM on the original VAROS which has nothing resembling marine snow. On VAROS, out of 3,000 keypoints, on average 36 and 201 rejections, *i.e.*, false positives, are made by D-CLAS and P-CLAS, respectively.

7.3.6 Summarising the Results with pySLAM

In our experiments with pySLAM, we have tried to understand the impact of marine snow on SLAM with and without our keypoint classification methods.

When running without keypoint detection, pySLAM's outlier rejection scheme (ratio testing and RANSAC) was able to ignore some marine snow. However, it was only on the light marine snow sequence that a sufficient amount of marine snow correspondences were removed to reliably use pySLAM. On Snowy-VAROS and the sequences with heavy marine snow, pySLAM struggles to initialise, and on the occasions which it does, pySLAM initialises with correspondences on marine snow keypoints which impacted the motion hypotheses and typically lead to rapid failure of the tracking.

Introducing either P-CLAS or D-CLAS seems to have equal effect on pySLAM's tracking. In Snowy-VAROS, the results with both classifiers were comparable to pySLAM's standalone results on the original VAROS sequence without marine snow. On the heavy sequence pySLAM's stability increases significantly, while pySLAM's tracking on the light sequence is not influenced by the classifiers.

The main weakness observed is the significant computational cost which is imposed by P-CLAS, although we are unsure of how well this generalises to other systems. On the heavy sequence, both P-CLAS and D-CLAS were reasonably efficient at classifying approximately 1800 keypoints, spending 0.38 and 0.09 seconds, respectively. However, as other sequences were introduced, the additional time pySLAM used with the classifiers continued to rise. For example, on the sequence with light snow, the computational cost of D-CLAS and P-CLAS was measured in seconds—multiple in the case of P-CLAS. This leads us to believe that the cycle times are influenced by how pySLAM allocates compute resources under various workloads. While the results give an important insight into how pySLAM behaves with our models, the timings can not be considered representative of the actual duration spent by the networks to classify keypoints.

7.4 Ablation Study

This section contains our ablation study, beginning with an evaluation of single- vs multi-scale P-CLAS, and continuing with results on colour versus greyscale and our snow proximity filtering and variance-based visibility filtering.

7.4.1 Comparing Single and Multi Scale Patch Classifiers

Table 7.5 lists the binary classification metrics of the standard Multi-Scale (MS) P-CLAS model (meaning the same results as in Table 7.1), and the Single-Scale (SS) P-CLAS implementation.

		Unmodified				UW + U				OW + U				All datasets			
		F1	Acc	TPR	TNR	F1	Acc	TPR	TNR	F1	Acc	TPR	TNR	F1	Acc	TPR	TNR
Multi	U	0.999	0.999	0.999	**1.0**	0.617	0.792	0.446	0.999	0.47	0.687	0.309	0.997	0.402	0.692	0.252	0.998
	UW + U	0.968	0.972	0.991	0.957	0.948	0.961	0.931	0.98	0.854	0.861	0.904	0.825	0.877	0.897	0.894	0.9
	OW + U	0.998	0.998	0.996	0.999	0.913	0.94	0.84	**0.999**	0.94	0.948	0.894	0.993	0.91	0.932	0.839	0.996
	All	0.996	0.996	0.996	0.997	**0.975**	**0.982**	0.955	0.998	0.961	0.965	0.954	0.975	**0.964**	**0.971**	0.95	0.985
Single	U	**1.0**	**1.0**	**1.0**	**1.0**	0.589	0.781	0.418	1.0	0.463	0.685	0.302	**0.999**	0.373	0.684	0.23	**0.999**
	U+UW	0.99	0.991	0.987	0.994	0.916	0.941	0.851	0.996	0.802	0.828	0.775	0.872	0.823	0.864	0.772	0.928
	U+OW	0.998	0.998	1.0	0.997	0.922	0.946	0.857	**0.999**	**0.965**	**0.969**	0.95	0.985	0.935	0.949	0.888	0.992
	All	0.966	0.969	0.995	0.949	0.962	0.971	**0.981**	0.965	0.949	0.953	**0.965**	0.944	0.956	0.963	**0.969**	0.959

Table 7.5: Binary classification results by the multi-scale P-CLAS compared to a single scale version. Rows denote the network and its training data, while columns denote the test dataset. The Unmodified dataset (U), Underwater superimposed (UW), and Overwater superimposed (OW) were combined and used for testing. We provide F1 scores, accuracy, True Positive Rates (TPR), and True Negative Rates (TNR) for each case.

Overall, the two P-CLAS models perform more similar than we expected. On the unmodified dataset, the SS classifier achieves near-perfect results, only classifying some tens of samples incorrectly. This tells us that the original results of P-CLAS on the U dataset were not due to its multi-scale design. Therefore, the SS network actually improves performance on the U dataset, possibly because it has to model a less highly dimensional function than the MS network.

The All-trained SS implementation consistently outperforms all other architectures on the TPR metric, although its TNR is consistently below average. The TPR metric is particularly important because it denotes the share of snow samples which are correctly labelled snow. Since the SS models always did better on this metric, the benefit of MS seems dubious, at least for our CNN architecture. This is covered further in our discussion.

The All-trained SS model's lowered TNR is interesting because it suggests that single-scale 'clean' patches are more easily mistaken for single-scale 'snow'. However, the reduced TNR does not manifest itself in the U+UW and U+OW trained single-scale models, which both present comparable and even improved TNR compared to their multi-scale counterparts.

Looking at the F1 score and accuracy, the All-trained MS models outperform the other models with around a margin of 0.01 or more on the UW+U and All testsets, while being a close second on the others. This could indicate that MS implementations offer a better trade-off between precision and recall. However, if that is the case, it should be questioned if it is worth the additional computational burden of triple patch extraction and resizing. When examining the computational cost of multi-scale patches, we found that our three-scale solution uses on average 0.106 seconds to extract 2000 sets of multi-scale patches from one image, compared to

0.0289s for single-scale, and 0.0646s for two-scale. At the very least, we would consider a 2-scale approach, *e.g.* 64×64 and 32×32.

7.4.2 Comparing Colour and Grayscale

In our literature review, we found methods that relied on pixel saturation to determine the location of snow[Farhadifard, Radolko, and U. v. Lukas 2017; Boguslaw Cyganek and Gongola 2018]. We wanted to test if our CNN classifiers were able to make use of this heuristic, so we trained P-CLAS networks using grayscale and RGB input. Moreover, many descriptors—including ORB—are based on grayscale images. Consequently, these results can highlight the significance of this difference, and if D-CLAS may perform worse because of ORB's use of grayscale imagery (hereby, denoted GS). The results are listed in table 7.6.

		Unmodified				UW + U				OW + U				All datasets			
		F1	Acc	TPR	TNR	F1	Acc	TPR	TNR	F1	Acc	TPR	TNR	F1	Acc	TPR	TNR
RGB	U	**0.999**	**0.999**	**0.999**	**1.0**	0.617	0.792	0.446	0.999	0.47	0.687	0.309	**0.997**	0.402	0.692	0.252	**0.998**
	UW + U	0.968	0.972	0.991	0.957	0.948	0.961	0.931	0.98	0.854	0.861	0.904	0.825	0.877	0.897	0.894	0.9
	OW + U	0.998	0.998	0.996	0.999	0.913	0.94	0.84	0.999	0.94	0.948	0.894	0.993	0.91	0.932	0.839	0.996
	All	0.996	0.996	0.996	0.997	**0.975**	**0.982**	**0.955**	0.998	**0.961**	**0.965**	**0.954**	0.975	**0.964**	**0.971**	**0.95**	0.985
Grayscale	U	0.998	0.998	**0.999**	0.998	0.671	0.814	0.505	0.999	0.495	0.685	0.343	0.965	0.455	0.702	0.304	0.978
	UW+U	0.948	0.957	0.903	0.998	0.747	0.848	0.598	0.999	0.67	0.763	0.534	0.951	0.646	0.777	0.496	0.972
	OW+U	0.147	0.6	0.079	**1.0**	0.56	0.77	0.389	**1.0**	0.692	0.787	0.532	0.995	0.697	0.809	0.538	0.997
	All	0.981	0.983	0.992	0.976	0.939	0.955	0.914	0.98	0.952	0.956	0.953	0.959	0.94	0.951	0.925	0.97

Table 7.6: Binary classification results by grayscale P-CLAS models compared to the RGB version. Rows denote the network and its training data, while columns denote the test dataset. The Unmodified dataset (U), Underwater superimposed (UW), and Overwater superimposed (OW) were combined and used for testing. We provide F1 scores, accuracy, True Positive Rates (TPR), and True Negative Rates (TNR) for each case.

The differences here are far more substantial than those seen in the single-scale experiment. The GS-UW+U trained models achieve relatively poor results, only achieving somewhat good performance on the U testset. This could not be said for the GS-OW+U network, which classifies almost all of the U-dataset as 'clean', and otherwise gives subpar results.

While the GS-U trained classifier still performs very poorly on the superimposed testsets, it was consistently able to outperform the normal U-trained P-CLAS. Of note, is that All-trained grayscale has not nearly as significant of a loss in performance as the other synthetically trained grayscale models.

While the degree to which training on grayscale images impacts performance seems to vary based on training data, it seems clear that models trained on RGB data are consistently more able to model the datasets, while generalising more effectively to other datasets. Particularly consistent is the reduction in TPR seen on nearly all models. Considering that the computational benefit of grayscale is mainly memory related, we believe RGB patches to be the preferable input, when available.

7.4.3 Keypoint Filtering

In Section 4.2.2, we introduce keypoint filtering, with which snowy keypoints are filtered based on their visibility in the final superimposed sequence and clean keypoints are filtered based on their proximity to marine snowflakes in the final superimposed sequence. Here we test the impact of keypoint filtering on the final trained models.

To improve readability, we divide these results into separate parts for P-CLAS and D-CLAS. Beginning with Table 7.7, we list the results of P-CLAS trained on UW, OW, and UW+OW datasets with and without filtering (denoted F and NF, respectively, *e.g.* NF-OW). All results are given on test splits *with* filtering.

P-CLAS's results without keypoint filtering

P-CLAS

		UW				OW				UW + OW			
		F1	Acc	TPR	TNR	F1	Acc	TPR	TNR	F1	Acc	TPR	TNR
F	UW	**0.967**	**0.977**	**0.961**	0.986	0.812	0.798	0.954	0.666	0.876	0.89	0.958	0.845
	OW	0.933	0.955	0.888	**0.992**	0.948	0.954	0.92	**0.982**	0.943	0.956	0.91	**0.987**
	UW+OW	0.961	0.973	0.945	0.988	0.943	0.946	**0.974**	0.923	**0.955**	**0.963**	**0.965**	0.961
NF	UW	0.633	0.761	0.589	0.854	0.553	0.581	0.567	0.592	0.596	0.676	0.592	0.733
	OW	0.934	0.955	0.912	0.978	**0.951**	**0.956**	0.944	0.965	0.945	0.956	0.933	0.972
	UW+OW	0.85	0.895	0.851	0.918	0.896	0.9	0.939	0.868	0.881	0.902	0.905	0.899

Table 7.7: Results of P-CLAS from models trained on data with (F) and without filtering (NF). The evaluation was done on test splits which *did* use snow proximity filtering and variance-based visibility filtering.

Glancing at the results in Table 7.7, it may seem like keypoint filtering improves P-CLAS's performance, with the NF-trained models generally performing significantly worse than their F counterparts. However, looking closer at the results, a clear difference between NF-models is apparent. Importantly, the NF-OW model remains competitive with the F models and is seemingly unaffected by the lack of filtering. Meanwhile, the NF-UW model's performance is significantly lower across all metrics, giving some of the worst results seen so far from a P-CLAS model trained on superimposed data. The NF-UW+OW model (whose training data is very nearly 50% UW) also shows worse than normal results, but to a lesser extent.

This would indicate a stark difference in the effect of keypoint filtering on UW and OW. We examined the keypoint filtering process and found that snow proximity filtering removed on average 18.6% and 17.8% of the keypoints in UW and OW, respectively. Variance-based visibility filtering removed 16.34% and 12.1% of the keypoints in UW and OW, respectively. Snow proximity filtering being relatively similar among the two datasets is not surprising given that they superimpose the

same extracted snow. While the difference in the need for variance-based visibility filtering is interesting, it does not seem substantial enough to confidently explain the reduced impact of filtering on the OW models. More importantly, when visualising some of the filtered keypoints, we did not find the removed OW keypoints to be more or less visible than the UW keypoints.

D-CLAS's Results without Keypoint Filtering

In Table 7.8, we list the results of D-CLAS when trained on datasets with and without keypoint filtering.

D-CLAS

		UW				OW				UW + OW			
		F1	Acc	TPR	TNR	F1	Acc	TPR	TNR	F1	Acc	TPR	TNR
F	UW	0.907	0.935	0.904	0.952	0.883	0.889	0.917	0.865	0.893	0.912	0.911	0.912
	OW	0.894	0.923	0.93	0.92	0.903	0.91	0.917	0.905	0.899	0.917	0.922	0.913
	UW+OW	0.917	0.941	0.923	0.951	**0.906**	**0.913**	0.914	0.912	0.91	0.927	0.918	0.933
NF	UW	0.899	0.926	**0.942**	0.918	0.868	0.868	**0.949**	0.799	0.881	0.896	**0.946**	0.863
	OW	0.911	0.936	0.928	0.941	0.9	0.91	0.891	**0.925**	0.905	0.923	0.907	0.934
	UW+OW	**0.924**	**0.947**	0.922	**0.961**	0.902	0.91	0.907	0.913	**0.911**	**0.928**	0.913	**0.938**

Table 7.8: Results of D-CLAS from models trained on data with (F) and without filtering (NF). The evaluation was done on test splits which *did* use snow proximity filtering and variance-based visibility filtering.

These results may seem comparably uneventful; the significant differences brought by the NF-UW dataset in Table 7.7 are not present here. Particularly the two UW+OW-trained models perform almost identically, although the NF variant typically comes out on top. The NF-UW model consistently sees significant improvements on TPR, but equally significant losses on TNR. In fact, NF-UW has the best TPR and worst TNR on every testset. With NF-OW it is the other way around, with gains on the TNR metric, and losses in TPR.

Again, a possible reason behind the different results from P-CLAS and D-CLAS is the encoding of the ORB descriptor, which in this case may encode keypoint patches in a manner which allows D-CLAS to be less influenced by unfiltered keypoints. We are not sure of what the specifics of this benefit is, but one hypothesis is that it could be related to the illumination-invariance of ORB descriptors. Alternatively, architectural differences, such as the number of network parameters, could also explain the difference between P-CLAS and D-CLAS.

Summary of Keypoint Filtering Study

The results of our keypoint filtering study are less clear than we had hoped for. D-CLAS's abilities to model the datsets was consistent between NF and F models when considering F1 and accuracy. The main difference was the balance between

preciscion and recall of various models. P-CLAS results presented themselves quite differently from D-CLAS, with NF-results favouring models not trained on datasets which included the UW dataset.

In our view, further qualitative experiments are needed to fairly evaluate the filtering methods and whether they behave as intended. Based on our own results, we will only conclude that the visibility filtering has varying impact based on training data and the kind of architecture used.

Chapter 8

Discussion

In this section, we discuss the implications of our results and consider the research questions introduced in the introduction. The research questions defined in Chapter 1 are:

- RQ1: How can the effect of marine snow on keypoint detection, matching, and real-time SLAM be mitigated?
  - RQ1.1: Can ML methods mitigate the impact of marine snow on underwater keypoint detection, matching, and real-time SLAM?
- RQ2: How can marine snow be digitally synthesised for use in machine learning?
  - RQ2.1: What are the characteristics of marine snow?

8.1 Comparing Qualitative and Quantitative Results

In our experiments, we generated quantitative results on four combinations of our test datasets, gave qualitative evaluations of keypoint classification on 6 underwater sequences, and qualitatively evaluated pySLAM with and without keypoint classification on 4 real-world sequences, and the synthetic VAROS and Snowy-VAROS sequences. We also conducted ablative studies on P-CLAS' multi-scale design and RGB patches, and the keypoint filtering used with our synthetic datasets.

Our impression of the quantitative results is that they were less applicable to real-world sequences than we had preferred. While the quantitative results indicated that P-CLAS was consistently outperforming D-CLAS, and that All-trained models generalised to more data than other networks, both observations turned out to be false in the qualitative tests. For this reason, we found it more difficult to rely on the quantitative results used in our ablation studies. In retrospect, the issues

with the qualitative results may have been avoided with a manually labelled test dataset which could stand as a definitive benchmark for marine snow classification. This would, however, present inevitable challenges related to manual effort, and the surprising difficulty of discerning the label of some positive and negative samples.

Since the ablation studies were largely based on quantitative metrics on the test datsets, they too were impacted by the lack of a clear benchmark for quantitative evaluation. With this in mind, we believe a greater focus on qualitative evaluation may have provided better results for discussion, or alternatively, the manually labelled dataset mentioned earlier. With that said, we still found some general tendencies related to the networks' ability to model the datasets on which we base our discussion on.

The qualitative experiments, however, were very valuable in developing our understanding of P-CLAS, D-CLAS, and to an extent, the datasets. The main issue of our qualitative results is their added subjectivity and need for approximations. Consequently, this is a topic we discuss in the limitations section (Section 6.3). Additionally, it can be challenging to convey observations from videos in a written format. With that said, the broad characteristics of P-CLAS and D-CLAS, and their impact on pySLAM appeared reasonably unambiguous, and we find the qualitative experiments enabled us to reliably characterise the performance and limitations of P-CLAS and D-CLAS.

One drawback which we observed during our testing of pySLAM, was that the frame timings it produced could not be used to characterise the actual computational load imposed by P-CLAS and D-CLAS since time use was highly dependent on the particular workload pySLAM was subjected to, and not the number of keypoints which were classified by the networks. With that said, the timings presented by pySLAM give an accurate view of how pySLAM specifically performs with P-CLAS and D-CLAS, and therefore serve as a reminder that introducing new, computationally demanding methods in a SLAM system demands careful integration to efficiently distribute computing resources.

Additionally, since some of our results relate to pySLAM's point clouds, it should be mentioned that it can be difficult to evaluate point clouds from still images, especially when asked to compare point clouds. This is because point clouds can appear very differently based on which position they are observed from. Moreover, understanding the 3D structure of a point cloud from a 2D image can be very challenging. However, we have tried to maintain consistent viewpoints in our figures which present the 3D clouds such that their structure translates well to 2D images. Importantly, we believe that the conclusions which base themselves on similarities between point clouds are general enough to be supported by the data. Moreover, we have had the benefit of examining the point clouds in pySLAM's 3D visualisation window when making our observations, meaning we could view the point clouds from any 3D position.

8.2 Classifiers

In our experiments, we evaluated two approaches to keypoint classification of marine snow: the descriptor-based D-CLAS and the patch-based P-CLAS.

P-CLAS's design was motivated by existing methods from the literature, where multi-scale CNN architectures and RGB inputs were common in snow and marine snow suppression. In our ablation study, we evaluated our multi-scale design of P-CLAS which is one of the reasons why D-CLAS is faster than P-CLAS. While the multi-scale implementation did surpass the single-scale implementation overall, the margin with which it did so was, in our opinion, too small to justify the current multi-scale design, due to its relatively significant computational burden. In the case of RGB inputs, the results were more in favour of the current design, leading us to the conclusion that P-CLAS benefits from RGB input.

Before we had our first results, we were uncertain whether classifying on descriptors would be successful. However, the D-CLAS architecture with its ORB descriptors proved it possible. With that said, our evaluation of other descriptors showed how ORB was an exception to the norm. This is unfortunate because initially, the main downside we could find with descriptor-based classification was the need to tailor and train neural networks for specific descriptors. However, based on our experiments with different descriptors, we now consider the main downside of descriptor-based marine snow classification to be the significant limitations it imposes on which descriptors it can be used with for SLAM.

8.2.1 Comparing P-CLAS and D-CLAS

Our qualitative tests are perhaps the most relevant for a head-to-head comparison of the real-world capabilities of P-CLAS and D-CLAS. While both classifiers were generally able to accurately classify marine snow, both presented recurring issues in certain situations. On sequences where P-CLAS models exhibited local concentrations of false positives like the pipe in Video 2, D-CLAS was often less affected. However, this was generally accompanied by an increased amount of false negatives, even on flat backgrounds.

Similarly, on sequences where P-CLAS struggled with the textured backgrounds—often signalled by U-trained P-CLAS classifying all keypoints as good—D-CLAS presented more reasonable results on the marine snow, alongside an increase in false positives compared to P-CLAS. All in all, D-CLAS appears to be a more robust classifier. However, considering that P-CLAS did better on the quantitative tests, we believe P-CLAS might maintain greater consistency, and potentially surpass D-CLAS's performance where it currently does not, if it receives even more diverse and realistic data, and perhaps undergoes some architectural modifications. Additional forms of data augmentation may also benefit P-CLAS, *e.g.* changes to the hue and saturation of the training patches.

Our main hypothesis to explain the differences between P-CLAS and D-CLAS performance on the qualitative and quantitative tests is that P-CLAS's CNN architec-

tures makes it able to rely on contextual clues from the background which are not available through descriptors. This could lead to P-CLAS models performing better when encountering backgrounds familiar from training—*i.e.* the quantitative tests—but worse on unfamiliar backgrounds in the qualitative tests.

We also performed an ablation study with P-CLAS where we replaced its RGB input with grayscale. This experiment was also relevant to D-CLAS since ORB descriptors are based on grayscale images. Ultimately our observations suggested the differences between grayscale and RGB input were small, but consistently in favour of RGB patches. With that said, the difference was not so large that we think D-CLAS is held back by ORB using grayscale images.

With pySLAM, we found that both P-CLAS and D-CLAS gave similar improvements to consistency and tracking performance, despite differing somewhat in the number of keypoints they removed. This may indicate that tracking can be done with traditional outlier rejection as long as the number of marine snow keypoints is reduced sufficiently such that the snow is no longer dominating the RANSAC motion hypothesis.

While the execution speed of the two methods was not the main priority, it was still an important point of comparison, and directly related to RQ1. As discussed earlier, D-CLAS was significantly faster in both cycle times within pySLAM and keypoints per second, compared to P-CLAS. With modifications to P-CLAS's patch-based input and the use of industry-standard optimisation tools, we believe both P-CLAS- and D-CLAS-type architectures can be used for real-time SLAM. In the case of pySLAM, our results indicate that significant gains in computational efficiency can be achieved if its computational resources are distributed more effectively when using P-CLAS and D-CLAS.

Considering that D-CLAS is the most computationally efficient classifier while appearing to generalise better to different sequences, we find D-CLAS to be the preferred classifier among our two designs. If this preference should be extended to descriptor-based classification in general, can not be determined from our results and it should still be highlighted how descriptor-based classification seemingly entails significant restrictions on which descriptors can be used.

8.2.2 Need For Iteration

It is no secret that the current designs of P-CLAS and D-CLAS are very similar to the first implementations we tested, and hence present quite generic CNN and FCNN architectures. Our initial plan was to iterate the design of the architectures throughout the project, however, once the first results came in, we observed that both P-CLAS and D-CLAS were far more capable of modelling the datasets than we expected. Since neither architecture had trouble learning the datasets, we found it more worthwhile to focus on the datasets themselves, because we were confident that this would give greater returns on real-world performance than modifying the architectures. Given the results from our experiments, we consider

computational efficiency as the main motivation for further iteration for the classifiers. Consequently, we still view improving the datsets as the most important way of increasing accuracy on underwater sequences.

Specifically, the ideas we had for iterating D-CLAS was mainly to examine smaller versions of the same general FCNN architecture, *i.e.* reducing the number of layers, and the number of nodes within each layer. For P-CLAS, both the configuration of patch extraction and the network itself would have been subject to iteration. The choice of the number of channels in P-CLAS's layers, as well as the number of layers, was somewhat improvised. Consequently, there may be considerable gains in computational efficiency from modifying these attributes. Finally, more complex learning rate schedules, different activation functions, and additional regularisation techniques could be considered for both networks, alongside additional data augmentation for P-CLAS's image patches.

8.3 Datasets

For this project, we developed three datasets for training, validation and testing: U, OW, and UW. We also developed Snowy-VAROS for evaluation with pySLAM.

The results in Chapter 7 were not only intended to evaluate P-CLAS and D-CLAS, but also the three training datasets. As we suspected, both qualitative and quantitative results indicated that the two classes in the U dataset were trivially separable, with their marine snow class characterised by flat backgrounds. Despite this, U-trained D-CLAS was able to achieve competitive baselines during qualitative evaluation, which other models were not always able to surpass. However, in general training on superimposed data seemed to be a benefit.

Conceptually, using 'clean' and 'marine snow' videos to rapidly label ORB keypoints, and subsequently extending this data by superimposing the marine snow images onto the clean backgrounds seems like a valid approach to the labelling and data variety issues we faced when creating marine snow datasets. In practice, our U and UW datasets could still benefit from more varied backgrounds and perhaps more variations of marine snow as well. It was actually surprisingly difficult to find background sequences for superimposing, *i.e.* sequences entirely without marine snow. Of course, such sequences are not neatly labelled in NOAA's portal, however, a more important challenge is that many sequences which appear free of marine snow are actually full of it. However, due to lighting conditions and qualities of the surrounding terrain, such as brightly lit rocks, the marine snow in these sequences is not clearly visible. Consequently, when we superimpose marine snow on these sequences, it too becomes difficult to notice.

The challenge of finding backgrounds fit for superimposing in NOAA's video collection, motivated us to explore other datasets, eventually leading us to create the OW dataset with backgrounds from the Exclusively Dark Images Dataset [Loh and Chan 2019]. This was a somewhat optimistic approach since the connection to underwater images is based on a naive argument that dark images are similar

to underwater images. On the other hand, one may ask how relevant the background really is: shouldn't a true marine snow detector be background-agnostic?

In practice, we could not see any clear disadvantage for models trained on the OW dataset, and in the qualitative test, OW+U trained models were often among the best. With that said, we do believe that more care should be taken in selecting backgrounds for OW. Specifically, the Exdark dataset features a large number of images with dynamic ranges which are rarely seen underwater. This comes from light bulbs captured in the images which can wash out superimposed marine snow. We do not know what the consequence of these images are and believe this should be examined. However, given that these images are relatively common in Exdark and the performance of OW+U-trained models, they may not have a particularly detrimental impact on classification performance.

One curious tendency we observed in the qualitative tests, was how training on the All-dataset frequently degraded the accuracy of the networks. This was true for both P-CLAS and D-CLAS models, and was often characterised by an increase in false negatives compared to the other models. It is striking how the false negatives were often found on flat backgrounds. Such keypoints often appear with a forward-facing camera, or when facing the underwater horizon, and are therefore common when the robot is navigating its surroundings. A possible explanation of why All-trained models struggles with marine snow on flat backgrounds is that the marine snow keypoints in the U dataset, which are all on flat backgrounds, are important to ensure that P-CLAS and D-CLAS learn to recognise such keypoints. In the All-dataset, these simple patches may become too outnumbered in the training data, such that All-trained models do not learn to reliably classify marine snow on untextured backgrounds. This is supported by the TPR scores on the U dataset, where All-trained D-CLAS is the worst model by a relatively large margin, however, All-trained P-CLAS scores around the P-CLAS average.

Snowy-VAROS was made with our superimposing approach to create a dataset with relative pose ground truths and marine snow motion noise. We believe the use of Snowy-VAROS in our experiments was successful, and it was particularly useful to be able to compare results with and without superimposed snow. It should be said that the marine snow sequence we superimposed featured particularly large and dense marine snow, consistent with the Eelume sequences, which may not be relevant for all use cases. However, with the superimposing approach, it is straightforward to create multiple variations of Snowy-VAROS with varying densities and types of marine snow. It would also be interesting to create a sequence in which the presence of marine snow gradually increases from low to high to see how SLAM systems behave as the marine snow conditions change.

8.4 Keypoint Filtering

When designing our superimposed datasets, we observed that we could detect substantially more keypoints on marine snow if we did detection separately on

the extracted snow and background as opposed to the combined, superimposed image. This approach, however, necessitated the development of our snow vicinity filtering and variance-based visibility filtering to make sure the keypoints remained correctly labelled in the superimposed image.

Our ablation study on keypoint filtering showed that P-CLAS models trained on data which included the UW split had worse results without keypoint filtering. D-CLAS's results, however, were far less intuitive with models both improving TPR at the cost of TNR, and vice versa. We find these results particularly difficult to analyse, other than to say that the impact of keypoint filtering varies both depending on the dataset and the classifier. Specifically, it seems like the ORB encoding makes D-CLAS able to handle unfiltered data better than P-CLAS, and that visibility filtering is more used on the UW dataset. Alternatively, other architectural differences, like the number of network parameters, may explain the difference between P-CLAS and D-CLAS.

Unfortunately, this means that we are unable to draw any meaningful conclusions about how effective our filtering methods are at improving the realism of the dataset, and the accuracy of the labels. Additional qualitative tests of the NF-trained classifiers, as well as the filtering schemes themselves, may be required to understand this issue fully.

8.5 Research Questions

We now return to our research questions from Chapter 1 and discuss them in view of our results.

Beginning with RQ1: How can the effect of marine snow on keypoint detection, matching, and real-time SLAM be mitigated? On the topic of marine snow suppression, we have listed approaches related to marine snow removal in our literature review, hypothesised an approach based on semantic segmentation (see Chapter 9 Future Work), and, most importantly, developed and evaluated our own keypoint classification approach.

Based on our extensive experiments, we argue that both descriptor-based classification and patch-based classification are viable approaches to mitigate the impact of marine snow on SLAM. With our pySLAM implementation, we demonstrated how the use of keypoint classification enabled tracking and mapping in adverse marine snow conditions, where pySLAM would otherwise fail immediately. Although our methods are not real-time, and P-CLAS consumes a significant portion of pySLAM's resources, they were mainly designed to evaluate the feasibility of their approach to marine snow suppression, meaning real-time operation was not a high priority. Importantly, we have not observed any behaviour which suggests that real-time performance can not be achieved with the keypoint classification approach.

RQ2: How can marine snow be digitally synthesised for use in machine learn-

ing? This has been covered both by developing our own semi-synthetic approach for marine snow superimposing and in Section 3.3 in the literature review. The literature review divided snow and marine snow synthesis into three domains: superimposing methods, 3D-environment-based methods, and hybrid methods.

During the early design phase of our project, we examined two schemes for synthesising marine snow: one 3D-environment-based method and one superimposing method. Through these early examinations, we argued that the superimposing method is superior for machine learning use cases, as much larger data quantities can be generated, with greater variety, in a shorter amount of time. This comes from the fact that with 3D-environment methods, large underwater environments need to be made on a per sequence basis, while with the superimposing method all that is needed is collections of sequences exclusively with and without snow.

From our qualitative tests, we were able to compare the results of networks trained with and without our synthetic datasets. The results showed that classifiers trained primarily on synthetic data were able to generalise to a greater amount of underwater footage than classifiers trained exclusively on the non-synthetic U dataset. Moreover, the synthetic approach accomplished our primary goal to improve the classifiers' performance on marine snow keypoints with textured backgrounds, although this effect was far more pronounced with P-CLAS than D-CLAS. We were also unable to draw any significant conclusions from our ablation study about the filtering mechanisms which were developed to improve the realism of the datasets.

For RQ2.1: What are the characteristics of marine snow, we defer to Section 2.4.1. The most important points from this section is the diverse appearances of marine snow, its widespread presence throughout the entire open ocean, and the complex set of factors which determine its appearance in video recordings.

Chapter 9

Future Work

In this chapter, we suggest ways of extending our work and some other approaches for marine snow suppression which could be worth pursuing. To facilitate extensions of our project, we have included the names of all NOAA videos we used in Appendix C, documented our code and shared it with the AROS research group, published our U dataset with the snow extraction and superimposing code at `https://zenodo.org/record/6424752`, and published and presented our paper for other computer vision researchers at the 2022 CVPR conference. A recording of the presentation will be published by the workshop organisers before October 2022.

9.1 Extending our Work

This section details the most significant ways of continuing work on our classifiers and datasets. We also include advice on how to evaluate methods like ours going forward.

9.1.1 Keypoint Classifiers

Considering that our classifiers, despite their relative simplicity, were able to model their training data with high accuracy and achieve high performance on the test datasets, we argue that modifying the classifiers should not be a priority, unless for the purpose of creating more computationally efficient models. Given the models' ability to approximate their data, it seems to us that the datasets themselves are the key to improving real-world performance in marine snow keypoint classification. To increase computational efficiency, we would focus on simplifying the pre-processing steps of P-CLAS and evaluating shallower architectures for both classifiers.

9.1.2 Datasets

To improve the datasets, we would look at including even more diverse backgrounds both during dataset synthesis and possibly through data augmentation tricks during training and dataset synthesis. One should also look at options to improve the dataset pipelines, particularly the keypoint filtering steps which are needed to maintain realism in the datasets. Specifically, we worry that our visibility filtering approach does not generalise as well as it should due to its use of a constant threshold. In the case of visibility filtering, we suggest both looking into thresholds that are adapted to statistical qualities of the input. We also think it is worthwhile to consider other filtering criteria than just variance, *e.g.* comparing the pixel intensities of the background and superimposed snow.

To thoroughly evaluate the realism of synthesised data, we propose inviting members of the public to differentiate between real and synthetic snow keypoints. With a large enough group, their ability or inability to predict the origin of a keypoint can quantify how effective the synthetic approach is. Likewise, the filtering methods can be evaluated by seeing if respondents would filter similarly to the filtering approach being evaluated. However, this final suggestion can be risky because keypoints that the respondents think should be removed may not align with what is actually ideal for the task.

Our approach to snow extraction, while reasonably effective and robust against artefacts, is significantly slowed down by the strided approach, and typically needs 3–4 seconds to extract snow from the image. Consequently, a more efficient way of extracting snow could save multiple hours during dataset synthesis. Another step of our superimposing approach which may have room for improvement is how we manage the hue of the marine snow sequence, which remains very visible in the extracted snow, *e.g.* as a light blue colour on all of the marine snow. Our current approach to removing this colouration is to convert the snow to grayscale such that the hue disappears. However, a more sophisticated approach may consider qualities of the background image, and replace the hue of the superimposed snow with something which matches the background more closely.

On the subject of Snowy-VAROS, we would be interested in developing additional sequences with different severity of marine snow conditions. Furthermore, a snowy-VAROS sequence which features gradually increasing severity of marine snow would also be useful, *e.g.* to evaluate how SLAM systems react when they initialise in good conditions, which slowly deteriorate as the sequence progresses. This latter suggestion may necessitate superimposing marine snow from multiple sequences with differing conditions, either in series or on top of each other.

9.1.3 Improved Evaluation

Because we are among the first to work on marine snow keypoint classification, we had to create our own benchmarks to evaluate P-CLAS and D-CLAS. Our approach consisted of multiple synthetic testsets and one non-synthetic testset for

quantitative evaluation and use in our ablative studies. We also performed a qualitative evaluation of both stand-alone classification performance and SLAM performance with pySLAM.

Our main issue with the quantitative tests is the difficulty of knowing how reliably the results translate to real world scenarios. Among other things, this limited the insights we could gather from our ablation study. Consequently, we have highlighted our desire for a manually labelled dataset for use in quantitative evaluation.

We are mostly content with our qualitative experiments on keypoint classification. However, these results could be elevated further with a graphical interface that gives users the ability to produce quantitative metrics within a user-specified bounding box. With this tool, users would place bounding boxes around areas of the video where all keypoints are of a known class. Within these boxes, one could generate metrics by counting the number of keypoints classified correctly within the selected area, thereby offering more fine-grained comparisons of different classifiers.

It could also be useful to test our classifiers in other SLAM systems and even in an actual underwater environment. We would also be interested in seeing how the networks can complement IMU-aided VSLAM and similar systems which use multiple sensors to estimate pose. The evaluation with VAROS and Snowy-VAROS would also have been improved if we had managed to use their relative pose ground truths to calculate the trajectory error of pySLAM in the synthetic sequences. Additionally, it may have been possible to use the snow masks which were made alongside Snowy-VAROS to quantitatively evaluate the classifications of P-CLAS and D-CLAS on this sequence.

9.2 Other Approaches

Our datasets and snow superimposing methods can be used for other purposes than keypoint classification. Here, we list two approaches, different from ours, which could support SLAM front ends that are immune to marine snow motion noise.

9.2.1 Keypoint Detector Immune to Marine Snow

Our keypoint classifiers are relatively easy to train and use with SLAM, but this comes at the cost of requiring a new keypoint rejection step before keypoint matching, and risks leaving the SLAM systems with no good keypoints after the rejection step. Although the added computation does not prohibit this approach, an ideal solution would be a keypoint detector which does not detect marine snow to begin with. We have a proposal for such a keypoint detector, which combines advances in learned keypoint detection with our snow superimposing method.

Recent research in keypoint detection has introduced neural network-based detectors and descriptors. Our proposal draws inspiration from one of these architectures, named SuperPoint [DeTone, Malisiewicz, and Rabinovich 2018]. This method is known for an approach called *Homographic Adaption*, in which ground truths are automatically generated by detecting keypoints in a series of warped versions of an image. The SuperPoint network is then trained to detect the unwarped equivalents of these keypoints in the original image.

Our proposal is based on extracting ground truth keypoints from snow-free images, and then superimposing snow onto the snow-free image. Similar to SuperPoint, we would use Homographic Adaptation to find keypoints in the snow-free image, and then train the keypoint detector to recognise these keypoints in the snowy, superimposed image. This scheme is represented in Figure 9.1. An additional benefit of this approach is that it can be extended to include other challenges of underwater keypoint detection, *e.g.* dynamic illumination, marine wildlife, and water turbidity.

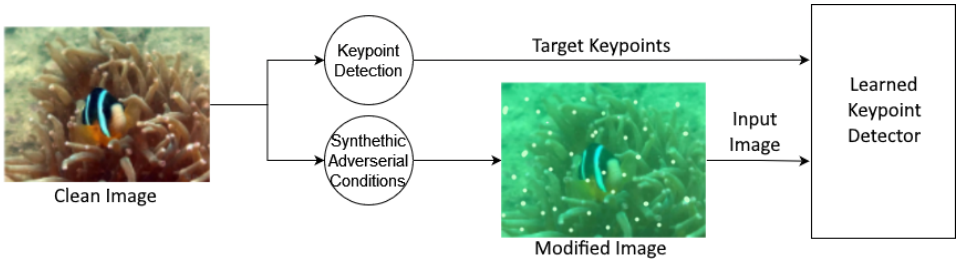


Figure 9.1: Our proposal for a marine snow resistant keypoint detector. The underwater images originate from the Underwater Image Enhancement Benchmark Dataset [C. Li et al. 2020].

9.2.2 Parallel Architecture

Here, we propose a parallel architecture based on a snow segmentation approach. Since snow segmentation does not depend on the output of keypoint detectors, nor does keypoint detection depend on segmented snow, these processes can run simultaneously, possibly reducing computational cycle time compared to systems that must be run in series. Once both processes are done, their outputs combine to reject marine snow keypoints, as seen in Figure 9.2.

The keypoint rejection could follow a similar scheme to our snow proximity filtering, where a small area around a keypoint in the snow segmentation is probed for the presence of marine snow. For the segmentation architecture, we propose the following:

- A CNN-based semantic segmentation network made for small-object segmentation.
- Training data consisting of:

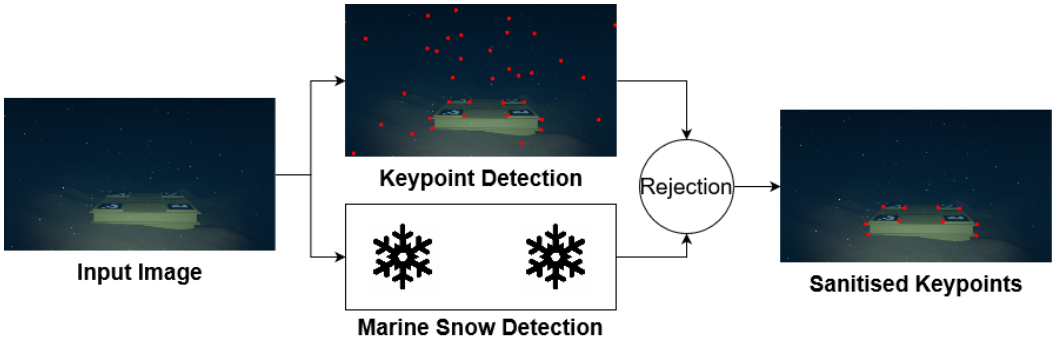


Figure 9.2: A parallel pipeline, running keypoint detection and snow detection at the same time.

  - The clean background image.
  - A marine snow image from our snow extraction pipeline to generate the ground truth.
  - The combined image of snow and background to use as input.
- A loss function which accounts for extreme class imbalance within each ground truth.

Alternatively, a hand-crafted approach could be developed by modifying our snow extraction pipeline such that it works on both textured and untextured backgrounds.

Chapter 10

Conclusion

This thesis has explored the challenge of marine snow for underwater Simultaneous Localisation and Mapping (SLAM). Marine snow is a broad term denoting various forms of detritus found throughout the open ocean. When feature-based Visual SLAM (VSLAM) methods operate in conditions with marine snow, their pose estimates and mapping can be influenced—or interrupted completely—by keypoint correspondences placed on the moving marine snow particles.

In our work, we have presented two schemes, D-CLAS and P-CLAS, which classify and remove marine snow keypoints before correspondences are generated. D-CLAS classifies keypoints by their ORB descriptor, while P-CLAS classifies keypoints after extracting patches around their position in the image. We also developed three training datasets, two of which were made with our novel snow synthesis method which removes marine snow from images with a flat background and superimposes it onto any image.

We performed both quantitative and qualitative experiments to evaluate the performance of the classifiers and their ability to generalise to real-world data based on which combination of synthetic and non-synthetic data was used to train them. Our quantitative results showed that our classifiers successfully learn the classification task, and also highlighted how the choice of the descriptor is particularly important for D-CLAS to be useful. However, the tendencies observed in the quantitative results, such as P-CLAS outperforming D-CLAS, and networks trained on all datasets outperforming other networks with the same architecture, did not match our observations of real-world performance. We believe the quantitative results may have been more useful if we could perform testing on a manually labelled test dataset as a benchmark for the classifiers, but we do not know the feasibility of such an approach.

In the qualitative results, we examined the strengths and weaknesses of the two classifiers in six underwater sequences and observed how both classifiers identified marine snow with acceptable accuracy. The most important observations included P-CLAS being particularly dependent on synthetic data to generalise to marine snow keypoints with textured backgrounds, the tendency of models trained on all training datasets to perform worse on keypoints mainly covered by the non-synthetic dataset (*i.e.* marine snow on a flat background), and D-CLAS' overall superior performance compared to P-CLAS.

In another qualitative experiment, we implemented the classifiers in the SLAM framework pySLAM and evaluated its behaviour on four underwater sequences with varying levels of marine snow, as well as two synthetic sequences whose only difference was the presence of superimposed snow. Our results show an improvement in pySLAM's ability to reliably initialise under heavy marine snow when using our classifiers. Specifically, once P-CLAS or D-CLAS removed a large portion of the marine snow keypoints, pySLAM's built-in outlier rejection scheme was able to adequately remove the remainder. Consequently, both P-CLAS and D-CLAS had similar effects on tracking, however, P-CLAS was substantially more computationally demanding than D-CLAS. Moreover, we observed no adverse impact from the classifiers when operating pySLAM under lighter marine snow conditions. On the synthetic sequences, our methods were able to produce results on the sequence with marine snow which were comparable to the results obtained on the snowless, synthetic sequence.

We believe the main areas to improve are computational efficiency to facilitate real-time SLAM, and in the development of more realistic datasets for training and evaluation to increase performance in real sequences. The latter may be accompanied by a survey in which participants evaluate the realism of the synthetic data.

Bibliography

Alldredge, Alice L. and Mary W. Silver (1988). "Characteristics, dynamics and significance of marine snow". In: *Progress in Oceanography* 20.1, pp. 41–82. ISSN: 0079-6611. DOI: https://doi.org/10.1016/0079-6611(88)90053-5. URL: https://www.sciencedirect.com/science/article/pii/0079661188900535.

Arnold-Bos, Andreas, Jean-Philippe Malkasse, and Gilles Kervern (Mar. 2005). "A preprocessing framework for automatic underwater images denoising". In: *European Conference on Propagation and Systems*. Brest, France. URL: https://hal.archives-ouvertes.fr/hal-00494314.

Banerjee, Soma et al. (2014). "Elimination of Marine Snow effect from underwater image - An adaptive probabilistic approach". In: *2014 IEEE Students' Conference on Electrical, Electronics and Computer Science*, pp. 1–4. DOI: 10.1109/SCEECS.2014.6804438.

Bay, Herbert et al. (2008). "Speeded-Up Robust Features (SURF)". In: *Computer Vision and Image Understanding* 110.3. Similarity Matching in Computer Vision and Multimedia, pp. 346–359. ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2007.09.014. URL: https://www.sciencedirect.com/science/article/pii/S1077314207001555.

Bernuth, Alexander von, Georg Volk, and Oliver Bringmann (2019). "Simulating Photo-realistic Snow and Fog on Existing Images for Enhanced CNN Training and Evaluation". In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pp. 41–46. DOI: 10.1109/ITSC.2019.8917367.

Boffety, Matthieu and Frédéric Galland (May 2012). "Phenomenological marine snow model for optical underwater image simulation: Applications to color restoration". In: *2012 Oceans - Yeosu*, pp. 1–6. DOI: 10.1109/OCEANS-Yeosu.2012.6263448.

Calonder, Michael et al. (2010). "BRIEF: Binary Robust Independent Elementary Features". In: *Computer Vision – ECCV 2010*. Ed. by Kostas Daniilidis, Petros

Maragos, and Nikos Paragios. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 778–792. ISBN: 978-3-642-15561-1.

Campos, Carlos et al. (Dec. 2021). "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM". In: *IEEE Transactions on Robotics* 37.6, pp. 1874–1890. ISSN: 1941-0468. DOI: 10.1109/tro.2021.3075644. URL: http://dx.doi.org/10.1109/TRO.2021.3075644.

Chen, T. and Hong Ren Wu (Jan. 2001). "Adaptive impulse detection using center-weighted median filters". In: *IEEE Signal Processing Letters* 8.1, pp. 1–3. ISSN: 1558-2361. DOI: 10.1109/97.889633.

Chen, Wei-Ting et al. (Oct. 2021). "ALL Snow Removed: Single Image Desnowing Algorithm Using Hierarchical Dual-Tree Complex Wavelet Representation and Contradict Channel Loss". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4196–4205.

Chli, Margarita and Andrew Davison (Oct. 2008). "Active Matching." In: pp. 72–85. ISBN: 978-3-540-88681-5. DOI: 10.1007/978-3-540-88682-2_7.

Cho, Younggun and Ayoung Kim (2017). "Visibility enhancement for underwater visual SLAM based on underwater light scattering model". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 710–717. DOI: 10.1109/ICRA.2017.7989087.

Cybenko, G. (Dec. 1989). "Approximation by superpositions of a sigmoidal function". In: *Mathematics of Control, Signals and Systems* 2.4, pp. 303–314. ISSN: 1435-568X. DOI: 10.1007/BF02551274. URL: https://doi.org/10.1007/BF02551274.

Cyganek, Boguslaw and Karol Gongola (July 2018). "Real-time marine snow noise removal from underwater video sequences". In: *Journal of Electronic Imaging* 27, p. 1. DOI: 10.1117/1.JEI.27.4.043002.

DeTone, Daniel, Tomasz Malisiewicz, and Andrew Rabinovich (2018). *SuperPoint: Self-Supervised Interest Point Detection and Description*. arXiv: 1712.07629 [cs.CV].

Ding, Xinghao et al. (2016). "Single image rain and snow removal via guided L0 smoothing filter". In: *Multimedia Tools and Applications* 75.5, pp. 2697–2712.

Farhadifard, Fahimeh, Martin Radolko, and Uwe von Lukas (2017). "Single Image Marine Snow Removal based on a Supervised Median Filtering Scheme". In: *VISIGRAPP*.

Fasel, B (Mar. 2003). "An introduction to bio-inspired artificial neural network architectures". en. In: *Acta Neurol Belg* 103.1, pp. 6–12.

Fischler, Martin A. and Robert C. Bolles (June 1981). "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". In: *Commun. ACM* 24.6, pp. 381–395. ISSN: 0001-0782. DOI: 10.1145/358669.358692. URL: https://doi.org/10.1145/358669.358692.

Gautam, Ashmita and Dr K Raj (2018). "Rain removal in digital images using guided filter". In: *Int. J. Inform. Technol. Electr. Eng* 7.5.

Harris, Chris, Mike Stephens, et al. (1988). "A combined corner and edge detector". In: *Alvey vision conference*. Vol. 15. 50. Citeseer, pp. 10–5244.

Hartmann, Wilfried, Michal Havlena, and Konrad Schindler (June 2014). "Predicting Matchability". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Hase, H., K. Miyake, and M. Yoneda (Oct. 1999). "Real-time snowfall noise elimination". In: *Proceedings 1999 International Conference on Image Processing (Cat. 99CH36348)*. Vol. 2, 406–409 vol.2. DOI: 10.1109/ICIP.1999.822927.

He, K., J. Sun, and X. Tang (2013). "Guided image filtering". English. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.6. Cited By :3346, pp. 1397–1409. URL: www.scopus.com.

Hildebrandt, Marc and Frank Kirchner (June 2010). "IMU-aided stereo visual odometry for ground-tracking AUV applications". In: pp. 1–8. DOI: 10.1109/OCEANSSYD.2010.5603681.

Hodne, Lars Martin and Eirik Leikvoll (Dec. 2021). *Using Machine Learning for Autnomous Underwater Vehicles*. Project report in IT3915. Department of Computer Science, NTNU – Norwegian University of Science and Technology.

Hodne, Lars Martin, Eirik Leikvoll, et al. (June 2022). "Detecting and Suppressing Marine Snow for Underwater Visual SLAM". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 5101–5109.

Huang, Gao et al. (2016). *Densely Connected Convolutional Networks*. DOI: 10.48550/ARXIV.1608.06993. URL: https://arxiv.org/abs/1608.06993.

Huang, Shih-Chia et al. (Jan. 2020). "Single Image Snow Removal Using Sparse Representation and Particle Swarm Optimizer". In: *ACM Trans. Intell. Syst. Technol.* 11.2. ISSN: 2157-6904. DOI: 10.1145/3372116. URL: https://doi.org/10.1145/3372116.

Ioffe, Sergey and Christian Szegedy (2015). *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. DOI: 10.48550/ARXIV.1502.03167. URL: https://arxiv.org/abs/1502.03167.

Irisson, Jean-Olivier et al. (2022). "Machine Learning for the study of plankton and marine snow from images". In: *Annual review of marine science* 14.

Jaw, Da-Wei, Shih-Chia Huang, and Sy-Yen Kuo (Apr. 2021). "DesnowGAN: An Efficient Single Image Snow Removal Framework Using Cross-Resolution Lateral Connection and GANs". In: *IEEE Transactions on Circuits and Systems for Video Technology* 31.4, pp. 1342–1350. ISSN: 1558-2205. DOI: 10.1109/TCSVT.2020.3003025.

Kampffmeyer, Michael, Arnt-Borre Salberg, and Robert Jenssen (June 2016). "Semantic Segmentation of Small Objects and Modeling of Uncertainty in Urban Remote Sensing Images Using Deep Convolutional Neural Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Kim, Hyo Jin, Enrique Dunn, and Jan-Michael Frahm (Dec. 2015). "Predicting Good Features for Image Geo-Localization Using Per-Bundle VLAD". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Kim, Hyun-Koo et al. (2019). "Traffic Light Recognition Based on Binary Semantic Segmentation Network". In: *Sensors* 19.7. ISSN: 1424-8220. DOI: 10.3390/s19071700. URL: https://www.mdpi.com/1424-8220/19/7/1700.

Kingma, Diederik P. and Jimmy Ba (2014). *Adam: A Method for Stochastic Optimization*. DOI: 10.48550/ARXIV.1412.6980. URL: https://arxiv.org/abs/1412.6980.

Köser, Kevin and Udo Frese (2020). "Challenges in Underwater Visual Navigation and SLAM". In: *AI Technology for Underwater Robots*. Springer, pp. 125–135.

Koziarski, Michał and Bogusław Cyganek (2019). "Marine Snow Removal Using a Fully Convolutional 3D Neural Network Combined with an Adaptive Median Filter". In: *Pattern Recognition and Information Forensics*. Ed. by Zhaoxiang Zhang et al. Cham: Springer International Publishing, pp. 16–25. ISBN: 978-3-030-05792-3.

Lampitt, R.S. (2001). "Marine Snow". In: *Encyclopedia of Ocean Sciences (Third Edition)*. Ed. by J. Kirk Cochran, Henry J. Bokuniewicz, and Patricia L. Yager. Third Edition. Oxford: Academic Press, pp. 160–169. ISBN: 978-0-12-813082-7. DOI: https://doi.org/10.1016/B978-0-12-813081-0.00218-4. URL: https://www.sciencedirect.com/science/article/pii/B9780128130810002184.

Langer, Michael et al. (Jan. 2004). "A Spectral-particle hybrid method for rendering falling snow." In: pp. 217–226. DOI: 10.2312/EGWR/EGSR04/217-226.

Leonardi, Marco, Luca Fiori, and Annette Stahl (2020). "Deep learning based keypoint rejection system for underwater visual ego-motion estimation∗∗This work was supported by the Norwegian Research Council through the Centre for Autonomous Marine Operations and Systems at NTNU." In: *IFAC-PapersOnLine* 53.2. 21st IFAC World Congress, pp. 9471–9477. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2020.12.2420. URL: https://www.sciencedirect.com/science/article/pii/S2405896320331025.

Li, Chongyi et al. (2020). "An Underwater Image Enhancement Benchmark Dataset and Beyond". In: *IEEE Transactions on Image Processing* 29, pp. 4376–4389. DOI: 10.1109/TIP.2019.2955241.

Li, H. H., S. Liu, and Y. Piao (2016). "Snow Removal of Video Image Based on FPGA". In: *Proceedings of the 5th International Conference on Electrical Engineering and Automatic Control*. Ed. by Bo Huang and Yufeng Yao. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 207–215. ISBN: 978-3-662-48768-6.

Li, Lei et al. (2021). "Removal of Floating Particles from Underwater Images Using Image Transformation Networks". In: *Pattern Recognition. ICPR International Workshops and Challenges*. Ed. by Alberto Del Bimbo et al. Cham: Springer International Publishing, pp. 414–421. ISBN: 978-3-030-68790-8.

Li, Pengyue et al. (2019). “Stacked dense networks for single-image snow removal”. In: *Neurocomputing* 367, pp. 152–163.

Li, Zhi et al. (2019). “Single Image Snow Removal via Composition Generative Adversarial Networks”. In: *IEEE Access* 7, pp. 25016–25025. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2900323.

Liu, Yun-Fu et al. (June 2018). “DesnowNet: Context-Aware Deep Network for Snow Removal”. In: *IEEE Transactions on Image Processing* 27.6, pp. 3064–3073. ISSN: 1941-0042. DOI: 10.1109/tip.2018.2806202. URL: http://dx.doi.org/10.1109/TIP.2018.2806202.

Loh, Yuen Peng and Chee Seng Chan (2019). “Getting to Know Low-light Images with The Exclusively Dark Dataset”. In: *Computer Vision and Image Understanding* 178, pp. 30–42. DOI: https://doi.org/10.1016/j.cviu.2018.10.010.

Lowe, David G. (Nov. 2004). “Distinctive Image Features from Scale-Invariant Keypoints”. In: *International Journal of Computer Vision* 60.2, pp. 91–110. ISSN: 1573-1405. DOI: 10.1023/B:VISI.0000029664.99615.94. URL: https://doi.org/10.1023/B:VISI.0000029664.99615.94.

Mathai, A. M., P. Moschopoulos, and G. Pederzoli (Feb. 1999). “Random points associated with rectangles”. In: *Rendiconti del Circolo Matematico di Palermo* 48.1, pp. 163–190. ISSN: 1973-4409. DOI: 10.1007/BF02844387. URL: https://doi.org/10.1007/BF02844387.

Mikolajczyk, K. and C. Schmid (2001). “Indexing based on scale invariant interest points”. In: *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*. Vol. 1, 525–531 vol.1. DOI: 10.1109/ICCV.2001.937561.

Mitchell, Tom M (1997). *Machine learning*. eng. New York.

Mur-Artal, Raul, J. M. M. Montiel, and Juan D. Tardos (Oct. 2015). “ORB-SLAM: A Versatile and Accurate Monocular SLAM System”. In: *IEEE Transactions on Robotics* 31.5, pp. 1147–1163. DOI: 10.1109/tro.2015.2463671. URL: https://doi.org/10.1109%2Ftro.2015.2463671.

Mur-Artal, Raul and Juan D. Tardos (Jan. 2017). “ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras”. In: *IEEE Transactions on Robotics* 33.5, pp. 1255–1262. ISSN: 1941-0468. DOI: 10.1109/tro.2017.2705103. URL: http://dx.doi.org/10.1109/TRO.2017.2705103.

Nister, D. (2004). “An efficient solution to the five-point relative pose problem”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.6, pp. 756–770. DOI: 10.1109/TPAMI.2004.17.

Papadaki, Alexandra I. and Ronny Hänsch (2020). “Match or No Match: Keypoint Filtering based on Matching Probability”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 4371–4378. DOI: 10.1109/CVPRW50498.2020.00515.

Radolko, Martin, Fahimeh Farhadifard, and Uwe Freiherr von Lukas (Sept. 2016). "Dataset on underwater change detection". In: *OCEANS 2016 MTS/IEEE Monterey*, pp. 1–8. DOI: 10.1109/OCEANS.2016.7761129.

Ramaiah, V Subba et al. (2021). "A real-time High-Quality image snow removing using Adaptive Matched filter". In: *Turkish Journal of Computer and Mathematics Education (TURCOMAT)* 12.10, pp. 4741–4749.

Robbins, Herbert and Sutton Monro (1951). "A Stochastic Approximation Method". In: *The Annals of Mathematical Statistics* 22.3, pp. 400–407. DOI: 10.1214/aoms/1177729586. URL: https://doi.org/10.1214/aoms/1177729586.

Rosten, Edward and Tom Drummond (July 2006). "Machine Learning for High-Speed Corner Detection". In: vol. 3951. ISBN: 978-3-540-33832-1. DOI: 10.1007/11744023_34.

Rublee, Ethan et al. (2011). "ORB: An efficient alternative to SIFT or SURF". In: *2011 International Conference on Computer Vision*, pp. 2564–2571. DOI: 10.1109/ICCV.2011.6126544.

Ruder, Sebastian (2016). *An overview of gradient descent optimization algorithms*. DOI: 10.48550/ARXIV.1609.04747. URL: https://arxiv.org/abs/1609.04747.

Santurkar, Shibani et al. (2018). *How Does Batch Normalization Help Optimization?* DOI: 10.48550/ARXIV.1805.11604. URL: https://arxiv.org/abs/1805.11604.

Sato, Yuya, Takumi Ueda, and Yuichi Tanaka (2021). *Marine Snow Removal Benchmarking Dataset*. arXiv: 2103.14249 [cs.CV].

Segl, K. and H. Kaufmann (Sept. 2001). "Detection of small objects from high-resolution panchromatic satellite imagery based on supervised image segmentation". In: *IEEE Transactions on Geoscience and Remote Sensing* 39.9, pp. 2080–2083. ISSN: 1558-0644. DOI: 10.1109/36.951105.

Sen, Prithwish, Anindita Das, and Nilkanta Sahu (2021). "Rendering Scenes for Simulating Adverse Weather Conditions". In: *Advances in Computational Intelligence*. Ed. by Ignacio Rojas, Gonzalo Joya, and Andreu Català. Cham: Springer International Publishing, pp. 347–358. ISBN: 978-3-030-85030-2.

Shanmugasundaram, M., S. Sukumaran, and N. Shanmugavadivu (Aug. 2013). "Fusion based denoise-engine for underwater images using curvelet transform". In: *2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 941–946. DOI: 10.1109/ICACCI.2013.6637303.

Szegedy, Christian et al. (2016). *Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning*. DOI: 10.48550/ARXIV.1602.07261. URL: https://arxiv.org/abs/1602.07261.

Takeki, Akito et al. (Sept. 2016). "Detection of small birds in large images by combining a deep detector with semantic segmentation". In: *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 3977–3981. DOI: 10.1109/ICIP.2016.7533106.

Teixeira, Bernardo et al. (2020). “Deep Learning for Underwater Visual Odometry Estimation”. In: *IEEE Access* 8, pp. 44687–44701.

Voronin, V. et al. (2019). “Rain and snow removal using multi-guided filter and anisotropic gradient in the quaternion framework”. In: *Artificial Intelligence and Machine Learning in Defense Applications*. Ed. by Judith Dijk. Vol. 11169. International Society for Optics and Photonics. SPIE, pp. 227–233. DOI: 10.1117/12.2534744. URL: https://doi.org/10.1117/12.2534744.

Wang, Changbo et al. (May 2006). “Real-time snowing simulation”. In: *The Visual Computer* 22.5, pp. 315–323. ISSN: 1432-2315. DOI: 10.1007/s00371-006-0012-8. URL: https://doi.org/10.1007/s00371-006-0012-8.

Wang, Hong et al. (June 2020). “A Model-Driven Deep Neural Network for Single Image Rain Removal”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Wang, Wenshan et al. (2020). “TartanAir: A Dataset to Push the Limits of Visual SLAM”. In.

Wang, Yaqian et al. (2021). “Underwater image enhancement and marine snow removal for fishery based on integrated dual-channel neural network”. In: *Computers and Electronics in Agriculture* 186, p. 106182. ISSN: 0168-1699. DOI: https://doi.org/10.1016/j.compag.2021.106182. URL: https://www.sciencedirect.com/science/article/pii/S016816992100199X.

Wei, Xiaoming et al. (July 2003). “Blowing in the wind”. In.

Xu, Jing et al. (May 2012). “Removing rain and snow in a single image using guided filter”. In: *2012 IEEE International Conference on Computer Science and Automation Engineering (CSAE)*. Vol. 2, pp. 304–307. DOI: 10.1109/CSAE.2012.6272780.

Zheng, Xianhui et al. (2013). “Single-Image-Based Rain and Snow Removal Using Multi-guided Filter”. In: *Neural Information Processing*. Ed. by Minho Lee et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 258–265. ISBN: 978-3-642-42051-1.

Zou, Chengming, Xiufeng Xie, and Guanghui Zhao (2010). “Algorithm for generating snow based on GPU”. In: *ICIMCS '10*.

Zwilgmeyer, Peder Georg Olofsson et al. (Oct. 2021). “The VAROS Synthetic Underwater Data Set: Towards Realistic Multi-Sensor Underwater Data With Ground Truth”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pp. 3722–3730.

Appendices

Appendix A

Conference paper

On the next page we include our conference paper *Detecting and Suppressing Marine Snow for Underwater Visual SLAM* which was presented at the CVPR 2022 Image Matching workshop and published to IEEE Xplore/CVF and the CVPR conference proceedings.

Detecting and Suppressing Marine Snow for Underwater Visual SLAM

Lars Martin Hodne[1*] Eirik Leikvoll[1*] Mauhing Yip[2] Andreas Langeland Teigen[1]
Annette Stahl[2] Rudolf Mester[1]

[1]Department of Information and Computer Sciences [2]Department of Engineering Cybernetics
Norwegian University of Science and Technology

Abstract

Conventional SLAM methods which work very well in typical above-water situations, are based on detecting keypoints that are tracked between images, from which egomotion and the 3D structure of the scene are estimated. However, in underwater environments with marine snow — small particles of organic matter which are carried by ocean currents throughout the water column — keypoint detectors are prone to detect the marine snow particles. As the vast majority of SLAM front ends are sensitive against outliers, and the marine snow acts as severe "motion noise", failure of the regular egomotion and 3D structure estimation is expected. For this reason, we investigate the structure and appearance of marine snow and developed two schemes which classify keypoints into "marine snow" or "clean" based on either the image patches obtained from usual keypoint detectors or the descriptors computed from these patches. This way the subsequent SLAM pipeline is protected against 'false' keypoints. We quantitatively evaluate the performance of our marine snow classifier on both real underwater video scenes as well as on simulated underwater footage that contains marine snow. These simulated image sequences have been created by extracting real marine snow elements from real underwater footage, and subsequently overlaying these on "clean" underwater videos. Qualitative evaluation is also done on a night-time road sequence with snowfall to demonstrate applicability in other areas of autonomy. We furthermore evaluate the performance and the effect of marine snow detection & suppression by integrating the snow suppression module in a full SLAM pipeline based on the pySLAM system.

1. Introduction

When applied to underwater scenarios, Visual Odometry and Simultanous Localisation And Mapping (SLAM) face numerous challenges which appear significantly less frequently in regular above-water applications. Such challenges are *e.g.* moving illumination and the reduced zone of usable image landmarks, limited by the illumination cone and the achievable depth of field in a low illumination, turbid environment. Marine snow, the challenge in focus in this paper, describes particles present throughout the water column, ranging from millimeter scale up to decimeter scale [1]. As its name suggests, marine snow can have the appearance of snowfall; its movement is heavily influenced by ocean currents, and under illumination it fills its surroundings with bright white spots. This combination of dynamic motion and an appearance which contrasts with most backgrounds makes the snowflakes salient for keypoint de-

Figure 1. In our approach, we extract marine snow from underwater footage with untextured background (top), and superimpose this snow on arbitrary footage to create labelled training data (bottom images) for training snowflake detectors .

*These authors contributed equally

tectors and constitutes a significant source of motion noise. Thus, keypoint-based SLAM runs a significant risk of producing wrong egomotion estimates or even tracking failure if the number of snowflakes detected, and thus the outlier rate, becomes too high. Of course, this issue also exists for perception in case of heavy snowfall in autonomous driving.

In this paper, we present our approach to mitigate the effect of marine snow by developing two machine-learning systems to filter 'false' keypoints. The main contributions of this paper are:

- We developed two efficient classifiers for marine snow, P-CLAS and D-CLAS; they are designed to run in piggy-back mode on top of arbitrary keypoint detectors — this is done to limit processing to image areas which are actually candidates for being regarded as keypoints. While classifier P-CLAS works on the image area around the detected keypoint, the second classifier, D-CLAS, works on the binary keypoint descriptors provided by the ORB [18] detector/descriptor.
- We investigate how the descriptor-based classifier D-CLAS (which is computationally 'cheaper' than the one working on image patches) compares in performance to the patch classifier P-CLAS. This comparison is done both on a large image dataset as well as for the case of being integrated into a SLAM pipeline.
- We provide a method to extract snow and marine snow from images with essentially untextured backgrounds. We have considerable underwater footage with this characteristic, and thus could collect a huge set of 'ground truth' marine snow examples. The resulting 'snowflake dataset' is used by us for superimposing marine snow on 'clean' images or video sequences. It is publicly available[1], and to our knowledge it is the first of its kind.
- We extend an existing underwater pose-estimation dataset (VAROS [20]) with superimposed marine snow to provide a new benchmark with marine snow motion noise, and accurate ground-truth values.
- We test our method on an above-water snowy sequence, and demonstrate that with further fine-tuning our results should be transferable to the above-water domain.

2. Related Work

Marine snow mitigation for computer vision tasks is a relatively recent research topic. Early methods aimed at a more broad form of image enhancement modelled marine snow as a simple form of additive noise, however, more recent methods aimed specifically at marine snow point out the weaknesses of this approach, like its disregard of properties such as water absorption, size, shape, and backscattering [4].

Most methods pursue marine snow removal in the interest of improving object detection pipelines, and therefore detect snow in the entire image. A family of filter-based approaches for marine snow detection and removal can be traced back to the work of Banerjee *et al*. [2]. It presents a basic approach which does snow removal using median filtering and implicit snow detection based on the luminance channel of a YCbCr (luminance, blue-difference, red-difference) image-representation. The image is traversed with a 7x7 window, and locations which have a high luminance center and high luminance variance are selected for marine snow removal. There exists an extension of this method with multi-scale filters to address particles of varying size, however further details are not given [17].

From Farhadifard *et al*. [7], we find another multi-scale approach, which like earlier filter-based methods uses the dissimilarity of the moving window center value to the window mean as a selection metric. To identify additional outliers within a patch, the patch is represented in RGB color-space, and an outlier detection step selects all pixels which are closer to the pixel-center than a threshold based on a weighted standard-deviation value. As a final criteria, high-saturation patches are considered false detections, and consequently removed due to the typically grayscale appearance of marine snow.

The paper [6] highlights and addresses one shared shortcoming of the aforementioned methods, namely their implicit dismissal of the temporal information present in video sequences. Allegedly, this is the first spatio-temporal marine snow removal method. From three input frames, the method detects and removes snow in the center frame.

In a 2021 paper, Sato *et al*. [19] state that they are unaware of any deep learning based marine snow removal methods. However, neural networks have been used in an intermediate marine snow detection step before filter-based removal [12]. This method considers the temporal nature of marine snow by utilising 3D neural networks. Their architecture first detects snow using a combination of 3D and 2D convolutions, before using adaptive median filtering to remove the snow.

In another paper [14], the authors do multi-scale detection and removal of above water snow. Their system is divided into three main parts. First, feature maps are calculated at three different scales using a multi-scale Convolutional Neural Network (CNN). Next, the feature maps are concatenated and fed through the snow detection module—a 40-layer modified DenseNet. Finally, to remove the snow, the output from the snow detection module is concatenated with the feature maps from the multi-scale CNN and passed

[1] https://www.ntnu.edu/arosvisiongroup/varos

through yet another densely connected CNN.

With our focus on keypoint classification, performing snow-detection on the full image entails a significant amount of unnecessary computation. A more efficient approach is to only focus on those specific points which are used by the affected SLAM pipeline, *i.e.* the keypoints given by its keypoint detector. Keypoint rejection of stand-alone keypoints is somewhat rare, as most outlier rejection methods are based on a set of matched keypoints from a pair of images. For example, the seminal paper [8] introduces RANSAC, an outlier rejection method which creates motion hypotheses from different subsets of the keypoint correspondences, and tests these estimates against the remaining data.

However, waiting until the matching step is complete wastes resources on matching keypoints which should be removed either way. Therefore, rejecting keypoints as soon as possible, or not detecting them at all should be the preferred method. The workshop paper [9] uses random forest classifiers to predict the suitability of the keypoints for matching, thereby implicitly evaluating keypoints for pose-estimation. Their implementation uses a random forest with 25 decision trees with a maximum tree depth of 25. Importantly, their method classifies on the keypoint descriptors, meaning there is no additional cost related to feature extraction before classification. In scenes with high amounts of foliage or dynamic objects their method removed 70% of the keypoints while retaining 60% of the matches.

The conference paper [10], uses a Support Vector Machine (SVM) to predict the suitability of an image region for image retrieval in geo-localization. To increase the discriminative power of the input, they perform classification based on bundles of descriptors retrieved from the same local image region.

Another paper [13] does keypoint rejection in underwater images for lighting artefacts and dynamic phenomena, such as fishes and caustics, as well as marine snow. They take a 257×257 image patch around each keypoint and scale them down to 65×65. Each patch is classified by a CNN as either suitable or unsuitable for tracking. Their architecture consists of a shallow network with three convolutional layers with ReLU and maxpooling, followed by a fully connected layer and a soft-max layer. Training is supervised, with manually labelled images from other datasets. The proposed real time plug-and-play keypoint rejection system has been verified by comparing drift accumulated by ORB-SLAM [16] and DynaSLAM [3], a SLAM system which accounts for dynamic environments.

3. The ANN-based Approaches to Snow Classification

We created two neural networks for snow classification of keypoints, P-CLAS which extracts image patches from keypoints based on their coordinates such that it can be used with any keypoint-based pipeline, and D-CLAS with descriptors as input, meaning it must be trained for the particular descriptor it is to be combined with. Common to both methods is the Sigmoid activation function in their last layer which makes the final output a pseudo-probability estimate for membership of the positive (snow) class.

Layer	L1	L2	L3	L4	L5	L6	L7
Activation	ReLu						Sigmoid
Dimensionality	256	196	196	128	64	16	1

Table 1. Neural Network architecture for the descriptor classifier D-CLAS

Layer	L1	L2	L3	L4	L5	L6
Layer type	CNN w/ 3x3 filters					Dense
Activation	ReLu					Sigmoid
Input Depth	9	32	64	64	64	256
Input Height/Width	64	32	16	8	4	N/A

Table 2. Neural Network architecture for the image-patch classifier P-CLAS

D-CLAS was designed for ORB-descriptors with a Fully Connected Neural Network (FCNN) architecture (cf. Table 1).

P-CLAS is structured as a multi-scale CNN + FCNN architecture (cf. Table 2). The multi-scale input allows the classifier to discriminate marine snow of different sizes, which is significant because marine snow can vary from a few pixels to 50×50 image regions which can contain multiple undesirable keypoints. With keypoint coordinates as input, we extract patches at three scales ([64×64, 48×48, 32×32]) and, using bilinear interpolation, rescale and subsequently stack them to create a $9 \times 64 \times 64$ input. The network has 5 layers with ReLU, BatchNorm, and maxpooling, and a 6th dense layer with 1 neuron.

Both networks were trained with the Adam optimizer [11]. During training, we frequently validate on the validation splits of the datasets. The models with the highest F2-score in validation were saved for further evaluation.

4. Datasets

We developed our own datasets for training and evaluation. We first collected underwater sequences in which all features are either suitable for SLAM, or all features are marine snow. This means sequences near the ocean floor with no visible marine snow, and sequences distant from both the ocean surface and the ocean floor, in which only marine snow is visible and nothing else. Such sequences circumvent the need to manually label marine snow, which can easily amount to thousands of samples per frame.

With these images, we generate four datasets with full

HD images and keypoints and descriptors labeled as "snow" and "clean". The first Unmodified (U) dataset uses the images as-is to detect keypoints and store their coordinates and descriptors. However, the U dataset has some notable caveats. First and foremost, the presence of marine snow can easily be determined by the colour and texture of the image, since all images of marine snow inevitably come with a background with various shades of blue. Conveniently, we can use these untextured backgrounds to reliably extract marine snow and superimposing it onto more varied backgrounds, using a weighted sum (alpha-keying). With the extracted snow, we create three datasets named Underwater (UW), Overwater (OW), and Snowy-VAROS. These are discussed later in this section.

To extract snow, we use a strided window approach with stride 10. In each 60×60 window, P, we calculate the Euclidean RGB-distance, D, of each individual pixel value at location $p \in P$ to the median colour M_P of the window. As indicated in Eqs. (1) and (2), these distances are scaled by the inverse maximum distance to create a pixel-wise weighting between 0 and 1. The weight, W_p, is set to 0 if $D(M_P, p)$ is below a threshold value $\tau_D = 30$, or if the grayscale intensity of p, $I_{GS}(p)$, is below $\tau_I = 20$.

$$W_p = \begin{cases} 0 & \text{if } I_{GS}(p) < \tau_I \\ 0 & \text{if } D(M_P, p) < \tau_D \\ \dfrac{D(M_P, p)}{\max\limits_q D(M_P, q)} & \text{otherwise.} \end{cases} \quad (1)$$

where

$$D(p, q) = |I(p) - I(q)|. \quad (2)$$

For each pixel, the average weight across all windows is used when extracting snow. This is to ensure that windows with large marine snow particles which shift the median colour away from the background colour do not introduce unwanted artefacts when superimposing the snow. With a background image B, alpha-key weight W, and snowy image S, we superimpose images according to Equation 3:

$$I = B \odot (1 - W) + S \odot W. \quad (3)$$

After extracting snow from the snowy images in the U dataset, we create the three other datasets. The Underwater (UW) dataset superimposes the extracted snow onto the remaining images in U which are free from snow. The Overwater (OW) dataset uses above-water images from the *Exclusively Dark Images Dataset* (ExDark) [15] as backgrounds for superimposing to introduce more variation in the background images. Images in the ExDark dataset which featured rain, starry skies, or snowfall were removed because of their exceptional similarities to marine snow.

Finally, to demonstrate the flexibility of our superimposing approach we use it to add a video sequence of snow

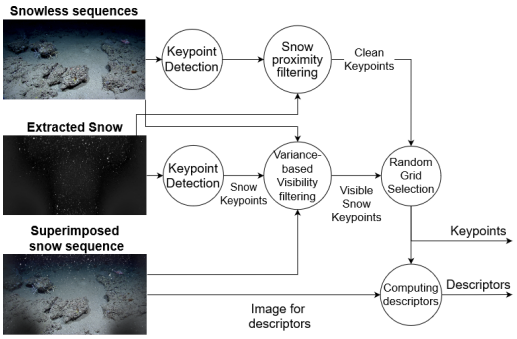


Figure 2. Data generation pipeline with superimposed snowy sequences

Figure 3. Snow superimposed on the VAROS dataset

to the synthetic underwater benchmarking dataset VAROS [20]. This has the benefit of a known camera matrix and ground-truth pose. By superimposing snow from a video sequence, we ensure that the motion of the snow is consistent between frames. A sample frame from this Snowy-VAROS sequence is seen in Figure 3.

To create keypoint coordinates and descriptors for training, we perform keypoint detection separately on the background image, and extracted snow image, as opposed to detecting keypoints on the combined, superimposed image. This was done primarily to increase the number of keypoints on marine snow, and because it makes for a more difficult dataset in which some of the detected snow is less visible than normal. Figure 2 presents the pipeline used to generate keypoints for the datasets from a three-tuple of extracted snow, background, and their combined image. A challenge of this approach is that snow can be superimposed either over a good keypoint in the background image or in an image region where the snow is not visible, meaning keypoints can become mis-labeled in the combined image. Consequently, a keypoint, K_s, detected on the extracted snow is rejected if inequality 4 is not true, where

	Images	Snow KPs	Background KPs
Unmodified	6,008	598,931	1,181,570
Underwater	10,051	1,525,556	2,227,102
Overwater	8,705	1,772,123	2,055,001
Total	24,764	3,896,610	5,463,673

Table 3. Datasets and their sizes. Train, val and test splits were made following the 80/10/10 convention

P_{Si}, and P_{BG} are image patches of K_s in the superimposed image and background image, respectively, and $\mathcal{E} = 14$ is an empirically selected threshold.

$$\mathsf{Var}\left[P_{Si}\right] > \mathsf{Var}\left[P_{BG}\right] + \mathcal{E} \tag{4}$$

Secondly, to verify that keypoints detected on the background image can not be perceived as mis-labelled after superimposing due to abutting snow, we verify that the maximum color channel value of a small 8×8 neighbourhood surrounding this keypoint within the extracted snow is below an empirically selected threshold, $\tau_S = 70$. Finally, we divide the image into a 10×10 grid and select keypoints at random from these bins to limit the dataset size, and to reduce the presence of overlapping samples. Importantly, the ORB descriptors are still generated on the combined image.

5. Experiments

We conducted experiments to evaluate stand-alone classification performance, and performance in SLAM use cases. For stand-alone performance, we used test splits of our U, OW, and UW datasets and evaluated F1 score, accuracy, True Positive Rate (TPR) and True Negative Rate (TNR). The datasets and their sizes are listed in Table 3.

Qualitative assessments of keypoint classification were performed on four diverse underwater sequences, each pictured in Figure 4, by extracting 2000 keypoints with the ORB detector and classifying these frame-by-frame.

For evaluation in SLAM use-cases, we implemented our classifiers into the pySLAM framework[2] which offers a very customiseable SLAM-platform intended for experimentation and education. pySLAM features most of the expected attributes of a modern SLAM-system, including keyframe management, local and global bundle adjustment, outlier rejection with RANSAC, ratio testing, and motion models with active matching [5]. Our experiments in pySLAM were done on the synthetic VAROS and Snowy-VAROS sequences.

5.1. Binary classification metrics

While training classifiers, we store the checkpoint which achieved the best F2 score on the validation data-split. In Table 4, we list these models, and their binary classification metrics on the separate test-splits of our datasets.

It is clear that both D-CLAS and P-CLAS have learned the classification task successfully, yet P-CLAS maintains remarkable results on most datasets, outperforming the descriptor classifier in all datasets. However, D-CLAS has an unavoidable benefit in that it requires no pre-processing of the image if descriptors are present, and can operate far more efficiently, surpassing speeds of 66000 keypoints per second, compared to 14600 for P-CLAS, both on a GTX 1080 GPU. However, our testing shows that these differences can be explained by overhead from patch-extraction, which can be improved compared to our implementation since it assumes that keypoints in the same batch come from different images, which is true for training, but otherwise is typically false.

Both classifiers, when trained on the U dataset, score high on the U test-split. However, we notice a decrease in TPR when the superimposed OW and UW datasets are included in the test data. This strongly suggests that U-trained models only learn to recognise white blobs on an untextured background, hence when more textured backgrounds appear, the number of false negatives increase. P-CLAS in particular, seems to rely too much on the predictable backgrounds of the U training data, since its TNR is high on both the U testset, and the unmodified + UW testset. This suggests that training on varied backgrounds, and thus the superimposed datsets, is particularly important for P-CLAS models, since they will otherwise latch onto background characteristics which are not encoded by the descriptors. To be clear, the near perfect scores on U, highlight the simplicity of the U datasets, rather than the prowess of the methods.

On the topic of what the descriptor encodes, it is feasible that the discrepancy which is consistently present in all testsets between P-CLAS and D-CLAS can be explained by the CNN being able to use more contextual clues from the background which are not available from descriptors. This could lead to P-CLAS models performing better than D-CLAS models when encountering backgrounds familiar from training, but worse on unfamiliar backgrounds.

When it comes to networks trained on superimposed data, the models which were trained on all datasets performed the best on every testset, except the U testset. Even if the test dataset only included two of the three datasets used in training, training on every dataset gave the best overall performance which could indicate an improved ability to generalise to unseen data.

5.2. Qualitative results in Keypoint Classification

We begin with video A) in Figure 4, which features a smooth ocean floor with small mounds of sand, and a somewhat dense cover of small and bright marine snow. Some spots on the ground can be mistaken for marine snow in still

[2]https://github.com/luigifreda/pyslam

		Unmodified				UW + U				OW + U				All datasets			
		F1	Acc	TPR	TNR	F1	Acc	TPR	TNR	F1	Acc	TPR	TNR	F1	Acc	TPR	TNR
Patch	U	**0.999**	**0.999**	**0.999**	**1.0**	0.617	0.792	0.446	0.999	0.47	0.687	0.309	**0.997**	0.402	0.692	0.252	**0.998**
	UW + U	0.968	0.972	0.991	0.957	0.948	0.961	0.931	0.98	0.854	0.861	0.904	0.825	0.877	0.897	0.894	0.9
	OW + U	0.998	0.998	0.996	0.999	0.913	0.94	0.84	**0.999**	0.94	0.948	0.894	0.993	0.91	0.932	0.839	0.996
	All	0.996	0.996	0.996	0.997	**0.975**	**0.982**	**0.955**	0.998	**0.961**	**0.965**	**0.954**	0.975	**0.964**	**0.971**	**0.95**	0.985
Desc	U	0.945	0.951	0.98	0.929	0.778	0.848	0.712	0.929	0.809	0.833	0.784	0.873	0.763	0.82	0.707	0.899
	UW + U	0.944	0.949	0.978	0.927	0.913	0.933	0.931	0.935	0.892	0.898	0.943	0.86	0.893	0.909	0.93	0.895
	OW + U	0.954	0.959	0.977	0.946	0.916	0.937	0.917	0.949	0.917	0.925	0.919	0.93	0.909	0.926	0.906	0.939
	All	0.955	0.961	0.964	0.958	0.935	0.952	0.926	0.967	0.919	0.928	0.912	0.941	0.921	0.936	0.909	0.955

Table 4. Binary classification results by the classifiers. Rows denote the the network and its training data, while columns denote the test dataset. The Unmodified dataset (U), Underwater superimposed (UW), and Overwater superimposed (OW) were combined and used for testing. We provide F1-scores, accuracy, True Positive Rates (TPR), and True Negative Rates (TNR) for each case.

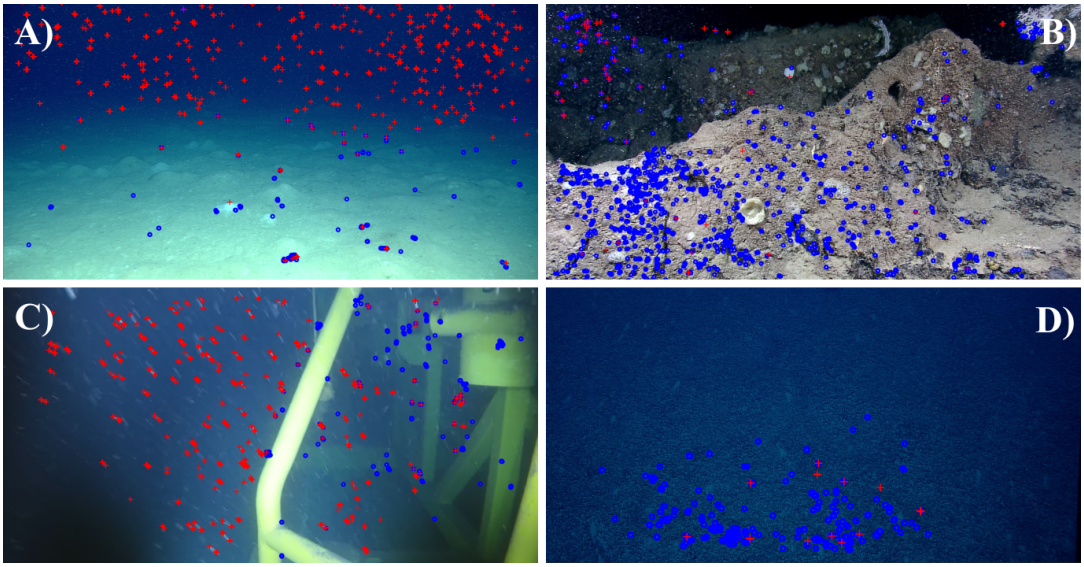


Figure 4. Frames from the four videos used for visualisation of keypoint classification. Red crosses indicate a snow classification, while blue circles indicate a clean keypoint classification. 2000 Keypoints were detected with ORB per frame.

images. Models tested on this video mainly struggled with classifying marine snow when transitioning between the textureless region in the upper half of the image, and the textured region below. With P-CLAS models, those trained on the U dataset and at least one superimposed dataset seemed to handle this issue the best. This was true also for D-CLAS models, though these showed slightly worse False Negative Rates, and False Positive Rates on the ground.

Video B), features a complex structure of large, jagged, and overlapping rocks. Of particular interest is the marine snow seen in the top left corner during the beginning of the sequence, which is strikingly bright and visible, despite the rocks in the background. Thus, this sequence offers a break from the typically textureless backgrounds which are far more common. Performance on this sequence was particularly bad from P-CLAS models trained only on the U-data, which labelled all keypoints as "clean". Other P-CLAS models trained on superimposed data were able to improve upon this, but none were able to compete with D-CLAS models, not even U-trained D-CLAS models. D-CLAS models were hard to tell apart, but it seems like the OW-trained model did worse, and both the U-trained model and OW-trained model did best. These results may be an indication that P-CLAS models rely more on properties of the background when classifying. False Positive Rates were very low for all models on this sequence.

Video C) features a bright yellow charging station, with extreme amounts of large marine snow particles. This sequence highlights a weakness of current feature detectors underwater, in the sense that the sequence's lack of corners

and blobs (other than marine snow) leaves most corner detectors with nearly no useful features, *e.g.* for pose estimation. Despite a limited presence of useful keypoints, the classifiers labelled many keypoints as clean. While TPR rates were good on this sequence, false negatives were also seen frequently. In the final part of this sequence, the robot moves rapidly, giving the marine snow a stretched appearance which is not present in the training data, leading to high amounts of false negatives. While all classifiers exhibited this trait, D-CLAS models were the worst afflicted.

Notably, models which were not trained exclusively on the OW dataset showed a tendency to switch from a true positive detection to a false negative when snow particles moved in front of an uncommon background, e.g., the yellow beams of the charger sequence. However, OW-trained models struggled more with classifying snow in textureless regions. All classifiers struggled with the sequence's largest snow particles which are particularly close to the camera.

As a final test on False Positive Rates, video D) deliberately features an insignificant amount of snow, yet in certain frames, the pebbled texture of the ocean floor carries an appearance somewhat (though not completely) reminiscent of marine snow. Most classifiers tested on this sequence, be it patch or descriptor based, were not prone to mislabel the ground as snow, with P-CLAS models generally achieving near-perfect accuracy, and D-CLAS models not far behind. However, the P-CLAS model which was trained on both the underwater superimposed data and unmodified data was a curious exception. Once the keypoint detector began detecting on the ground, most keypoints were classified incorrectly as snow. This continued as the camera came closer and the number of ground keypoints increased, but eventually stopped once the robot came even closer to the ground and the likeness to marine snow disappeared.

To summarise these results, P-CLAS models typically perform better than D-CLAS ones if the background is known from training. With textured backgrounds, performance drops off, in which case training with superimposed datasets can help, but not completely. Compared to the image-patches, ORB-descriptors seem to encode less information about the background, which can remove irrelevant information, and in certain instances help classification, such as in video B) where D-CLAS models consistently outperformed P-CLAS models. However, it could be the case that too much information is lost through the ORB representation, such that overall performance is reduced.

5.3. Qualitative results in an above-water sequence

To examine generalisability and applicability on a broader range of tasks, we visualised classification on a night-time road sequence with real snowfall, using an OW+U-trained P-CLAS model. The sequence features a twisting, snow-covered road with dimly lit trees on both sides and snowfall illuminated by the car's headlights. At the bottom of the image is the contour of the car's dashboard. Performance on this sequence was mixed, but showed some promise. Clean points placed on the roadside, dashboard, and trees are typically classified correctly. The same goes for snow keypoints in the darker regions of the image. However, one struggle of the classifiers is the snow just in front of the right headlight which is particularly bright in front of a white background. This kind of image patch is not found in the training data, so unsurprisingly it is classified incorrectly. However, considering the overall performance on this sequence it seems probable that given finetuning on above-water data, our results should be transferable to the road domain as well. Generally, in above-water scenarios snow often appears on more textured backgrounds, which can be a source of decreased performance not covered by this particular video. On the other hand, increased illumination during the daytime can make the snow less prominent in some footage.

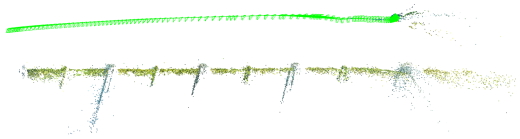


Figure 5. A point map from running pySLAM on VAROS without snow and no classification. The pipe is properly tracked.

5.4. Qualitative results with pySLAM

Testing SLAM performance on real-world sequences has the potential to give the most realistic view of the effect of snow classification. However, by using the synthetic VAROS sequence with and without superimposed snow, we are able to control the difficulty of both the background sequence and snow conditions. Furthermore, we are able to compare results between Snowy-VAROS and the original, snow-free VAROS sequence which lets us evaluate the results of keypoint rejection more definitively than most qualitative tests. However, we must expect that models trained on superimposed images perform disproportionally better on Snowy-VAROS, due to similarities in the superimposing process of Snowy-VAROS and the training datasets. We choose a subsequence of VAROS in which the robot travels adjacent to a straight pipe (see Fig. 3). This pipe offers more defined features for keypoint detection compared to other sections of VAROS, and makes it easy to judge the tracking quality by how accurately the straight pipe is mapped.

When testing with pySLAM alone on the Snowy-VAROS sequence, a considerable amount of keypoints are detected on snow, which lead to rapid tracking failure and inconsistent behaviour between runs. During some runs,

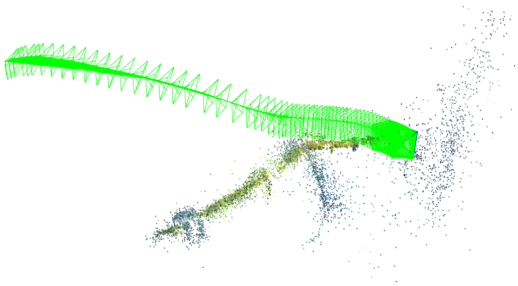


Figure 6. A point map from running pySLAM without keypoint classification on Snowy-VAROS. The path stops in a wall of snow and the pipe appears to bend, unlike the source video.

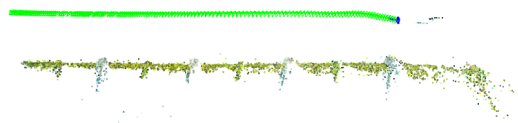


Figure 7. A point map from running pySLAM with keypoint classification on Snowy-VAROS. The pipe is properly tracked, and few snow keypoints are added to the map.

tracking fails completely, while in other runs the tracking is closer to the movement seen in the sequence. When visualising the sparse map made by pySLAM, seen in Figure 6, we see what looks like a wall of snow prominently in the map. Furthermore, the pipe which is completely straight in the sequence appears bent in the point cloud.

Both P-CLAS and D-CLAS stabilised tracking in pySLAM, to the extent that they were difficult to tell apart. While they were unable to remove all unreliable points, pySLAM continued tracking for far longer and was far more reliable, giving consistent tracking outputs between runs. An example can be seen in Figure 7, where the pipe appears straight in the point cloud like it should, with the exception of the very end. This behaviour is similar to that seen in the VAROS sequence without snow and no classification, as seen in Figure 5, and occurs because pySLAM is unable to detect a sufficient amount of good keypoints, irrespective of the presence of snow. Since P-CLAS and D-CLAS differed in earlier testing, their comparable performance with pySLAM could indicate that as long as the number of marine snow keypoints is reduced such that the snow is no longer dominating the RANSAC motion hypotheses, tracking can continue with traditional outlier rejection. On Snowy-VAROS, out of 3000 features, D-CLAS removed 1,627 keypoints and P-CLAS removed 1,366 keypoints in each frame on average.

For comparison, we run pySLAM on the original VAROS dataset, which has nothing resembling marine snow. With snow rejection enabled on the unmodified VAROS sequence, out of 3,000 keypoints, we see on average 36 and 201 rejections, *i.e.*, false positives, by D-CLAS and P-CLAS, respectively.

6. Conclusion

In this paper we have demonstrated two methods for classification of keypoints obtained from the ORB detector [18] in order to suppress the effect of marine snow. The methods can be used to aid pose estimation, create keypoint detection masks or assist in underwater image restoration. Our results show that classifying snow, either with ORB descriptors or image patches, can achieve near perfect performance for snow in front of an untextured background. To enable snow detection also on textured backgrounds, additional training data is necessary. We created such data by extracting snow from underwater footage with untextured background. This allowed us to overlay real marine snow on arbitrary image material. Despite a lack of training on such scenes, initial experiments on a night-time driving sequence featuring snowfall suggest that the classifiers can be applied in above-water scenarios with some further finetuning. Using the pySLAM framework we demonstrated how our method can be incorporated as a keypoint rejection component in a SLAM pipeline. We showed that our methods were able to overcome the difficulties that a SLAM system with standard outlier removal has with underwater footage affected by marine snow. We provide the snow dataset to the public in order to foster further research on the challenging topic of underwater and above-water SLAM under difficult visibility conditions. Extensions of our research could examine other descriptors than ORB, novel classifiers, and new methods of extracting snow.

References

[1] Alice L. Alldredge and Mary W. Silver. Characteristics, dynamics and significance of marine snow. *Progress in Oceanography*, 20(1):41–82, 1988. 1

[2] Soma Banerjee, Gautam Sanyal, Shatadal Ghosh, Ranjit Ray, and Sankar Nath Shome. Elimination of marine snow effect from underwater image - an adaptive probabilistic approach. In *2014 IEEE Students' Conference on Electrical, Electronics and Computer Science*, pages 1–4, 2014. 2

[3] Berta Bescos, José M. Fácil, Javier Civera, and José Neira. Dynaslam: Tracking, mapping, and inpainting in dynamic scenes. *IEEE Robotics and Automation Letters*, 3(4):4076–4083, 2018. 3

[4] Matthieu Boffety and Frédéric Galland. Phenomenological marine snow model for optical underwater image simulation:

This paper is financially supported by the Norwegian Research Council in the project Autonomous Robots for Ocean Sustainability (AROS), project number 304667.

Applications to color restoration. In *2012 Oceans - Yeosu*, pages 1–6, May 2012. 2

[5] Margarita Chli and Andrew Davison. Active matching. pages 72–85, 10 2008. 5

[6] Boguslaw Cyganek and Karol Gongola. Real-time marine snow noise removal from underwater video sequences. *Journal of Electronic Imaging*, 27:1, 07 2018. 2

[7] Fahimeh Farhadifard, Martin Radolko, and Uwe von Lukas. Single image marine snow removal based on a supervised median filtering scheme. In *VISIGRAPP*, 2017. 2

[8] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, jun 1981. 3

[9] Wilfried Hartmann, Michal Havlena, and Konrad Schindler. Predicting matchability. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6 2014. 3

[10] Hyo Jin Kim, Enrique Dunn, and Jan-Michael Frahm. Predicting good features for image geo-localization using per-bundle vlad. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015. 3

[11] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 3

[12] Michał Koziarski and Bogusław Cyganek. Marine snow removal using a fully convolutional 3d neural network combined with an adaptive median filter. In Zhaoxiang Zhang, David Suter, Yingli Tian, Alexandra Branzan Albu, Nicolas Sidère, and Hugo Jair Escalante, editors, *Pattern Recognition and Information Forensics*, pages 16–25, Cham, 2019. Springer International Publishing. 2

[13] Marco Leonardi, Luca Fiori, and Annette Stahl. Deep learning based keypoint rejection system for underwater visual ego-motion estimation. *IFAC-PapersOnLine*, 53(2):9471–9477, 2020. 21th IFAC World Congress. 3

[14] Pengyue Li, Mengshen Yun, Jiandong Tian, Yandong Tang, Guolin Wang, and Chengdong Wu. Stacked dense networks for single-image snow removal. *Neurocomputing*, 367:152–163, 2019. 2

[15] Yuen Peng Loh and Chee Seng Chan. Getting to know low-light images with the exclusively dark dataset. *Computer Vision and Image Understanding*, 178:30–42, 2019. 4

[16] Raúl Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015. 3

[17] Martin Radolko, Fahimeh Farhadifard, and Uwe Freiherr von Lukas. Dataset on underwater change detection. In *OCEANS 2016 MTS/IEEE Monterey*, pages 1–8, 9 2016. 2

[18] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011. 2, 8

[19] Yuya Sato, Takumi Ueda, and Yuichi Tanaka. Marine snow removal benchmarking dataset, 2021. 2

[20] Peder Georg Olofsson Zwilgmeyer, Mauhing Yip, Andreas Langeland Teigen, Rudolf Mester, and Annette Stahl. The varos synthetic underwater data set: Towards realistic multi-sensor underwater data with ground truth. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pages 3722–3730, 10 2021. 2, 4

Appendix B

Additional Results

		U				UW+U				OW+U				All datasets			
		F1	Acc	TPR	TNR	F1	Acc	TPR	TNR	F1	Acc	TPR	TNR	F1	Acc	TPR	TNR
ORB	U	0.945	0.95	0.984	0.923	0.778	0.848	0.718	0.924	0.814	0.833	0.789	0.87	0.77	0.821	0.719	0.893
	UW + U	0.94	0.946	0.98	0.92	0.907	0.929	**0.934**	0.927	0.901	0.902	**0.96**	0.852	0.897	0.91	**0.944**	0.885
	OW + U	0.956	0.961	0.975	0.95	0.921	0.942	0.921	0.954	**0.939**	**0.943**	0.951	0.935	**0.927**	**0.939**	0.932	0.944
	All	0.953	0.959	0.977	0.945	**0.924**	**0.943**	0.929	0.951	0.934	0.938	0.957	0.921	0.926	0.937	0.94	0.935
FREAK	U	**0.981**	**0.984**	0.992	**0.979**	0.716	0.854	0.584	**0.979**	0.667	0.777	0.528	0.96	0.61	0.784	0.463	**0.968**
	UW + U	0.922	0.931	**0.997**	0.885	0.766	0.852	0.769	0.89	0.808	0.837	0.808	0.858	0.762	0.829	0.752	0.873
	OW + U	0.925	0.933	**0.997**	0.889	0.772	0.855	0.778	0.89	0.846	0.869	0.848	0.884	0.792	0.85	0.784	0.887
	All	0.916	0.925	0.996	0.876	0.771	0.852	0.79	0.881	0.844	0.865	0.856	0.872	0.792	0.847	0.796	0.877
SIFT	U	0.935	0.942	0.965	0.925	0.607	0.766	0.488	0.929	0.528	0.684	0.382	0.943	0.466	0.684	0.33	0.938
	UW + U	0.862	0.869	0.941	0.814	0.775	0.826	0.809	0.836	0.793	0.807	0.796	0.817	0.77	0.807	0.777	0.829
	OW + U	0.889	0.897	0.942	0.863	0.773	0.835	0.759	0.88	0.823	0.843	0.785	0.894	0.784	0.829	0.744	0.891
	All	0.851	0.855	0.949	0.784	0.778	0.822	0.841	0.811	0.83	0.838	0.855	0.823	0.8	0.826	0.831	0.822
VGG	U	0.978	0.981	0.988	0.975	0.655	0.8	0.512	0.969	0.652	0.75	0.505	**0.961**	0.573	0.738	0.422	0.963
	UW + U	0.935	0.942	0.957	0.931	0.88	0.912	0.872	0.936	0.87	0.879	0.87	0.887	0.863	0.887	0.855	0.909
	OW + U	0.939	0.946	0.968	0.929	0.864	0.901	0.855	0.927	0.905	0.913	0.898	0.926	0.879	0.901	0.864	0.926
	All	0.928	0.935	0.976	0.903	0.875	0.904	0.9	0.907	0.903	0.909	0.922	0.897	0.884	0.902	0.902	0.902

Table B.1: Binary classification results with different descriptors. Rows denote the descriptor and its training data, while columns denote the test dataset. The Unmodified dataset (U), Underwater superimposed (UW), and Overwater superimposed (OW) were combined and used for testing. We provide F1-scores, accuracy, True Positive Rates (TPR), and True Negative Rates (TNR) for each case.

Appendix C

Videos from NOAA

C.1 NOAA Video collection

The following files were collected as sources for our datasets:

- EX1004L2_VID_20100629T051417Z_ROVHD_ROCK_CROP.mov - Accessed Feb 7 2022
- EX1004L3_VID_20100724T034634Z_ROVHD_IRON_ROCKS_DETAIL.mov - Accessed Feb 7 2022
- EX1202L2_VID_20120322T161540Z_ROVHD_BLK_BELLY_ROSE_CR.mov - Accessed Feb 7 2022
- EX1304L1_VID_20130724T191508Z_ROVHD_AUD_EXPED_SUMMARY.mov - Accessed Feb 7 2022
- EX1304L2_VID_20130803T195342Z_CPHD_BEAUTY.mov - Accessed Feb 7 2022
- EX1402L3_VID_20140412T174503Z_CPHD_ROV_CLOSE.mov - Accessed Feb 3 2022
- EX1402L3_VID_20140412T174846Z_CPHD_ROV_SURVEYS.mov - Accessed Feb 3 2022
- EX1402L3_VID_20140412T175300Z_CPHD_ROV_OVER_CARBONATE.mov - Accessed Feb 3 2022
- EX1402L3_VID_20140427T170016Z_ROVHD_MARINE_SNOW_AUDIO.mov - Accessed Feb 3 2022
- EX1502L3_VID_20150410T151332Z_ROVHD_WOOD_ROCK_AUD.mov - Accessed Feb 7 2022

- EX1504L2_VID_20150802T201839Z_ROVHD_ROCK_SURVEY.mov - Accessed Feb 7 2022
- EX1504L2_VID_20150802T205836Z_ROVHD_PILLOW_ROCKS_PSD.mov - Accessed Feb 7 2022
- EX1504L2_VID_20150803T233310Z_ROVHD_COR_ROCK_AUD.mov - Accessed Feb 7 2022
- EX1504L2_VID_20150803T235626Z_ROVHD_ROCK_AUD.mov - Accessed Feb 7 2022
- EX1504L2_VID_20150804T000217Z_ROVHD_ROCK_AUD.mov - Accessed Feb 7 2022
- EX1504L2_VID_20150804T011544Z_ROVHD_ROCK_LEDGE.mov - Accessed Feb 7 2022
- EX1504L2_VID_20150804T022335Z_PTMAN_ROCK_SAMPLE02_1.mov - Accessed Feb 7 2022
- EX1504L4_VID_20150913T192415Z_ROVHD_ROCK_COR_ACN.mov - Accessed Feb 7 2022
- EX1504L4_VID_20150913T194155Z_ROVHD_ROCK_SURVEY.mov - Accessed Feb 7 2022
- EX1504L4_VID_20150914T001624Z_ROVHD_AUD_ROCK_CPEN.mov - Accessed Feb 7 2022
- EX1603_VID_20160228T202604Z_ROVHD_ROCK_AUD.mov - Accessed Feb 7 2022
- EX1603_VID_20160229T000408Z_ROVHD_ROCK_CRACK.mov - Accessed Feb 7 2022
- EX1605L1_VID_20160421T012731Z_CPHD_FIS_05.mov - Accessed Feb 3 2022
- EX1605L1_VID_20160421T040610Z_CPHD_ROV_COR_05.mov - Accessed Feb 3 2022
- EX1606_VID_20160729T231121Z_PTMAN_ROCK_COR_SPO.mov - Accessed Feb 7 2022
- EX1606_VID_20160729T235319Z_ROVHD_ROCK_SLOPE.mov - Accessed Feb 7 2022
- EX1608_VID_20161203T184000Z_CPHD.mov - Accessed Feb 3 2022
- EX1708_VID_20170907T230500Z_ROVHD.mov - Accessed Feb 7 2022
- EX1803_VID_20180412T134000Z_CPHD.mov - Accessed Feb 3 2022
- EX1803_VID_20180412T134500Z_CPHD.mov - Accessed Feb 3 2022

- EX1803_VID_20180412T135000Z_CPHD.mov - Accessed Feb 3 2022
- EX1803_VID_20180428T210000Z_ROVHD.mov - Accessed Feb 3 2022
- EX1903L2_VID_20190622T171459Z_ROVHD.mov - Accessed Feb 7 2022
- EX1811_VID_20181110T154826Z_PTMAN_GEO.mov - Accessed Feb 24 2022
- EX1811_VID_20181110T155710Z_PTMAN_GEO_HL.mov - Accessed Feb 24 2022
- EX1811_VID_20181110T160320Z_PTMAN_GEO_WIDE.mov - Accessed Feb 24 2022
- EX1811_VID_20181110T162851Z_PTMAN_ROCK_CRUST.mov - Accessed Feb 24 2022
- EX1811_VID_20181110T163601Z_PTMAN_AUD_GEO.mov - Accessed Feb 24 2022
- EX1811_VID_20181112T165953Z_PTMAN_GEO.mov - Accessed Feb 24 2022
- EX1811_VID_20181112T192613Z_PTMAN_CHANNEL_GEO.mov - Accessed Feb 24 2022
- EX1904_VID_20190728T151332Z_SBMAN_COR_SCAN.mov - Accessed Feb 24 2022
- EX1904_VID_20190731T170004Z_PTMAN_BIV.mov - Accessed Feb 24 2022
- EX1904_VID_20190731T175252Z_PTMAN_SKATE.mov - Accessed Feb 24 2022
- EX1905L2_VID_20190829T181328Z_PTMAN_ROC_COR_SPO.mov - Accessed Feb 24 2022
- EX1905L2_VID_20190903T174545Z_SBMAN_SPO.mov - Accessed Feb 24 2022
- EX1905L2_VID_20190903T174546Z_PTMAN_SPO.mov - Accessed Feb 24 2022
- EX1905L2_VID_20190903T181537Z_PTMAN_ROC.mov - Accessed Feb 24 2022
- EX1603_VID_20160228T210505Z_PTMAN_ROCK_SIP.mov - Accessed Feb 28 2022
- EX1606_VID_20160803T022516Z_PTMAN_COR_ROCK_WIDE.mov - Accessed Feb 28 2022
- EX1606_VID_20160811T053252Z_PTMAN_ROCK_SHI.mov - Accessed Feb 28 2022
- EX1606_VID_20160812T033213Z_PTMAN_COR_ROCK_HL.mov - Accessed Feb 28 2022
- EX1705_VID_20170506T030036Z_PTMAN_COR_FSH_HL.mov - Accessed Feb 28 2022

- EX1705_VID_20170511T224350Z_PTMAN_ROC.mov - Accessed Feb 28 2022
- EX1708_VID_20170921T231001Z_PTMAN_GEO.mov - Accessed Feb 28 2022
- EX1806_VID_20180701T171716Z_PTMAN_WALL.mov - Accessed Feb 28 2022
- EX1811_VID_20181105T163206Z_PTMAN_GEO.mov - Accessed Feb 28 2022
- EX1811_VID_20181110T155710Z_PTMAN_GEO_HL.mov - Accessed Feb 28 2022
- EX1811_VID_20181110T160320Z_PTMAN_GEO_WIDE.mov - Accessed Feb 28 2022
- EX1811_VID_20181110T163601Z_PTMAN_AUD_GEO.mov - Accessed Feb 28 2022
- EX1811_VID_20181115T145314Z_PTMAN_WIDE_HL.mov - Accessed Feb 28 2022
- EX1811_VID_20181115T150157Z_PTMAN_WIDE_HL.mov - Accessed Feb 28 2022
- EX1811_VID_20181115T161656Z_PTMAN_ROC_COR.mov - Accessed Feb 28 2022
- EX1811_VID_20181115T170015Z_PTMAN_LANDSCAPE.mov - Accessed Feb 28 2022
- EX1811_VID_20181115T173515Z_PTMAN_FSH.mov - Accessed Feb 28 2022
- EX1811_VID_20181115T181103Z_PTMAN_LANDSCAPE.mov - Accessed Feb 28 2022
- EX1811_VID_20181119T171402Z_SBMAN_ROCKS_WIDE.mov - Accessed Feb 28 2022
- EX1811_VID_20181119T172703Z_SBMAN_WIDE_ROCKS.mov - Accessed Feb 28 2022
- EX1304L1_VID_20130724T165918Z_ROVHD_SEAFL_RUBBLE_TRASH.mov - Accessed Mar 30 2022
- EX1304L2_VID_20130807T142137Z_ROVHD_COR_ROCK_WALL.mov - Accessed Mar 30 2022
- EX1304L2_VID_20130815T141415Z_ROVHD_800M_TOWTRANSCT_11.mov - Accessed Mar 30 2022
- EX1708_VID_20170908T212000Z_ROVHD.mov - Accessed Mar 31 2022
- EX1803_VID_20180413T163500Z_ROVHD.mov - Accessed Mar 31 2022
- EX1004L3_VID_20100722T212402Z_ROVHD_ROCK_LEDGE_SLOPE.mov - Accessed Feb 7 2022

- EX1202L2_VID_20120320T183826Z_ROVHD_SPO_BOTTOM_WIDE_00.mov - Accessed Mar 11 2022
- EX1202L2_VID_20120322T134828Z_ROVHD_CRA_CAR_COR_FSH.mov - Accessed Mar 11 2022
- EX1202L2_VID_20120322T135148Z_ROVHD_COR_SPO_FSH.mov - Accessed Mar 11 2022
- EX1202L2_VID_20120322T145121Z_ROVHD_CAR.mov - Accessed Mar 11 2022
- EX1304L1_VID_20130718T185323Z_ROVHD_SKATE.mov - Accessed Mar 10 2022
- EX1304L2_VID_20130804T180454Z_ROVHD_LANDSCAPE.mov - Accessed Mar 10 2022
- EX1304L2_VID_20130808T185813Z_ROVHD_GRATE_GREAT.mov - Accessed Mar 10 2022
- EX1711_VID_20171204T162500Z_CPHD.mov - Accessed Mar 10 2022
- EX1902_VID_20190514T194459Z_ROVHD.mov - Accessed Mar 11 2022

Appendix D

Marine odometry literature collection

It was requested of us to preserve this early literature collection which was made at the start of our master's thesis. It includes papers relevant to our research, their abstracts, and commentary from our side. Before this, we include our preliminary summary of our findings.

D.1 Preliminary summary

So far, our literature search has been focused on collecting as many (possibly) relevant papers as possible. We have mainly considered titles and abstracts for selection. Topics include characteristics of marine snow, simulation of snow and marine snow (movement and geometry), and methods of combating marine snow, snow, and rain in computer vision.

D.1.1 Elimination of marine snow

- More mature field than we expected.
- Most methods are removal methods, i.e. a degraded image is processed to remove marine snow. (Some of these methods also do other enhancement to remove underwater effects, like discoloration and contrast improvement)
- Existing methods seem to have solid results on still images (no video examples) with low density snow.
- Filter-based methods are common, e.g, a median-filter.

- You may have misunderstood the segmentation method we presented last time (pixel-wise classification). We want to create a system, e.g,. a CNN, with the entire image as input, and an output which shows the likelihood of each pixel being snow (binary semantic segmentation).
- Some methods do the binary segmentation described above, one with a multi-scale CNN, and one CNN with residual connections. We believe combining these concepts into one may yield good results.

D.1.2 Elimination of above water snow or rain

- Unsurprisingly, more research exists in these above-water scenarios.
- Most methods are removal methods, e.g., a degraded image is processed to remove rain.
- Rain removal is the considered easiest since snow has more variance in size, shape, and opacity than rain. Furthermore, this variance can be present in just one image, which is atypical for rain.
- Consequently, we believe there is more potential in adapting snow-removal methods than rain-removal methods.
- The datasets we have found so far are all synthetic ranging from hundreds to tens of thousands of images.
- We could consider transfer-learning on a synthetic snow dataset, but if we can synthesize our own marine-snow data, data-scarcity should not be an issue so there may be no benefit.
- Some methods have a primary-step similar to our segmentation-idea, where each pixel is labelled as snow or not-snow. This information is then used to remove the snow, e.g. by a CNN or filter-based process.
- We are unsure if removal is strictly needed for the SLAM keypoint-rejection case, but all methods so far seem to do removal. If we do marine snow removal, it would be before keypoint-detection, but a our worry is that the patches of removed snow are not fit for matching and tracking.

D.1.3 Marine snow

- Marine snow is a very broad term describing micro-scale particles up to decimeter-sizes.
- Most common around 50m depth, otherwise evenly distributed?
- Typically sink 50-100m per day.
- Marine snow can vary significantly in how it presents itself in different video-sequences, due to camera properties and the snow itself.

- We will need multiple sequences if we want to model most marine snow conditions. If there is some kind of snow which is especially common in the north sea, we could limit ourselves to these.

D.1.4 Simulation

- To model current (ocean or wind), Lattice Boltzman Models are common.
- We have only found one marine snow simulation;half of its snow did not look like the snow we were used to. Instead, it looked like an out of focus effect like bokeh on a light-strip. The other half was reminiscent to our method from last semester (ellipse-method)
- Should we make multiple sequences with different types of marine snow?
- How realistic does our marine-snow models need to be, to look good in VAROS (at a distance)?

D.1.5 General questions

- Are the existing marine-snow removal methods available for comparison?
- How do we include and cite our pre-project report, e.g., can we copy paragraphs from the background and literature chapters?

D.2 Marine Snow

Alldredge and Silver (1988): **"Characteristics, dynamics and significance of marine snow"**

Macroscopic aggregates of detritus, living organisms and inorganic matter known as marine snow, have significance in the ocean both as unique, partially isolated microenvironments and as transport agents: much of surface-derived matter in the ocean fluxes to the ocean interior and the sea floor as marine snow. As microhabitats, marine snow aggregates contain enriched microbial communities and chemical gradients within which processes of photosynthesis, decomposition, and nutrient regeneration occur at highly elevated levels. Microbial communities associated with marine snow undergo complex successional changes on time scales of hours to days which significantly alter the chemical and biological properties of the particles. Marine snow can be produced either de novo by living plants and animals especially as mucus feeding webs of zooplankton, or by the biologically-enhanced physical aggregation of smaller particles. By the latter pathway, microaggregates, phytoplankton, fecal pellets, organic debris and clay-mineral particles collide by differential settlement or physical shear and adhere by the action of various, biologically-generated, organic compounds. Diatom flocculation is a poorly understood source of marine snow of potential global significance. Rates of snow production and breakdown are not known but are critical to predicting

flux and to understanding biological community structure and transformations of matter and energy in the water column. The greatest challenge to the study of marine snow at present is the development of appropriate technology to measure abundances and characteristics of aggregates in situ.

Boffety and Galland (2012): **"Phenomenological marine snow model for optical underwater image simulation: Applications to color restoration"**

Optical imaging plays an important role in oceanic science and engineering. However, the design of optical systems and image processing techniques for subsea environment are challenging tasks due to water turbidity. Marine snow is notably a major source of image degradation as it creates white bright spots that may strongly impact the performance of image processing methods. In this context, it is necessary to have a tool to foresee the behavior of these methods in marine conditions. This paper presents a phenomenological model of marine snow for image simulation. In order to highlight the interest of such a modeling for image processing characterization, the impact of marine snow perturbation on a color restoration technique is analyzed and a solution to improve the robustness of the algorithm is finally proposed.

D.3 Simulating marine environments

Simulating marine environments eliminates the issues related to inaccuracy in ground truths, but at the cost of realism. However, advances in computer graphics and increasing interest from researches have steadily reduced the significance of this trade-off.

D.3.1 Synthesis of Marine Snow

Excerpt: The marine snow is simulated by five layers of randomly moving particles of varying size and speed, each layer consisting of 100.000 particles of which approximately 1200 are visible in each image. This fairly closely resembles real marine snow effects under calm sea conditions.

Hildebrandt and Kirchner (2010): **"IMU-aided stereo visual odometry for ground-tracking AUV applications"**

This paper addresses the problem of AUV navigation by showing the feasibility of a stereo visual-inertial approach to odometry retrieval for an AUV. This information is intended as input for a complete SLAM system. After its classification among many other similar approaches in recent work is shown, the algorithm is described in detail. A number of experiments conducted on synthetic data show the performance in respect to precision and computational cost. As a conclusion, future extensions and applications are briefly discussed.

Irisson et al. (2022): **"Machine Learning for the study of plankton and marine snow from images"**

Quantitative imaging instruments produce a large number of images of plankton and marine snow, acquired in a controlled manner, from which the visual characteristics of individual objects and their in situ concentrations can be computed. To exploit this wealth of information, machine learning is necessary to automate tasks such as taxonomic classification. Through a review of the literature, we highlight the progress of those machine classifiers and what they can and still cannot be trusted for. Several examples showcase how the combination of quantitative imaging with machine learning has brought insights on pelagic ecology. They also highlight what is still missing and how images could be exploited further through trait-based approaches. In the future, we suggest deeper interactions with the computer sciences community, the adoption of data standards, and the more systematic sharing of databases to build a global community of pelagic image providers and users.

The 'bokeh-effect' dataset mentioned earlier.

Sato, Ueda, and Tanaka (2021): ***Marine Snow Removal Benchmarking Dataset***

This paper introduces a new benchmarking dataset for marine snow removal of underwater images. Marine snow is one of the main degradation sources of underwater images that are caused by small particles, e.g., organic matter and sand, between the underwater scene and photosensors. We mathematically model two typical types of marine snow from the observations of real underwater images. The modeled artifacts are synthesized with underwater images to construct large-scale pairs of ground-truth and degraded images to calculate objective qualities for marine snow removal and to train a deep neural network. We propose two marine snow removal tasks using the dataset and show the first benchmarking results of marine snow removal. The Marine Snow Removal Benchmarking Dataset is publicly available online.

D.4 Marine Snow Detection and Removal

Three methods for snow removal. Two based on GANs and one with U-net and partial convolutions. They make their own synthetic dataset similar to ours, but seemingly without depth information. They do not describe how their data was synthesised. They present results on very difficult real-wporld data, and it seems snow removal is predictably too difficult in these scenarios. Noise, ghosting and color changes are present in these examples.

L. Li et al. (2021): **"Removal of Floating Particles from Underwater Images Using Image Transformation Networks"**

In this paper, we propose three methods for removing floating particles from underwater images. The first two methods are based on Generative Adversarial Networks (GANs). The first method uses CycleGAN which can be trained with an unpaired dataset, and the second method uses pix2pixHD that is trained with a paired dataset created by adding artificial particles to underwater images. The third method consists of two-step process – particle detection and image inpainting. For particle detection, an image segmentation neural network U-Net is trained by using underwater images added with artificial particles. Using the output of U-Net, the particle regions are repaired by an image inpainting network Partial Convolutions. The experimental results showed that the methods using GANs were able to remove floating particles, but the resolution became lower than that of the original images. On the other hand, the results of the method using U-Net and Partial Convolutions showed that it is capable of accurate detection and removal of floating particles without loss of resolution.

Very basic method in regards to marine snow; they simply model snow as additive noise. The main contributions are: dehazing considering underwater particle physics, artificial light model for underwater robot applications, independence to channel numbers and online processing performance. Their method for modeling image degradation is based on "Underwater image enhancement by wavelength compensation and dehazing" by Chiang et al. (TODO: Read) and they adopt light attenuation as an exponentially decaying term, however they add a non uniform artificial light estimation, blur modeling with point spread function, and random noise from particles (marine snow).

Cho and A. Kim (2017): **"Visibility enhancement for underwater visual SLAM based on underwater light scattering model"**

This paper presents a real-time visibility enhancement algorithm for effective underwater visual simultaneous localization and mapping (SLAM). Unlike an aerial environment, an underwater environment contains larger particles and is dominated by a different image degradation model. Our method starts with a thorough understanding of underwater particle physics (e.g., forward, back, multiple scattering, blur and noise). Targeting underwater image enhancement in a real-world application, we include an artificial light model in the derivation. The proposed method is effective for both color and gray images with substantial improvement in the process time compared to conventional methods. The proposed method is validated by using simulated synthetic images (color) and real-world underwater images (color and grayscale). Using two underwater image sets acquired from the same area but with different water turbidity, we evaluate the proposed visibility enhancement and camera registration improvement in SLAM.

"Compared to underwater image enhancement methods, the development of underwater marine snow removal methods proceed rather slowly." Seperates the image into high and low frequency components. Snow removal is done on the high frequency component, while other enhancement techniques are performed on the low frequency component. The marine snow removal network is a residual network. The paper achieves seemingly good results, but seems to struggle a little with anything more than light marine snow

Y. Wang et al. (2021): **"Underwater image enhancement and marine snow removal for fishery based on integrated dual-channel neural network"**

Computer vision technology can reduce the intensity and difficulty of marine aquaculture. Nonetheless, the degradation problems of marine fishery underwater images hamper further interpretation and analysis of underwater information by computer vision technology. In order to comprehensively solve the problems of degradation in marine fishery underwater images, this paper proposes an end-to-end integrated dual-channel network model, which uses the underwater image enhancement module based on the residual dense network to perform color correction and dehazing for the low-frequency layer of images, and the marine snow removal module based on local residual learning strategy to remove the white marine snow in the high-frequency layer of images. Moreover, the addition of refinement module further improves the textual details and colors of images. Experimental results on marine fishery dataset indicate that the approach proposed in this paper performs better than several state-of-the-art methods in quantitative metrics, including underwater image quality measure (UIQM), blur index and smooth index, effective in improving the contrast of underwater fishery images and reducing marine snow noise, while achieving better visual quality in qualitative evaluation such as color correction, dehazing, detail and feature restoration. Furthermore, the generalization tests and application tests have proved its effectiveness in underwater scenarios, which means it can meet the practical needs in the field of marine aquaculture.

Works by finding areas with snow, and then blurring only those areas. To find the snow, the authors calculate the probability (of a pixel being snow?) solely based on the luminance. The luma channel is found by converting the image from standard RGB to YCbCr, where Y is luma. Their approach for calculating the probability is done by sliding a window over the image and then calulating the probability of the center pixel of the window by dividing the number of high luminance pixels in the window over the total number of pixels in the image. a 7x7 sliding window is used.

Banerjee et al. (2014): **"Elimination of Marine Snow effect from underwater image - An adaptive probabilistic approach"**

Along with color loss, another severe problem of underwater optical imaging is Marine Snow effect which occurs because of back scattering from suspended organic detritus, solid particles or bubbles. Their appearance like tiny sparkling dots often reduces the scene perception and sometimes leads to spurious features on segmentation. This paper is concerned with removal of the marine snow effect from underwater images by a probabilistic approach considering the local statistics of luminance properties after a RGB to YCbCr transform.

This paper's previous work left a good impression

Boguslaw Cyganek and Gongola (2018): **"Real-time marine snow noise removal from underwater video sequences"**

Underwater images suffer from various degradation factors, such as blur, haze, color degradation, and marine snow. Marine snow is a type of noise, caused mostly by biological particles that fall into the ocean bottom, and which impedes proper object detection in underwater vision. A method for real-time marine snow removal from underwater color and monochrome video is presented. It is based on the proposed marine snow model, spatiotemporal patch analysis, and three-dimensional median filtering. The method was evaluated on a number of real underwater sequences endowed with the hand-annotated ground-truth data which were made available from the Internet. As shown by the experiments, the method attains high accuracy and performs in real time.

Farhadifard, Radolko, and U. v. Lukas (2017): **"Single Image Marine Snow Removal based on a Supervised Median Filtering Scheme"**

Underwater image processing has attracted a lot of attention due to the special difficulties at capturing clean and high quality images in this medium. Blur, haze, low contrast and color cast are the main degradations. In an underwater image noise is mostly considered as an additive noise (e.g. sensor noise), although the visibility of underwater scenes is distorted by another source, termed marine snow. This signal disturbs image processing methods such as enhancement and segmentation. Therefore removing marine snow can improve image visibility while helping advanced image processing approaches such as background subtraction to yield better results. In this article, we propose a simple but effective filter to eliminate these particles from single underwater images. It consists of different steps which adapt the filter to fit the characteristics of marine snow the best. Our experimental results show the success of our algorithm at outperforming the existing approaches by effectively removing this phenomenon and preserving the edges as much as possib

Supposedly a snow generating algorithm. To our eyes it looks more like marine snow, could be very interesting.

Zou, Xie, and Zhao (2010): **"Algorithm for generating snow based on GPU"**

The simulation of natural phenomena plays an important role in the area of computer game, film production, simulation and so on. Snow is an important part of the simulation of nature phenomena, so how to generate high-quality and controllable snow texture is significant. Now the algorithms of generating snow are based on triangulated surfaces, they put some triangles on multi-layer concentric spheres to form snow. The disadvantage of these algorithms is that the edges of snow are not smooth enough, the angles of snow are too acute. The approach is splitting up a triangle to four, and then rendering the triangles associated with vertexes using the rendering algorithm of Quadric Bezier Curve based on GPU, and rendering the triangle formed by points on edges normally. Finally it can generate high-quality snow texture with the technology of alpha blending.

Two step marine snow removal. First step similar to our semantic segmentation idea, but not multi-scale.

Koziarski and Bogusław Cyganek (2019): **"Marine Snow Removal Using a Fully Convolutional 3D Neural Network Combined with an Adaptive Median Filter"**

Marine snow is a type of noise that affects underwater images. It is caused by various biological and mineral particles which stick together and cause backscattering of the incident light. In this paper a method of marine snow removal is proposed. For particle detection a fully convolutional 3D neural network is trained with a manually annotated images. Then, marine snow is removed with an adaptive median filter, guided by the output of the neural network. Experimental results show that the proposed solution is capable of an accurate removal of marine snow without negatively affecting the image quality.

Very early paper on motion fields for snow. Cited by many of the other papers. Intended for above-water, but we assume it is trivial to adapt it to underwater simulation.

Wei et al. (2003): **"Blowing in the wind"**

We present an approach for simulating the natural dynamics that emerge from the coupling of a flow field to light-weight, mildly deformable objects immersed within it. We model the flow field using a Lattice Boltzmann Model (LBM) extended with a subgrid model and accelerate the computation on commodity graphics hardware to achieve real-time simulations. We demonstrate our approach using soap bubbles and a feather blown by wind fields, yet our approach is general enough to apply to other light-weight objects. The soap bubbles illustrate Fresnel reflection, reveal the dynamics of the unseen flow field in which they travel, and

display spherical harmonics in their undulations. The free feather floats and flutters in response to lift and drag forces. Our single bubble simulation allows the user to directly interact with the wind field and thereby influence the dynamics in real time.

D.5 Segmentation

H.-K. Kim et al. (2019): **"Traffic Light Recognition Based on Binary Semantic Segmentation Network"**

A traffic light recognition system is a very important building block in an advanced driving assistance system and an autonomous vehicle system. In this paper, we propose a two-staged deep-learning-based traffic light recognition method that consists of a pixel-wise semantic segmentation technique and a novel fully convolutional network. For candidate detection, we employ a binary-semantic segmentation network that is suitable for detecting small objects such as traffic lights. Connected components labeling with an eight-connected neighborhood is applied to obtain bounding boxes of candidate regions, instead of the computationally demanding region proposal and regression processes of conventional methods. A fully convolutional network including a convolution layer with three filters of (1 x 1) at the beginning is designed and implemented for traffic light classification, as traffic lights have only a set number of colors. The simulation results show that the proposed traffic light recognition method outperforms the conventional two-staged object detection method in terms of recognition performance, and remarkably reduces the computational complexity and hardware requirements. This framework can be a useful network design guideline for the detection and recognition of small objects, including traffic lights.

Kampffmeyer, Salberg, and Jenssen (2016): **"Semantic Segmentation of Small Objects and Modeling of Uncertainty in Urban Remote Sensing Images Using Deep Convolutional Neural Networks"**

We propose a deep Convolutional Neural Network (CNN) for land cover mapping in remote sensing images, with a focus on urban areas. In remote sensing, class imbalance represents often a problem for tasks like land cover mapping, as small objects get less prioritised in an effort to achieve the best overall accuracy. We propose a novel approach to achieve high overall accuracy, while still achieving good accuracy for small objects. Quantifying the uncertainty on a pixel scale is another challenge in remote sensing, especially when using CNNs. In this paper we use recent advances in measuring uncertainty for CNNs and evaluate their quality both qualitatively and quantitatively in a remote sensing context. We demonstrate our ideas on different deep architectures including patch-based and so-called pixel-to-pixel approaches, as well as their combination, by classifying each pixel in a set of aerial images covering Vaihingen, Germany. The results

show that we obtain an overall classification accuracy of 87%. The corresponding F1-score for the small object class ”car” is 80.6%, which is higher than state-of-the art for this dataset.

Takeki et al. (2016): **“Detection of small birds in large images by combining a deep detector with semantic segmentation”**

This paper tackles the problem of bird detection in large landscape images for applications in the wind energy industry. While significant progress in image recognition has been made by deep convolutional neural networks (CNNs), small object detection remains a problem. To solve it, we follow the idea that a detector can be tuned to small objects of interest and semantic segmentation methods can be complementary used to recognize large background areas. Specifically, we train a CNN-based detector, fully convolutional networks, and a superpixel-based semantic segmentation method. The results of the three methods are combined by using support vector machines to achieve high detection performance. Experimental results on a bird image dataset show the high precision and effectiveness of the proposed method.

The vegetation class example could pass as a marine snow segmentation.

Segl and Kaufmann (2001): **“Detection of small objects from high-resolution panchromatic satellite imagery based on supervised image segmentation”**

A new concept for the detection of small objects from modular optoelectronic multispectral scanner (MOMS-02) high spatial resolution panchromatic satellite imagery is presented. The authors combine supervised shape classification with unsupervised image segmentation in an iterative procedure which allows a target-oriented search for specific object shapes.

D.6 Other

Like many other papers, the paper’s removal process and ground truths are modelled as rainy_image = rainless_background + rain.

H. Wang et al. (2020): **“A Model-Driven Deep Neural Network for Single Image Rain Removal”**

Deep learning (DL) methods have achieved state-of-the-art performance in the task of single image rain removal. Most of current DL architectures, however, are still lack of sufficient interpretability and not fully integrated with physical structures inside general rain streaks. To this issue, in this paper, we propose a model-driven deep neural network for the task, with fully interpretable network structures. Specifically, based on the convolutional dictionary learning mechanism for

representing rain, we propose a novel single image deraining model and utilize the proximal gradient descent technique to design an iterative algorithm only containing simple operators for solving the model. Such a simple implementation scheme facilitates us to unfold it into a new deep network architecture, called rain convolutional dictionary network (RCDNet), with almost every network module one-to-one corresponding to each operation involved in the algorithm. By end-to-end training the proposed RCDNet, all the rain kernels and proximal operators can be automatically extracted, faithfully characterizing the features of both rain and clean background layers, and thus naturally lead to its better deraining performance, especially in real scenarios. Comprehensive experiments substantiate the superiority of the proposed network, especially its well generality to diverse testing scenarios and good interpretability for all its modules, as compared with state-of-the-arts both visually and quantitatively.

Does above water snow removal and introduces the Snow100k synthetic snow dataset. Begins to describe why snow removal is more challenging than rain removal (more varied size, shape and opacity. Less predictable trajectories and uneven density in the image). Tow modules are used to remove snow, the translucency recovery module which recovers areas obscured by translucent snow, and the residual generation module which recovers areas completely covered by opaque snow. Inception-v4 is used to get multi-scale features for the modules. Instead of summing up the multi-scale features, a concatenation function is learned which intends to preserve spatial information to a higher degree than summing.

Y.-F. Liu et al. (2018): **"DesnowNet: Context-Aware Deep Network for Snow Removal"**

Existing learning-based atmospheric particle-removal approaches such as those used for rainy and hazy images are designed with strong assumptions regarding spatial frequency, trajectory, and translucency. However, the removal of snow particles is more complicated because it possess the additional attributes of particle size and shape, and these attributes may vary within a single image. Currently, hand-crafted features are still the mainstream for snow removal, making significant generalization difficult to achieve. In response, we have designed a multistage network codenamed DesnowNet to in turn deal with the removal of translucent and opaque snow particles. We also differentiate snow into attributes of translucency and chromatic aberration for accurate estimation. Moreover, our approach individually estimates residual complements of the snow-free images to recover details obscured by opaque snow. Additionally, a multi-scale design is utilized throughout the entire network to model the diversity of snow. As demonstrated in experimental results, our approach outperforms state-of-the-art learning-based atmospheric phenomena removal methods and one semantic segmentation baseline on the proposed Snow100K dataset in both qualitative and quantitative comparisons. The results indicate our network would benefit appli-

cations involving computer vision and graphics.

The first part of the stacked network is very similar to our original 2-class semantic segmentation idea. Using densely connected CNN's, this paper used a stacked approach which improves performance. The bottom part of the stack performs a binary semantic segmentation task which identifies snow. This data is then concatenated to the image features and sent to the next part of the stack which removes the snow.

P. Li et al. (2019): **"Stacked dense networks for single-image snow removal"**

Single image snow removal is important since snowy images usually degrade the performance of computer vision systems. In this paper, we deduce a physics-based snow model and propose a novel snow removal method based on the snow model and deep neural networks. Our model decomposes a snowy image into a nonlinear combination of a snow-free image and dynamic snowflakes. Inspired by our model and DenseNet connectivity pattern, we design a novel Multi-scale Stacked Densely Connected Convolutional Network (MS-SDN) to simultaneously detect and remove snowflakes in an image. The MS-SDN is composed of a multi-scale convolutional sub-net for extracting feature maps and two stacked modified DenseNets for snowflakes detection and removal. The snowflake detection sub-net guides snow removal through forward transmission, and the snowflake removal sub-net adjusts snow detection through back transmission. In this way, snowflake detection and removal mutually improve the final results. For training and testing our method, we constructed a large-scale benchmark synthesis dataset which contains 3000 triplets of snowy images, snowflakes, and snow-free images. Specifically, the snow-free images are captured from snow scenes, and the snowy images are synthesized by using our deduced snow model. Our extensive quantitative and qualitative experimental results show that our MS-SDN performs better than several state-of-the-art methods, and the stacked structure is better than multi-branch structures in terms of snow removal.

Snow synthesis, with a promising method for creating a flow field. Wind interaction is based on a discrete form of the Boltzmann equation and works in 3D. Says wind-interaction has two main possible approaches, assuming that the fluid is continuous in both time and space, we can use the Navier-Stokes equations. However, the authors consider a statistical physics based approach with the Boltzman eq. to be easier (but not easy). Since the flow field is a macroscopic effect, discrete kinetics in time and space are used. "The wind field is sampled at many grids, and the moving state of air particles at the grid nodes is described by distribution functions. Let those particles move along the grid lines, and collide with each other at the grid according to certain rules. The evolution of the distribution function reflects the motion law of wind macrocosmically."

C. Wang et al. (2006): **"Real-time snowing simulation"**

A snowing scene has a unique fascination for people due to its incomparable beauty. However, little work has been presented on the real-time generation of a dynamic snowing scene, partially due to the difficulty that the simulation of a dynamic snowing process involves the complex modeling of the wind field and the interaction between wind and snow. In this paper, by fully considering the physical characteristics of wind and snow, we construct a three-dimensional wind field based on the discrete form of the Boltzmann equation. According to the interaction laws between wind and snow, we simulate the falling of snow, deposition and erosion in 3D space. Experimental results show that realistic wind-driven snow scenes under different speeds of wind with different amounts of snowfall can be rendered in real-time.

Important paper on snow-synthesis going by the number of citations, however the snowy graphics look outdated. To improve efficiency and realism, Langer's particle system intentionally keeps the particle count low. To 'fill in' the gaps, a dynamic texture, made using a spectral synthesis method, is composited into the scene. This method generates more convincing results than simply increasing the particle count. Wind-snow interaction is not explicitly handled.

Langer et al. (2004): **"A Spectral-particle hybrid method for rendering falling snow."**

Falling snow has the visual property that it is simultaneously a set of discrete moving particles as well as a dynamic texture. To capture the dynamic texture properties of falling snow using particle systems can, however, require so many particles that it severely impacts rendering rates. Here we address this limitation by rendering the texture properties directly. We use a standard particle system to generate a relatively sparse set of falling snow ak es, and we then composite in a dynamic texture to ll in between the particles. The texture is generated using a novel image-based spectral synthesis method. The spectrum of the falling snow texture is dened by a dispersion relation in the image plane, derived from linear perspective. The dispersion relation relates image speed, image size, and particle depth. In the frequency domain, it relates the wavelength and speed of moving 2D image sinusoids. The parameters of this spectral snow can be varied both across the image and over time. This provides the e xibility to match the direction and speed parameters of the spectral snow to those of the falling particles. Camera motion can also be matched. Our method produces visually pleasing results at interactive rendering rates. We demonstrate our approach by adding snow effects to static and dynamic scenes. An extension for creating rain effects is also presented.

Does snow synthesis and claims that rendering snow or fog onto existing images has never been covered before in the literature. "Introducing snow into an existing real scene requires the reconstruction of that scene – otherwise depth effects and occlusion of both snow flakes and road objects will not unfold.[...]Since single snow flakes become indistinguishable to an optical sensor as soon as they are far enough away, only a limited space in front of the camera has to be populated with flakes. Light attenuation effects that are caused by the consolidated snow masses are taken care of in a later step. Depending on the general motion vector of the snow, the space behind and left and right of the camera may be ignored as well"

Bernuth, Volk, and Bringmann (2019): **"Simulating Photo-realistic Snow and Fog on Existing Images for Enhanced CNN Training and Evaluation"**

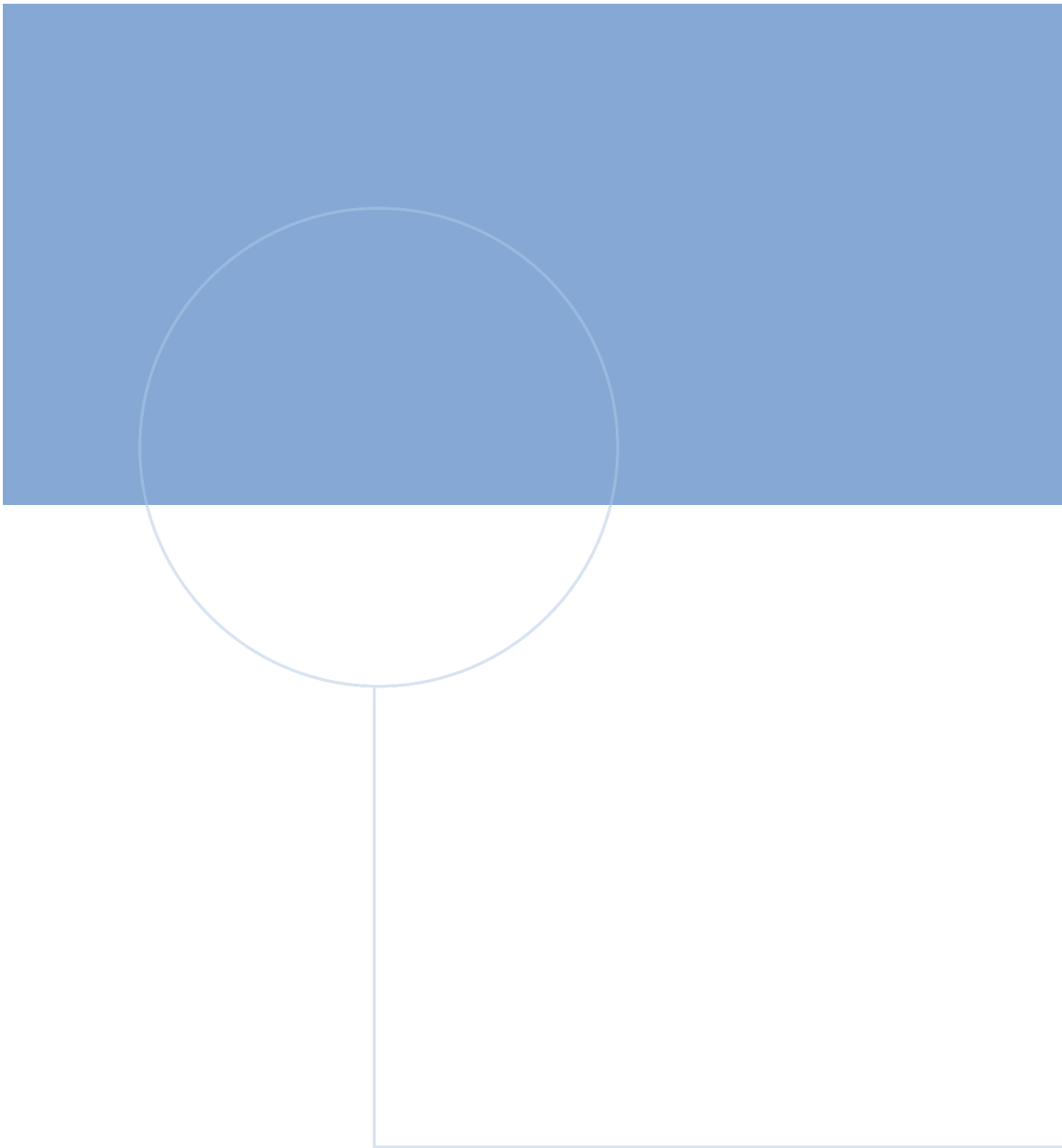
Verification and robustness testing of machine learning algorithms for autonomous driving is crucial. Due to the increasing complexity and quantity of those systems in a single vehicle, just driving the required distance with a newly developed vehicle is not feasible anymore: billions of hours on the street without failure are necessary to qualify for industry standards like ISO 26262. That is where simulation comes into play: machine learning algorithms are trained and evaluated on well known image data sets like KITTI or Cityscapes. But today's data sets mostly contain images taken under perfect weather conditions and therefore do not harden optical object detection algorithms against various weather conditions. This paper focuses on reusing these established and labeled data sets by augmenting them with adverse weather effects like snow and fog. Those effects are rendered physically correct and life like while being added to existing real world images. Thanks to easy parametrization the weather influences may be varied as necessary and allow for finely tuned learning and optimization processes. The weather effects are evaluated with regard to realism and impact on an established object detection algorithm. These newly created weather-influenced images may be used to validate or train new object detection algorithms.

Use depth to simulate rain and fog

Sen, Das, and Sahu (2021): **"Rendering Scenes for Simulating Adverse Weather Conditions"**

Most of the object detection schemes do not perform well when the input image is captured in adverse weather. Reason being that the available datasets for training/testing of those schemes didn't have many images in such weather conditions. Thus in this work, a novel approach to render foggy and rainy datasets is proposed. The rain is generated via estimation of the area of the scene image and then computing streak volume and finally overlapping the streaks with the scene image. As visibility reduces with depth due to fog, rendering of fog

must take depth-map into consideration. In the proposed scheme, the depth map is generated from a single image. Then, the fog coefficient is generated by modifying the 3D Perlin noise with respect to the depth map. Further, blending the corresponding density of the fog with the scene image at a particular region based on precomputed intensities at that region. Demo dataset is available in this https://github.com/senprithwish1994/DatasetAdverse.



NTNU

Norwegian University of
Science and Technology