Master's thesis

2021

Arild Madshaven

Master's thesis

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

Arild Madshaven

Robust Fish Cage Hole Detection in Challenging Environments

Rethinking Spatiotemporal Deep Learning and Advanced Computer Vision Techniques

May 2021

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

Abstract

In 2019 alone, close to 300,000 Atlantic Salmon reportedly fled Norwegian aquaculture sites, which is more than half the number of the remaining wild stock. A common escape route is through net holes, and frequent inspection of fish cage integrity is therefore a necessary preventative measure. A complete algorithmic framework has been initiated to fully automatise the search for net holes in a video sequence captured by a remotely controlled vehicle carrying out a net cleaning operation.

The framework presents a new strategy for net thread segmentation utilising a U-Net variant called MultiRes U-Net. A computationally efficient alteration to the U-Net's input layer is proposed to encourage its spatiotemporal coherency. The introduction of artificial intelligence for segmentation, contrary to traditional edge detection or histogram thresholding, allows seamless discrimination of intelligible net structure from noisy surroundings such as turbulent water, fish, and equipment. An adaptive scheme based on morphological operations and region growing is suggested as a frame-by-frame irregularity detector, and a spatiotemporal filter to verify irregularities that occur in the same area over a certain period of time. A deep convolutional neural network based on the VGG16 model has been specialised on separating net structure from fish and nonsense to classify verified irregularities – a final barricade to prevent objects falsely included in the segmentation from being reported as holes.

Promising results were achieved, and most holes present in a set of ten 10-second test videos shot in challenging real-world scenes were correctly identified. Particularly satisfactory were the performances of the deep learning approaches to scene segmentation and irregularity classification, whilst the local irregularity detector and the spatiotemporal filter require further work to improve the robustness and the efficiency of the framework.

Sammendrag

I 2019 alene ble nesten 300.000 atlanterhavslaks rapportert rømt fra norske oppdrettsanlegg. Dette antallet tilsvarer over halve den gjenværende villaksbestanden. En vanlig fluktrute går gjennom hull i nota, og regelmessig kontroll av nettintegriteten på et oppdrettsanlegg er derfor et nødvendig preventivt tiltak. Et komplett algoritmisk rammeverk har blitt initiert for å automatisere søket etter hull i nettmaskene i en videosekvens filmet av undervannsrobot under notvask.

Rammeverket presenterer en ny strategi for å segmentere nettmasker ved hjelp av en U-Net-variant kalt MultiRes U-Net. En liten endring i U-Nettet foreslås for å fremme koherensen til segmenteringene i både tid og rom uten å øke behovet for regnekraft stort. Å bruke kunstig intelligens, i motsetning til tradisjonelle metoder som kantdeteksjon eller histogramanalyse, mulliggjør en sømløs segmenteringsporosess hvor fisk, utstyr, grumsete vann og eventuelle fremmedlegemer ignoreres. For å oppdage uregelmessigheter i nota foreslås en adaptiv prosedyre basert på morfologiske operasjoner og segmentsvulming, som analyserer videosekvensen bilde-for-bilde. Et filter ser de oppdagede uregelmessighetene i sammenheng med tidligere oppdagelser og verifiserer de som oppstår på omtrentlig samme sted med omtrentlig samme utstrekning. Et dypt konvulerende nevralt nettverk basert på VGG16-modellen har videre blitt spesialisert på å se forskjell på nettstruktur, fisk, og vrøvl. Verifiserte uregelmessigheter blir inspisert av dette nettverket, og siden hull utelukkende kan oppstå innad i nettstruktur, avvises uregelmessigheten dersom den klassifiseres som fisk eller vrøvl.

Lovende resultater ble oppnådd, og de fleste hullene i ti krevende 10-sekunders testvideoer ble identifisert. Spesielt overbevisende var prestasjonene til de dype nevrale nettverkene brukt til segmentering og klassifisering. Bilde-for-bilde-prosedyren for uregelmessighetsdeteksjon og den påfølgende filtreringen behøver videre arbeid for å gjøre rammeverket ytterligere robust og effektivt.

Acknowledgements

Have you ever seen the façade of Gløshaugen's main building? It is pretty cool. We used to call it *Hogwarts*. It looks as though it was raised by great figures of the past -– intended to house something of uttermost importance. Glancing upon it at dusk, bathed in floodlights, it feels almost sacred.

I remember philosophising on these things when I was a first-year student in Trondheim. I felt so intimidated, as if these great figures of the past frowned upon me as I walked by, thinking to themselves *So this is what it has come down to. This place used to have standards. Through whichever crack in the system did this cockroach squeeze through? He could not even recall how to do division by hand in his MAP testing. What contribution can he possibly conjure up after a mere five years of education to justify his dwelling with us?* Well — forefathers -– this document constitutes my contribution after five years of cybernetics studies in Trondheim. I hope you will acknowledge, perhaps, that *this is not as dreadful an outcome as we anticipated.*

Here is my chance to publicly (assuming that this thesis is of interest to the public is perhaps overly optimistic, but hey) applaud the people who, literally, mean the world to me. To mum and dad, who produced me. This past year has been tumultuous. The frustration I have encountered whilst training my artificial neural networks pales in comparison to the hellish experience that is — the degeneration of our very own neural networks. It is in times like these that academic achievements reveal themselves to be of minuscule importance. I miss you very much, and it is about time I return home for a little while. I know we will come out ahead.

To my two brothers. To Inge for welcoming me to Trondheim, for helping me with school, and for commenting on my work (also this) and sharing my concerns with the word's problems (but not my solutions -– yet). To Esben for your politically incorrect group chat memes, for reminding Inge and myself that we still belong to the countryside at heart — and for always reminding us that academic people are just as broken (if not more) than people in general.

To Lars Olav for carrying me through university like a true champion. For countless hours in the lab inventing useless ping-pong machines, and for many an Arsenal game down at Three Lions. I am sure you had enough distress already — so I am somewhat sorry for adding to the burden by making of you a Gunner.

To my fiancee, Ingvild, for dragging me out on adventure. For your excitement at the sky's reflection in muddy puddles, and for sticking by me even though I spend most of my time leaping from one philosophical rabbit hole to another. You truly are my rock, and I cannot wait to build our home and life together in Oslo.

On a less personal note, but nevertheless of importance, I would like to thank my supervisors, Annette and Christian, for helping me through the last year whilst working on this subject. For your comments, suggestions, and for our informal weekly chats. They have been very much appreciated. Thanks also for your tremendous effort with our upcoming conference paper!

Big thanks be not least to Frøy gruppen for supplying the video material without which this thesis could never have been written, and for reaching out to me during a conference, asking me to write on this very exciting topic. I hope the outcome of this thesis will be of utility to you. It has surely been of great utility to me.

Dear reader. You have now made it through my acknowledgements section, are you ready for the ride?

Arild Madshaven

Arild Madshaven
Trondheim, 31 May 2021

Contents

Chapter 1

Introduction

A complete framework for robust fish cage hole detection in challenging environments — rethinking spatiotemporal deep learning and advanced computer techniques.

The title of this thesis – the *problem* at hand – is one worthy of careful consideration before engaging in discussions on implementational details. Why are fish caged in the first place? Why desire robust hole detection in such cages? What makes the environments challenging? What are computer vision techniques, and more, what is spatiotemporal deep learning? What is their contribution to a framework for hole detection in the cages of domesticated fish? The latter will be thoroughly investigated int this thesis, but providing a sufficiently grounded answer to the former questions can illuminate the context from which this work springs. Specifics of the thesis, its scope, contribution, and outline, will be justified and placed within this established context to finalise the introduction.

1.1 On Caged Fish

[1]Some 35,000 years ago man invented the knotted fish net. Although his hunt for fish stretches back hundreds of millennia, the ancient methods were probably far too primitive and his likes far too few to pose a considerable threat to the sustainability of the global marine ecosystem [2]. This has become subject to change due to technological advancements and rapid human expansion.

Since their advent in the 14th century AD, beam trawlers have arguably ravaged previously unbreachable depths, threatening bottom-dwelling aquatic organisms, constantly driving fish from their habitats. Early concerns with high levels of by-catch and the destruction of corals, sponges, and shellfish led to demonstrations and regional bans in several European countries only years after its invention [2]. These concerns are still prominent today. Up to 15% of marine catches are discarded at sea; either dead, badly hurt, or dying [3, 4].

[1]Readers familiar with the pre-project related to this thesis [1] will recognise content from chapter 1 and chapter 2.

The status of the global fish stocks were as of 2017 considered to be heavily threatened by overfishing, pollution, global warming and the likes [4, 5]. One third of all stocks were considered *overfished* whilst merely 6% were considered *underfished* - leaving most stocks fished to a *maximally sustainable degree* - explaining why marine catches have been more or less static the last three decades [4].

1.1.1 In Defence of Fish Consumption

Wild fish come in limited supply, but there are numerous reasons why they can be included in a healthy human diet. Fish is highly nutritious, and generally rich in protein, long-chained omega-3 fatty acids, and vitamins and minerals such as vitamin A, B, and D, and zinc, selenium, and calcium. Including fish in one's diet can benefit one's mental health and cognitive development, and reduce the risk of catching cardiovascular disease, stroke, and macular degeneration [4, 6].

Ensuring a continuous access to the benefits of fish for future generations means we need to increase production, but the wild stock capacities have been pushed to their limits [2, 4]. A proposed solution to this apparent conundrum is to invest in the farming of aquatic organisms — the aquaculture industry. Its contribution to the global fish production has increased more than five-fold over the past thirty years — now delivering even more fish for human consumption than fisheries [4].

1.1.2 Norwegian Aquaculture

Norway's coastline, with its deep sheltering fjords and oxygen-rich waters well-supported by the Gulf Stream, provides an excellent marine environment for the Atlantic Salmon which comprises more than 90% of the country's aquaculture activity [7]. Since the introduction of *sea cages* in the 1970s, the industry has steadily grown, now producing some 1.3 million tonnes of seafood annually, contributing in 2018 with 32 billion NOK to Norway's GDP [7]. However, in order for the industry to continue to grow, measures need to be made concerning environmental challenges.

One such challenge is the impact farmed salmon has on the remaining wild stocks. For instance, accumulation of *sea lice* within a densely populated salmon farm may pose a tremendous threat to any nearby wild salmon [8–10]. Especially so when farms have been established in and near fjords, through which vulnerable juvenile salmonoids have to swim on their journey from the rivers to the sea [9, 10].

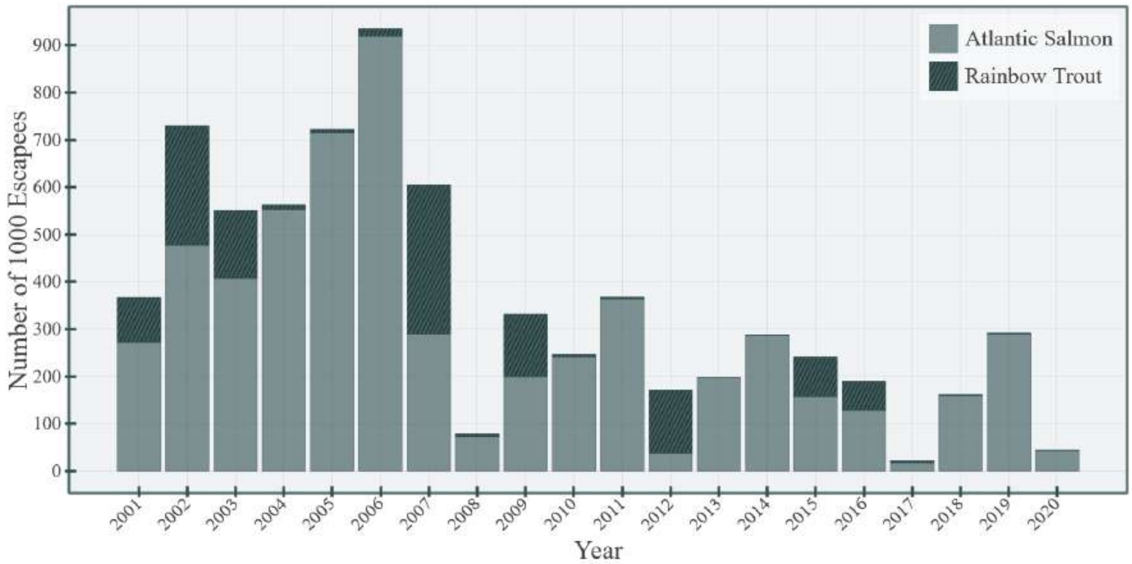


Figure 1.1: Reported number of escaped Atlantic Salmon and Rainbow Trout from Norwegian aquaculture sites the past two decades. Studies from the period 2005-2011 suggest the actual number might be up to four times the reported number [8, 10]. For comparison, the wild salmon stock is thought to be just under half a million individuals [8]. Figure based on numbers from the Norwegian Directorate of Fisheries [14, 15]. Reports can be up to one year late, so 2020 numbers are tentative.

.

Another challenge is fish escapes. In addition to the economic cost of losing fish, ecological costs can be severe in terms of interbreeding between wild and farmed stocks [7, 8, 10]. The genetic pool of farmed fish has for decades been directed in favour of traits desired from a farming perspective, and may alter the genetics of wild fish to their disadvantage [8, 10]. It is therefore of immense importance to prevent scenarios in which fish may escape from occurring.

1.2 On Fish Cage Hole Detection

Several standards and procedures were initiated in the early 2000s to lower the number of fish escapees (see fig. 1.1). For floating aquaculture sites, most notably the Norwegian Standard NS 9415 [11] introduced in 2003 and revised in 2009 along with the NYTEK regulations [12], which set requirements for technical design, dimensioning, and operation. Similar procedures for landbased aquaculture were simultaneously implemented, such as NS9416 [13] from 2013, and the call for double-secured drains from 2006 [10]. Among the demands from NS9415 is frequent control of fish net structure integrity.

Recent studies by SINTEF suggest 64% of all escaped fish between 2014 and 2018 left through net holes. The increasing number of equipment in the cages intentionally or unintentionally in contact with the net, such as cameras, bottom rings, weights, and lice skirts, might be a leading cause for defects. Furthermore, human handling of equipment and other operations account for a large part of the reported incidents [16].

Net inspections are for these reasons often carried out before and after operations that may stress the structure, as well as periodically, for instance monthly. Standard methods involve a team of divers or manual inspection of video captured by Remotely Operated Vehicles (ROVs) equipped with cameras [17]. The former approach is usually related to higher costs and longer delays than the latter, in addition to greater HSE concerns (for instance [18, 19]). Underwater drones may in principle serve to completely automate the process of continuous net integrity inspection if a robust algorithm can process its video stream and evaluate the pictured net structure.

Automatic processing of underwater net structure is not an idea coming to light as of recent. Significant effort has been made to design such systems (for example, [17, 19–22]) but proposed solutions have yet to convince the industry. The challenges are manifold; video quality might be poor, causing the net structure to appear broken. Current and waves might cause spatial deformations in the structure, creating awkward situations for naïve algorithms. Fish regularly swim past the camera and could be confused with holes. Not least – heavy algae growth often covers the net structure, totally, and have in many cases hole-like appearances. These are all reasons why *proof-of-concept* hole detection algorithms in *staged* environments and *robust* hole detection algorithms intended for *real* environments face difficulties of significantly different magnitude.

1.3 On Perception

Humans are incredible pattern recognisers. Perhaps did we not appreciate this fact fully until we strived to teach our machines to see what we see. Do you fully appreciate your ability to separate objects from background? Or your ability to tell defect net structure from healthy net structure occluded by fish and algae? Objects do not always have clear boundaries, in spite of which we still recognise our relatives in a crowded street, and we wisely assume they still have feet even if we cannot see them in the crowd. And *what* exactly are the rules for what constitutes an object? Your relative is indeed *one* object, but they certainly consist of smaller components: ten fingers, two palms, four limbs, one head. Even though these components apply to *most* humans, those who lack a few would never be confused for anything but people.

Furthermore, separating objects from one another is just part of the story. We unconsciously categorise objects based on context. For instance, it is simply not true that apples and bananas are separate entities:

$$1\,\mathsf{apple} + 1\,\mathsf{banana} = 2\,\mathsf{fruits} \tag{1.1}$$

If you crave fruit then eq. (1.1) might suffice, but if you shop ingredients for your significant other's apple pie you better know the difference. Likewise, we treat both raging bulls and sweet coconuts as *life-threatening dangerous things* if they happen to approach us at high velocities. On the flip side — both are *life-giving edible things* if they appear nicely garnished on a dinner plate. If a hitherto unseen entity appears on the plate alongside your steak, you immediately place it in the category of *life-giving edible things* if it roughly matches your prior knowledge of such entities. Perhaps it is a new kind of cutlery — in which case you would never confuse it for food. Before ever feeling it in your hands, you have already estimated its weight, texture, and size, and perfectly formed your grip to pick it up and start eating.

1.3.1 A Brief History of Computer Vision

The MIT scientist Jerry Lettvin famously discovered in 1959 that the eye of the frog reports to its brain not simply arrays of pixel intensities, but rather sophisticated responses of *bug-detecting feature detectors* [23]. The eyes were argued to be responsible for perception, rather than mere sensation, and extracted features such as *something small and jerky has entered my visual field*.

Likewise, the bulk of computer vision (CV) applications in the 20th century were heavily dependent on handcrafted feature extraction similar to that of the frog's eye, based on image *morphology* describing geometrical and textural properties of the image content [24, 25]. By utilising simple features such as area, perimeter, Freeman chain codes [26] and Levenshtein distance [27], Chamfer distance [28], Fourier descriptors [29], polygon approximation, projection, rectangularity, moments, and axes of inertia, one might quite successfully describe simple and semi-complex objects to the degree that they form multi-dimensional clusters with low intra-class variance and high inter-class variance. However, discriminating salmonoids from codfish, or Labrador Retrievers from Golden Retrievers, may require feature extraction more sophisticated than can easily be deduced by conscious brainpower alone.

The Neural Network: A Game Changer

Originating in the 1960s, but facilitated by the two-centuries-old work on the linear regressor, the neural networks (NNs) made their entrance to the public eye in the 2000s after winning several contests and achieving for the first time super-human performance in certain domains [30, 31]. A subset of the NNs, the *convolutional neural network* (CNN), especially so in the domain of CV. The traditional CNN applies to two-dimensional arrays (being for instance the pixel intensities of digital images) shifting convolutional units typically initialised with random weights. These units are called *filters*, and a CNN usually consists of several *convolutional layers* in which multiple filters are applied.

During a *training* process, the weights of the CNN and the filters are tuned towards best-fit convergence. The paradigm-shifting beauty of this process is that the *CNN itself learns to extract features in the convolutional layers*. In other words, the CNN proved to be a really good interpreter of *spatial* information, capable of identifying, itself, patterns in two-dimensional arrays. Moreover – CNNs can be extended to shift filters in a *third* dimension, exceptionally useful in applications such as magnetic resonance imaging (MRI) where two-dimensional images can be *stacked* to construct three-dimensional images. In video processing, this is analogous to stacking subsequent frames (see for instance [32]), enabling both spatial and temporal – *spatiotemporal* – feature extraction.

NNs with a significantly large number of layers are commonly referred to as *deep neural networks* (DNNs) and the training of such is called *deep learning* (DL). The granularity of the features extracted by filters in a certain layer depends on the depth of the CNN, where deeper layers extract finer-level features.

Training, Validation, and Testing

In order for an NN to be a robust learner, a sufficiently large *training* dataset needs to be organised. Likewise, a *validation* dataset should be used to evaluate the NN's performance *during* training, enabling learning monitoring (and, hence, encouraging *termination* when learning plateaus). Lastly, a disjoint *testing* datasets should be used to verify its performance on unseen data after training. Practically speaking, it is absolutely vital that testing and training datasets are completely separate, whereas validation datasets can be more heuristically handled. Due to the *indirect* usage of the validation data (determining when learning should terminate, in addition to comparing the performances of various architectorial choices such as depth and breadth of layers, learning rate, activation functions (yet to be discussed) et cetera), they are often drawn from the training data foundation to ease the burden of data acquisition. This can be safely executed utilising upcoming algorithms such as *K-Fold cross-validation* [33].

Now, if one wants to teach a CNN to recognise *net structure* from *turbid* background, one needs to supply a set of images where one explicitly tells the CNN what a proper *ground truth* looks like, and furthermore test the trained CNN's performance on a separate set to determine whether or not what it learned during training was generalisable to unseen data of the same sort. Generating such data requires significant labour. It is therefore common practice, when evaluating the effectiveness of new image processing architectures, to utilise available standard *labelled* datasets (i.e. datasets whose instances are accompanied by ground truths). Popular datasets for bench-marking CV applications include the MNIST dataset of handwritten digits [34] (60.000 training instances, 10.000 test instances) and ImageNet, consisting of depicted nouns such as animals, plants, and objects [35] (>14.000.000 labelled instances).

Learning Strategies

The abovementioned philosophy of learning is called *supervised learning*. Other philosophies include *unsupervised* learning, where one does not reveal to the machine learning model a ground truth during training. This approach is for instance utilised in clustering algorithms such as *K-means* [36] and *DBSCAN* [37]. Unsupervised learning might be useful to identify multi-dimensional similarity between data instances, but is not capable of explicit *classification* per se, other than assigning to the instances a cluster identity.

Another branch of learning is called *reinforcement learning* which takes an evolutionary approach. Such a model might implicitly learn how to act in a rule-governed mileu by random perturbations, and receiving rewards or penalties based on the success of its perturbations in the environment.

In this thesis, however, the focus will be on supervised learning, developing a deep CNNs capable of processing both spatial and temporal information. The CNNs will play integral roles in an ecosystem with traditional computer vision methods (but perhaps – in new clothes), aiming to achieve what the industry truly needs: a reliable hole detection system that performs well not only in a controlled environment.

1.4 Thesis Specifics

With a shared foundational understanding of fish as a *nutritious* contribution to human diets, Norwegian aquaculture's *economical* contribution to the wealth of the nation, the *ecological* concerns regarding fish escapes due to net defects, and methods of teaching computers to *perceive*, a proper definition of this thesis and *its* contribution to fulfil all of the above can be made.

The thesis proposes an algorithm capable of discovering, highlighting, tracking, and reporting, on areas that depict net holes in videos captured during net cleaning operations. The algorithm is general in nature, allowing for use on material caught by different hardware in a wide range of environments, given that respective DNNs are re-trained on video material suitable for the new application. Full-length video sequences of the entire cleaning process, offering not ideal scenarios but real-life ones, make up the data foundation for this work, keeping results and operation rooted in reality.

This work is exploratory and has therefore not stressed real-time-usability. However, implementational choices have been considered with *future* real-time-usability in mind, and testing (chapter 8) reveals that the current implementation of the algorithms executes on typical scenes with a speed that must be doubled to satisfy real-time demands. Thus, real-time usage is likely within reach if optimal implementations are considered, for instance by migrating from the Python environment, and with effective parallelisation and hardware upgrades.

New Contributions

Common for researched work is usually a concern with identifying irregularities in single video frames, often operating under ideal circumstances. This work brings to the table a handful of new ideas. The thesis acknowledges that a realistic video stream from a net cleaning (or inspection) operation displays *more* elements than intelligible net structure. A distinct contribution is the investigation of a U-Net approach to net segmentation, contrary to traditional binarisation schemes such as Otsu's method [38] or Canny's edge detector [39].

Three different segmentation strategies were initiated, all based on the MultiRes U-Net [40]; NeNoS (Net and Nonsense Segmentation), teaching the MultiRes U-Net to separate areas in a frame depicting net structure from irrelevant areas. The second strategy, 3CAS (Three-Class Attention Segmentation) is a refined version of the first, capable of separating areas of dark net structure (relative to its background) from areas of bright net structure. Both approaches were coupled with an adaptive thresholding algorithm [41] to achieve coherent binarisation of net structure only, and disregard of anything else. The third method, NTS (Net Thread Segmentation) needs not coupling with additional binarising schemes, and yielded very favourable results not least in terms of reduced overhead as compared to the former two strategies.

To achieve temporally consistent segmentations without compromising computational efficiency was investigated a lightweight *lag mask* strategy. This scheme allows the NN to *peek* at the previous segmentation by slightly expanding the dimension of the first layer of the MultiRes U-Net. A training regime with specific methods of regularisation was developed to effectively tune the model's reliance on the lag mask contra the current video frame. This scheme proved to stabilise the segmentation to a remarkable degree, at the cost of less than a millisecond of computation per frame.

The U-Nets required labelled data, all of which had to be manually constructed. More than one thousand images were therefore gathered by careful analysis of several hours of raw video material, collecting a wide range of scenes which were all manually segmented. In addition, a representative set of *test videos* was extracted from real operation, including challenging scenes and several holes. The level of difficulty in the testing material is unparalleled by comparable works investigated in this thesis.

Another central contribution of this work is the adaptive implementation of a hole detection module similar to that proposed by Haugene [17], based on mathematical morphology [42]. The proposed hole detector is capable of detecting irregular pieces of background in a binary image by tracking the size of the local neighbourhood with an adaptive variable called *The Elbow*. The scheme works regardless of zoom level, and requires not perpendicular relationship between the camera and the fish cage net.

To enable effective filtering of sporadic (apparent) irregularities caused by flickering or moving objects, a *spatiotemporal filter* is proposed, demanding both spatial and temporal continuity in arising irregularities prior to verification. Spatiotemporal consistency has not been a topic in researched work.

A deep CNN called the *scene interpreter* has been trained to interpret not only the binary image, but rather the content of the actual video frame in areas where verified irregularities occur. The scene interpreter is based on the popular VGG16 [43] model, and specialised, through transfer learning, to separate net structure from fish and nonsense. With this addition to the overall hole detection framework – irregularities which occur due to occluding fish or oversegmentation (for instance, if the MultiRes U-Net falsely includes parts of the water column in the segmentation) are effectively ruled out. The scene interpreter was trained on 300 images extracted from the available video material, and validated on a separate set of 300 images.

Thesis Outline

- Articles and scientific work relevant for this work is introduced in chapter 2.
- The video foundation is presented in chapter 3.
- The exploration of net segmentation through MultiRes U-Nets takes place in chapter 4.
- The adaptive irregularity detection scheme is discussed in chapter 5.
- The scene interpreter that separates fish and nonsense from net structure (in which actual holes can be found) is presented in chapter 6.
- The spatiotemporal irregularity filter (which is in fact tracking of *unverified* irregularities), and, eventually, the tracking of *verified* irregularities, is discussed in chapter 7.
- The entire framework was tested on representative sequences from real operations in chapter 8.
- Lastly, a discussion of the achieved results followed by a conclusion and suggested future improvements finalise this thesis in chapter 9.

If links are preferred to the QR-codes provided in this thesis, all hyperlinks to which QR codes point are listed in appendix A.

Chapter 2

Previous Work

This chapter introduces key methods in CV and, specifically, methods relevant for the work carried out in this thesis. The chapter consists of three main parts. Firstly discussed are two Master's theses concerned with fish cage analysis. Secondly, three peer-reviewed articles on the same topic. Lastly, a discussion of articles on NN architectures that were either *used* or *considered* for use in this thesis.

Firstly, Haugene's thesis from 2014 [17] discussing robust net tear detection in fish cages. Ideas utilised by Haugene and considered (or used) by this thesis include *(i)* optical flow, and, **particularly** mathematical morphology. Those ideas have been elaborated on afterwards. The idea of mathematical morphology in a hole detection module will be thoroughly investigated in chapter 5.

Subsequently, Jakobsen's thesis from 2011 [19] and his take on net cage integrity control will be discussed. Following Jakobsen's work is introduced a set of edge detectors used to convert video frames into binary images. This conversion, called *binarisation*, is an operation carried out by **every single one** amongst researched work. A more detailed case for binarisation will also be discussed in chapter 5, where practical aspects of the different methods as they relate to this application will be further investigated.

The relevant articles present five different takes on fish cage hole detection. Neither of which, it will be argued, solve the challenge to a satisfactory degree, but all of which provide valuable insight to the diverse considerations and obstacles that need be overcome.

Lastly, three interesting NN architectures; a novel CNN that promotes information inference. The model was initially thought to segment net structure in this thesis, perhaps enabling intact net structure inference *behind* occluding objects such as fish. Secondly, an article investigating 3D CNNs, an idea seemingly relevant because this thesis analyses *videos*. The architecture was ultimately not implemented, but it contributed to motivate the development of spatial coherence encouragement in the MultiRes U-Net, the final NN architecture discussed. This model was successfully implemented in the thesis as primary segmentation module. As coda, a short elaboration of the neuron and its activation function.

2.1 Theses

2.1.1 Evaluation of Methods for Robust, Automatic Detection of Net Tear with Remotely Operated Vehicle and Remote Sensing

Noting that previous theses concerning net tear detection were mainly operating on *ideal*-like environments and single-image toy examples, Haugene [17] set to develop a *robust* net tear detection algorithm. Robust, in the sense that the algorithm should function as intended in environments with various light conditions, and in the sense that *foreground objects* occluding the net, such as fish and algae, should not be confused with net tear.

His high-level approach was the following: *(i)* construct a binary mask separating foreground from background, and *(ii)* design a structuring element s.t. a morphological closing operation [42] fills all regular background regions, but fails to fill sufficiently large background regions — indicative of a hole.

Haugene viewed *(i)* as the backbone of his thesis. To achieve this he made a design scheme which he coined *Uniform Combinatorial Design*. The idea of his approach was to use a combination of three modules (edge-, temporal background-, and optical flow based segmentation) and have them vote to create a foreground binary mask. These three modules were working on the five image channels *red, green, blue, value* and *saturation*, separately, and all votes were collected with a binary OR-operation.

A substantial part of his thesis consisted of developing a sophisticated *background* estimate. His temporal background segmentation (see, for instance, [44]) estimated the background pixels through median historical pixel values. Pixels whose values were relatively unchanging in time were considered part of the background, but sometimes *smoothly-textured* algae and fish were incorporated in the background model.

Haugene regarded his work as promising, but highlighted a few weaknesses:

1. His algorithm depended on a *myriad* of parameters which required tuning.

2. Smooth and stationary foreground elements were occasionally included in the background model. Reflectance in fish scales were sometimes confused with net tear.

3. The size of the structuring object (used to discriminate regular pieces of background from net tear) was of a user-defined fixed size. In other words, the ROV would have to analyse the net structure from a constant angle and distance to the net in order for it to work properly.

Optical Flow

The purpose of optical flow calculations is to derive, on pixel level, relative spatial movement of brightness patterns in a sequence of images. By deducing a flow vector for each pixel, one might be able to tell different objects from another, and not least describe what objects move in what directions, relative to the viewer. Several methods have been deduced to calculate optical flow. One popular iterative implementation was presented by Horn & Schunck proposed in 1981 [45].

The algorithm works if several core assumptions are met; *(i)* the surface is assumed to be flat, so no brightness patterns arise from shading differences. *(ii)* The illumination is considered to be uniform. *(iii)* Patterns are assumed to move smoothly, with no spacial discontinuities.

If so, the brightness of a point in a pattern *cannot* change with time. Thus, the brightness of point (x, y) in an image at time t can be denoted $E(x, y, t)$ and

$$\frac{dE}{dt} = 0 \tag{2.1}$$

and hence, by the chain rule:

$$\frac{\partial E}{\partial x}\frac{dx}{dt} + \frac{\partial E}{\partial y}\frac{dy}{dt} + \frac{\partial E}{\partial t} = 0 \tag{2.2}$$

By letting $u = \frac{dx}{dt}$, $v = \frac{dy}{dt}$, and E_x, E_y, E_t denote the partial derivatives of the image brightness derivatives with respect to x, y, and t, such that

$$E_x u + E_y v + E_t = 0 \tag{2.3}$$

or, equivalently

$$(E_x, E_y) \cdot (u, v) = -E_t \tag{2.4}$$

the optical flow in the direction of the brightness gradient (E_x, E_y) can be deduced:

$$-\frac{E_t}{\sqrt{E_x^2 + E_y^2}} \tag{2.5}$$

Haugene [17], on the other hand, did not follow Horn & Schunck, but rather utilised techniques of *single-* and *double differencing* which are simpler methods of optical flow. The schemes subtract subsequent frames from one another, yielding no flow vector per se, but rather *highlighting* areas of motion. For Haugene's purpose of identifying background as *something that does not move* this was sufficient.

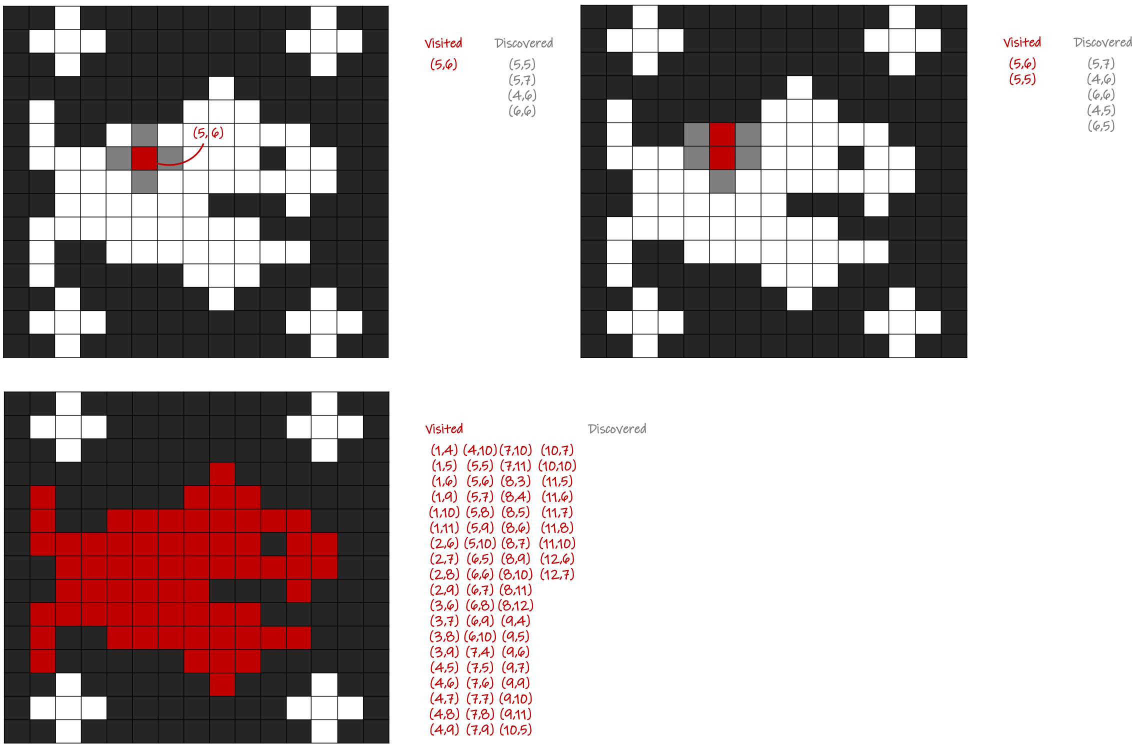


Figure 2.1: Region growing starts from a seed point and adds to the *discovered* list neighbouring pixels whose intensities satisfy an inclusion criterion. In this instance the criterion simply demands a pixel to be white. Pixels are iteratively moved from the *discovered* list to the *visited* list once their neighbourhood has been assessed. Once the *discovered* list is emptied, meaning the entire region has been visited, the *visited* list is returned. Above are shown the first two, and the final iteration of the algorithm starting from seed $(5, 6)$.

Region Growing

Region growing as often discussed in this work is a simple algorithm which, from a seed point, returns a bag of coordinates to neighbours and neighbours neighbours — and so forth — of the seed which satisfy a certain inclusion criterion. Subject to evaluation by the inclusion criterion is the pixel intensity value.

The inclusion criterion can be quite sophisticated, or simply state one or more static thresholds which decide whether or not a pixel will be included in the region. In this project region growing will be utilised on binary images, exclusively, and the criterion will simply look for neighbours that are either black or white, dependent on application. An example of a region growing algorithm looking for the region to which seed pixel $(5, 6)$ belongs is illustrated in fig. 2.1.

Mathematical Morphology

The theory of mathematical morphology includes a set of operators that have proven useful for image analysis [42]. *Dilation, erosion, opening,* and *closing*, are basic operations that can be applied to binary and grayscale images (and further extended to multi-dimensional colour spaces such as RGB) which *preserve* the shape of the original objects whilst still being capable of removing noise, filling gaps, smoothing edges, and breaking sparse connections.

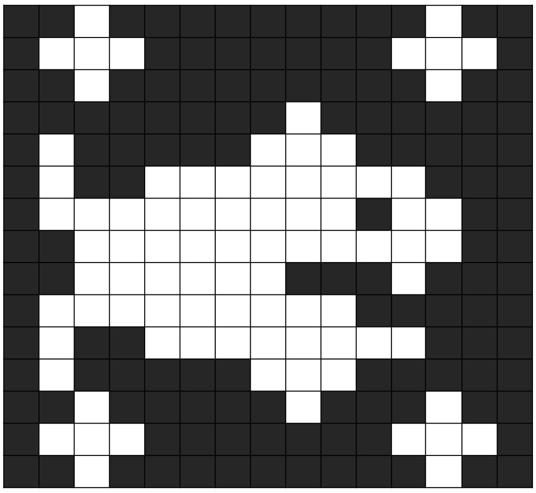
The principle of dilation and erosion is to compare the pixel intensities of an image I to a reference object with a given size and shape, called the *structuring object* B. By systematically sliding B across I, one may produce output images that are either *thinned* or *fattened* versions of the original. By combining the basic operations of erosion and dilation, one may preserve the original shape, but *scrape off* objects, or parts of objects, that are smaller than B, or, on the flip side, *fill gaps* that are smaller than B. These operations are called *opening* and *closing*, respectively, and consist of dilation and erosion in sequential order:

$$\begin{aligned} \text{I dilated by } B &= I \oplus B \\ \text{I eroded by } B &= I \ominus B \\ \text{I closed by } B &= (I \oplus B) \ominus B \\ \text{I opened by } B &= (I \ominus B) \oplus B \end{aligned} \tag{2.6}$$

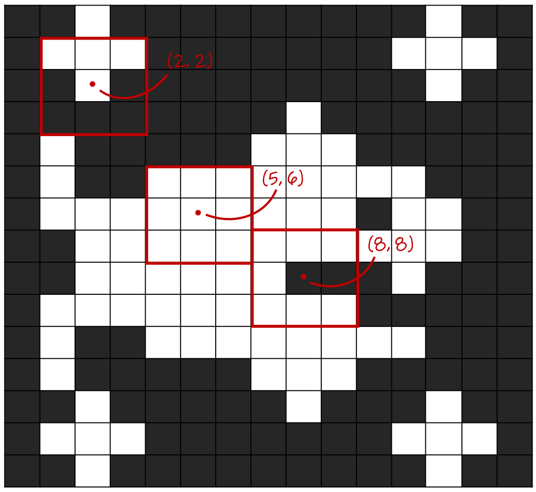
The procedures of dilation and erosion, and opening and closing, are visualised in fig. 2.2 and fig. 2.3.

Haugene [17] utilised morphology to find holes that were larger than the defined structuring element. One advantage to this approach, contrary to a region growing approach, is that one can identify holes whose shape resembles that of the structuring element, or, at least, whose shape cannot be entirely covered by the structuring element. If, for instance, poor image quality led a large, but thin background area to appear in a frame, a pure region growing approach could have concluded it to be a proper hole since a large number of background pixels were connected. However, if the hole were *thinner* than structuring element, it would have been closed by a closing operation. This phenomenon is further discussed in chapter 5.

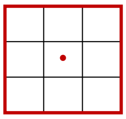
The theory of morphology was initially developed for binary images, but an extension to grayscale can easily be made. Whilst c (see fig. 2.2) in a binary case would either be set to 0 or 1, it would rather be set to the *smallest*, or *largest* value covered by B in I. This variant easily applies to RGB-images as well, where each pixel contains an array of pixel intensities for the red, green, and blue channels, respectively. The morphological operations would simply be carried out on each individual colour channel as it would on a grayscale image.



(a) Binary image I. White pixels are ones, black pixels are zeros.

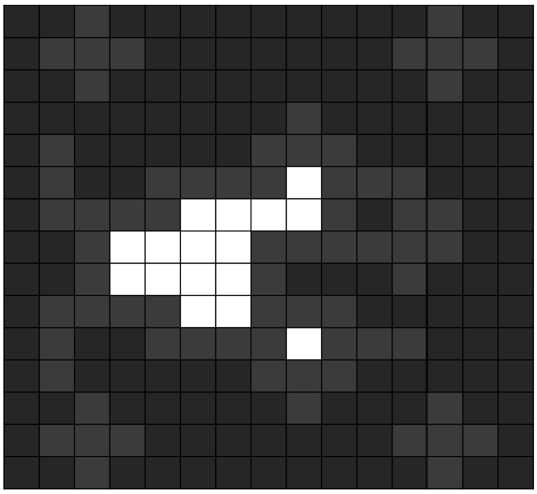


(b) Image I with the structuring element from (c) visualised.

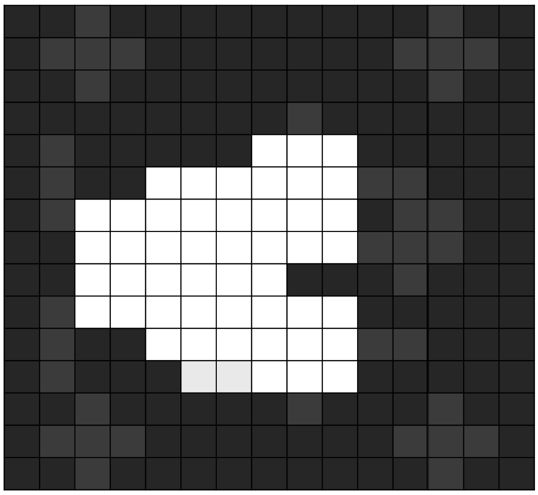


(c) A 3×3 structuring element, B, of ones. The centre is marked with a red dot.

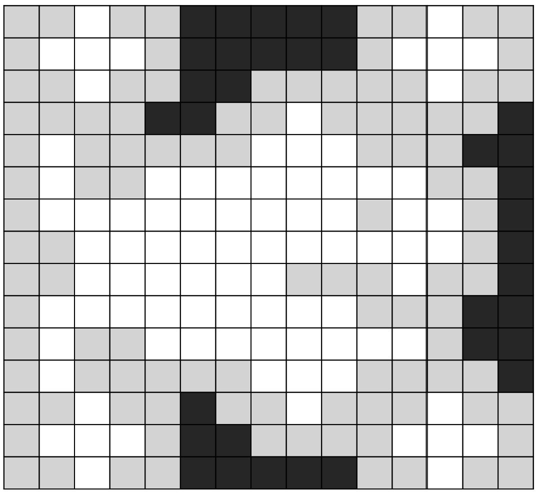
Figure 2.2: By sliding the structuring element B from left to right, top to bottom in the image I we can use either dilation or erosion to generate an output image. If we perform dilation, then the *centre* pixel c (marked with a red dot) of B in I will be set to 1 in the output image if B *hits* either set pixel in the neighbourhood of c in I. In (b) both $(2,2)$, $(5,6)$, and $(8,8)$ will be 1 in the output image in the case of dilation. If we perform erosion, then c in I will be set to zero if B does not *fill* the neighbourhood of c in I. In (b) $(2,2)$ and $(8,8)$ will be set to zero, whilst $(5,6)$ will remain 1 since B fills the neighbourhood of c in I.



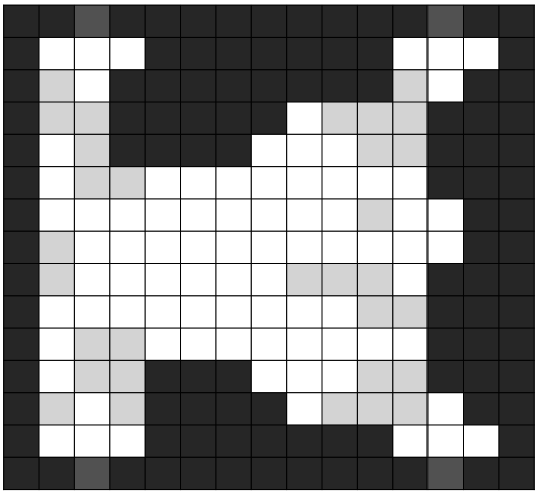
(a) Erosion: $I \ominus B$.



(b) Opening: $(I \ominus B) \oplus B$.



(c) Dilation: $I \oplus B$.



(d) Closing: $(I \oplus B) \ominus B$.

Figure 2.3: Dilation and erosion can be used in a complimentary manner to *open* an image, effectively *removing* elements from I that are smaller than B, or to *close* an image, effectively *filling* gaps that are smaller than B. Here, I and B from fig. 2.2 have been used and I has been overlayed to better visualise the effect of the operations.

2.1.2 Automatic Inspection of Cage Integrity with Underwater Vehicle

Unlike Haugene, Jakobsen's [19] work from 2011 also considered ROV hardware design, including a laser module to regulate the vehicle's distance to the fish net, and communication interfaces. His work therefore overlaps with this thesis only in parts.

Jakobsen's hole detection algorithm required ideal conditions: *(i)* the ROV is between 15 and 60 centimetres from the net. *(ii)* The camera faces the net structure more or less dead on. *(iii)* The view of the net is free from occlusions and disturbances such as fish and algae growth. His algorithm worked on each frame, independently, starting by applying to the images a binarising scheme based on histogram analysis, and, later, utilising edge detection.

Jakobsen explored the Marr-Hildreth kernel [46] (also known as the *Laplacian of Gaussian (LoG) method*) which is is an edge detector based on *first* limiting the impact of noise-induced false edges by smoothing the image using a Gaussian kernel, and then detecting edge points as zero-crossings of the blurred image's second derivatives.

Subsequently, he found the edge detector of Canny [39] to yield better results. This method is capable of tracking weak, but connected edges, and might suppress false edges more successfully than the LoG.

After constructing a binary image, Jakobsen conducted depth-first-searches to recognise straight lines spanning across the entire frame. The net integrity was verified by comparing the relative distances between the lines.

Jakobsen concluded that his results were promising, but never tested his algorithm on images that depicted structural damage.

Otsu's Method

Otsu proposed an optimal threshold selection algorithm for grayscale images [38]. By analysing the grey-level histograms, and assuming either two or more classes, he sought to find the threshold which would maximise inter-class weighted variance, or, equivalently, minimise intra-class weighted variance.

The first step of the algorithm normalises the histogram and treats it like a probability distribution:

$$p_i = \frac{n_i}{N}, \quad p_i \geqslant 0, \quad \sum_{i=1}^{L} p_i = 1. \tag{2.7}$$

where p_i is the probability of a grey level i, $N = n_0 + n_1 + ... + n_L$ is the number of pixels, and L is the number of grey levels $i = [1, 2, ..., L]$.

By assuming, for simplicity, two classes, C_0 and C_1 which are separable by a threshold k, the probabilities of each class occurrence is defined by

$$\begin{aligned}\omega_0 &= Pr\{C_0\} = \sum_{i=1}^{k} p_i = \omega(k) \\ \omega_1 &= Pr\{C_1\} = \sum_{i=k+1}^{L} p_i = 1 - \omega(k)\end{aligned} \tag{2.8}$$

and their mean values are

$$\begin{aligned}\mu_0 &= \sum_{i=1}^{k} iPr\{i|C_0\} = \sum_{i=1}^{k} ip_i/\omega_0 = \mu(k)/\omega(k) \\ \mu_1 &= \sum_{i=k+1}^{L} iPr\{i|C_1\} = \sum_{i=k+1}^{L} ip_i/\omega_1 = \frac{\mu_T - \mu(k)}{1 - \omega(k)}\end{aligned} \tag{2.9}$$

where μ_T is the mean pixel value of the entire image and $\omega(k)$ and $\mu(k)$ are the zeroth- and first-order cumulative moments up to k.

From this one can deduce the class variances

$$\begin{aligned}\sigma_0^2 &= \sum_{i=1}^{k} (i - \mu_0)^2 Pr\{i|C_0\} = \sum_{i=1}^{k} (i - \mu_0)^2 p_i/\omega_0 \\ \sigma_1^2 &= \sum_{i=k+1}^{L} (i - \mu_1)^2 Pr\{i|C_1\} = \sum_{i=k+1}^{L} (i - \mu_1)^2 p_i/\omega_1\end{aligned} \tag{2.10}$$

which can be used with the class probabilities to formulate cost function expressions such as the *weighted within-class variance* $\sigma_w^2 = \omega_0\sigma_0^2 + \omega_1\sigma_1^2$ and the *weighted between-class variance* $\sigma_B^2 = \omega_0(\mu_0 - \mu_T)^2 + \omega_1(\mu_1 - \mu_T)^2$.

Otsu noted that calculating σ_B^2 is based merely on means and is therefore an easier operation than calculating σ_W^2, and therefore favoured the cost function $\eta = \sigma_B^2/\sigma_T$ where σ_T^2 is the total image variance, independent of k. The optimal threshold k^* is thus the one that maximises η and hence maximises the weighted inter-class variance.

Otsu's method can be generalised to multi-class problems, and will perform well in situations where classes adhere strictly to disjoint sets of pixel values.

Marr-Hildreth's Laplacian of Gaussian Method

Contrary to *approximating* the first derivative (through for instance a Sobel [47] or Prewitt kernel) and finding its maxima, the Laplacian of the Gaussian [46] can be *analytically* deduced with no need for approximation. The Gaussian is defined as

$$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.11}$$

where σ^2 is the variance of the distribution. This property is set by the user. By convoluting the original image I with G we blur the original image where the value of σ decides the level of blurriness. The level of blurriness will further decide *how strong* an edge has to be in the original image in order to come through as an edge in the blurred image.

Due to properties of convolution, the Gaussian kernel and its Laplacian needs only be calculated once. Hence, the Marr-Hildreth method is extremely efficient, only requiring for each image a convolution with a pre-calculated LoG-kernel:

$$\nabla^2(G(x,y) * I(x,y)) = (\nabla^2 G(x,y)) * I(x,y) \tag{2.12}$$

where ∇^2 is the Laplacian operator:

$$\nabla^2 f(x,y) = \frac{\partial f(x,y)}{\partial x^2} + \frac{\partial f(x,y)}{\partial y^2} \tag{2.13}$$

Canny's Method

Canny's *computational approach to edge detection* [39] from 1986 aimed to create an edge detector which satisfied the following criteria: *(i)* being a good detector in the sense that non-existing edges will not be marked, and existing edges will not fail to be marked. *(ii)* Being a good localiser in the sense that marked edges will be as close as possible to the centre of the edge. *(iii)* Yielding one and only one response to a single edge. Canny achieved this by first convoluting the original image with a Gaussian kernel (similar to the Marr-Hildreth method) and then approximating the first derivatives of the resulting image, G_x and G_y, for instance by utilising the Sobel kernel in $x-$ and $y-$direction. For each pixel (x,y) one can then identify a direction θ and gradient G:

$$\begin{aligned} \theta &= tan^{-1}(\frac{G_y}{G_x}) \\ G &= \sqrt{G_x^2 + G_y^2} \end{aligned} \tag{2.14}$$

Furthermore, Canny suggested edge thinning through *nonmaximum suppression*: By analysing every pixel's neighbouring pixels in the *gradient direction*, a decision would be made either to *suppress* the current pixel if it weren't a local maximum, or let it prevail. This step satisfies criterion *i* and *ii* from the checklist; representing each edge by a single, strong response.

The remaining edge responses were further subject to *double thresholding* where responses below a lower threshold T_l were suppressed and those above an upper threshold T_u were verified. Responses between T_l and T_u were verified through *hysteresis*, that is, if and only if they were part of an edge with already verified edge points. Canny's algorithm thus allowed weaker parts of edges to be included in the final product if other parts of the edge yielded a sufficiently strong response.

The specific values of T_l and T_u should be tuned by the user based on application and noise properties. Canny suggested the relationship of T_u to T_l to be approximately two or three to one.

2.2 Articles on Net Cage Inspection

2.2.1 Automated fish cage net inspection using image processing techniques

Paspalakis et al. [21] proposed in a recent paper two main strategies to detect net tear. Their first approach was designed to be easily parallelisable: the frame was binarised using Otsu's method [38] and then divided into an grid of overlapping cells. The sums of pixel intensities were calculated per grid cell, and cells with a significantly low sum were considered to be irregular. Specifically, these were cells that failed the 0.05 p-value test under the assumption of a normal distribution.

This approach might be easily parallelisable, but fails to give anything but a crude estimate of where a net tear might be. In addition, if images are less than ideal, it cannot be said to be a robust method. For instance — if the net is covered by growth in half the image, the binary interpretation of the image might represent the net as *thicker* in that half, meaning cells that cover this area will have a much higher accumulated count of net pixels than the other half.

Their second idea introduced the detection of *Hough lines*. After binarisation, the Hough lines were compared to their nearest net pixel in the binary image. The net was assumed to be intact where it closely followed the suggested Hough line, and broken where the Hough line had no close contact with an edge point.

Paspalakis et al. considered their results to be good and promising. However, all examples depict *straight* net structure spanned across the entire image, with no noise neither in background nor in the foreground. They initially discussed the fact that net structure rarely appears in straight lines but deforms spatially to form curves rather than lines. Haugene also discovered this is his work [17] and found it hard to find Hough lines under typical circumstances.

The Hough line transform

The Hough line transform [48] can be used to identify straight lines in a binary images through a series of *votes*. The transform can, however, be extended to identify *any* shape that can be represented mathematically, so an extension could hypothetically be made to fit the needs of a net structure identifier.

Straight lines can be explained by $y = mx + c$ or, equivalently, $\rho = x\cos(\theta) + y\sin(\theta)$ where ρ and θ denote the length and the orientation, respectively, of the line's normal vector to the image origin. Since every *line* in the (x, y)-space can be described by a (ρ, θ)-pair, it translates to a *point* in the (ρ, θ)-space, also called the *Hough space* for 2D-lines.

At each edge point in the binary image, the Hough Line Finder searches for lines by iterating through (ρ, θ)-pairs. For each edge point that such line hits, votes accumulate. After analysing each and every point, a threshold can be determined for which (ρ, θ)-pairs that exceed this threshold represent the most prominent straight lines in that image.

2.2.2 An integrated ROV solution for underwater net-cage inspection in fish farms using computer vision

Betancourt et al. published late in 2020 a very interesting paper proposing a new take on fish cage analysis in real-life environments.

Their approach resembled other works with respect to several aspects such as the initial binarisation of each frame with Otsu's method. Following binarisation, they applied the Hough line transform to recognise the mesh structure. At this stage, they deduced from the intersection of the detected lines the location of the knot points in the net. From this information they reconstructed the depicted net structure, digitally, and recognised holes where knot points lacked connections.

The authors tested their scheme on a real fish cage. However, their results section depicts only staged test-images, on which their algorithm performed decently – reconstructing the net structure with high accuracy and recognising 79% of present holes. Questions could though be raised as to how robust their approach is when considering the fact that real-life video not always represents the net structure perfectly, and that net structure *will* appear broken in occasional frames. Challenges such as algae growth and occluding fish (which are indeed crucial talking points in real-life fish cage inspection applications) are not discussed, so it remains unclear how well this framework performs under such conditions.

2.2.3 An adaptive method of damage detection for fishing nets based on image processing technology

Zhao et al. [22] carried out recent work on fish cage damage detection utilising two methods; one based on knot point detection (in a sense like [20], but their knot detection schemes and their usage of the information differed), and one based on mesh hole area comparison.

Their first method consisted firstly of a manual *region of interest* (ROI) selection in the image – acknowledging that only some sub-part of the image is fit for analysis. Their assessment philosophy originated in the idea that considering the net structure as a whole would be too inefficient. Their solution was to reduce its integral information to a set of features, and the most prominent net structure descriptors, they argued, were the locations of knot points. After applying Harris corner detection [49] to their grayscale image (an algorithm similar to an edge detector that searches for conjoined edges) and, subsequently, analysing the distribution of corners within the ROI.

The authors assumed that intact net would distinguish itself from damaged net, clearly, by boasting regular patterns in corner distributions. Algae growth caused this assumption to crumble, however, effectively hindering the detection of corners. Realising this, they changed focus from knot points to *mesh holes*, synonymously with what we later in this thesis call *Background*.

With their new approach, they considered *clear images* only, noticing that several frames from video streams were affected by motion blur. Subsequently, they applied some filtering and binarised the image with Otsu's method [38]. Recognising that Otsu's global threshold tends to misrepresent net structure in certain scenarios (a single global threshold does not exist in complex images. This is further discussed in chapter 5, see for instance fig. 5.4), they applied morphological closing to the binary image to repair broken connections. After this, they compared the area of each and every mesh hole and extracted from the characteristics of the distribution significant deviations, if any.

Area comparison, essentially a region growing approach, has a few drawbacks. This is further elaborated on in chapter 5 (see fig. 5.9). For instance: *(i)* the algorithm breaks down if the binary image is slightly corrupt. That is why morphological closing was applied. This will eliminate some trouble, but not all. *(ii)* The counting of every pixel of every mesh hole is a tedious operation, and *(iii)* if the camera is tilted, not facing the net structure dead on, significant damage far away from the camera may have a smaller area than intact mesh holes closer to the camera.

The proposed algorithm of Zhao et al. identified holes in test images, and successfully so, also with some algae growth present. Their choice of mesh holes as subject to scrutiny, as opposed to net structure directly, is interesting, but their implementation is too immature for any robust and practical application.

2.3 Neural Network Architectures

2.3.1 Spatial As Deep: Spatial CNN for Traffic Scene Understanding

Pan et al. claimed in 2018 to improve the conventional CNN by enabling message passing between pixels across columns and rows in a layer. The method provided great results in traffic lane segmenting, where large continuous shapes may be occluded by objects such as lamp posts, pedestrians, and cars. If this architecture could *infer* information that was partly occluded, perhaps it could also identify net structure where it is partially disrupted by fish, growth, or perhaps holes.

Their key idea was instead of sliding filters over the entire frame, to slide filters over each *column* and *row* of the feature map and thus treat them as layers to which convolution and the following nonlinear activation function is applied, before it is passed on to the next layer. This architecture allowed richer information flow between neurons in the same original layer.

The model outputs probability maps (*probmaps*), and *pixel level targets* were used during training. Lane markings whose existence probability exceeded 0.5 were considered, and the *Union over Intersection*, also know as the *Jaccard index*, was used for scoring.

The spatial CNN outperformed competing state-of-the-art NNs such as the ResNet-101, the MRFNet, and the ReNet in most scenarios when tested on traffic lane segmentation. It seemed to infer well then nature of the lane segments even when not completely visible.

2.3.2 3D Convolutional Neural Networks for Human Action Recognition

Ji et al. proposed in 2013 to expand the conventional 2D CNN to extract features from a stream of images, instead of singular frames. Thus encompassing temporal, as well as spatial, dimensions. The material at hand was a set of surveillance videos from which they aimed to detect human actions.

From 2D to 3D convolution

In conventional 2D convolution, the value of a unit on the coordinate (x, y) in the jth feature map in the ith layer is given by

$$v_{ij}^{xy} = \tanh\left(b_{ij} + \sum_{m}\sum_{p=0}^{P_i-1}\sum_{q=0}^{Q_i-1} w_{ijm}^{pq} v_{(i-1)m}^{(x+p)(y+q)}\right) \tag{2.15}$$

where b_{ij} is the bias of the feature map at the current layer, m is the set of all feature maps in the previous layer (which are connected to the current), P_i and Q_i are the height and width of the kernel, w_{ijm}^{pq} is the weight of the kernel in the position (p, q) which is used to generate the current feature map, and $v_{(i-1)m}^{(x+p)(y+q)}$ is the value of the unit on the coordinate $(x+p)(y+q)$ in the previous layer of feature map m. The parameters b and w are typically learned during model training. Resolution can be reduced by pooling between layers, for instance *Max Pooling*. Max Pooling preserves only the largest value within a window, and can thus be used to downscale an image whilst preserving the most valuable information, which is considered to be carried by the most intense edges.

To capture motion, several adjacent frames were considered in a 3D convolution process:

$$v_{ij}^{xyz} = \tanh\left(b_{ij} + \sum_{m}\sum_{p=0}^{P_i-1}\sum_{q=0}^{Q_i-1}\sum_{r=0}^{R_i-1} w_{ijm}^{pqr} v_{(i-1)m}^{(x+p)(y+q)(z+r)}\right) \tag{2.16}$$

where R denotes the third (temporal) dimension of the kernel.

Three different actions were classified: *CellToEar*, *ObjectPut*, and *Pointing*. They first identified human heads using some pre-trained structure, then drew a rectangular box encapsulating the entire human, whose size was deduced from the scale of the head. The content of the video inside the rectangular box (they simply required a human to keep themselves within the boundaries of the box for some time) were collected from consecutive frames to make up a cube of data.

The results were good, but they required a lot of labelled data. The team set to explore unsupervised 3D CNN structures later. Given that this paper is a few years old, progress might have been made on this field. It seems, however, that information encapsulated in the *temporal* dimension should be utilised in the net cage analyser. Since *video* material is available, and not simply still images, movement should be used to make the application state-of-the-art.

2.3.3 MultiResUNet: Rethinking the U-Net architecture for multimodal biomedical image segmentation

Ibtehaz & Rahman reinvented in 2020 the classical U-Net widely used in biomedical image analysis with their *MultiRes U-Net* [40]. Where CNNs typically require vast amounts of annotated data, U-nets perform well even on scarce amounts of training data, probably due to their ability to generalise through the *decoder* and *encoder* architecture.

A U-Net consists of an encoder that downsamples images through convolutions and MaxPooling, and a decoder where images are upsampled through deconvolutions. A key feature of the architecture are the *skip connections* that output from each of the convolutional layers of the encoder, *before* MaxPooling, and connect to the respective upsampling layer where the information from the encoder is concatenated with the feature map provided the decoder. Thus spatial information lost in the MaxPooling might be retrieved in the upsampling process.

In the paper, Ibtehaz & Rahman remarked that — although the use of skip connections seems clever — that it is peculiar to concatenate the output of the *first* convolutional layer of the encoder (presumably detecting low level features) with the *last* convolutional layer of the decoder (being much more processed and presumably reconstructing high level features). To fix this, they applied a *Res path*, a series of convolutional operations to the skip connections, to have their resolution better match the feature map in the decoder to which they were concatenated.

A series of datasets were used to train and test their architecture. Some containing only a few dozens of images, others a few thousands. All activation functions were Rectified Linear Units (ReLus), except for in the output layer where they exploited the sigmoid to output a probmap. Again, the Jaccard index was used to score the prediction versus the binary mask representing the ground truth.

This research sparked an interest with respect to net structure segmentation, as the task of segmenting *cancerous cells* from healthy tissue and *net structure* from turbid water may pose similar challenges. In either scenario a binary ground truth can be supplied, and a predicted binary mask could help identify *what part of the image contains net structure* and thus help discard irrelevant parts of a frame. In addition, generating a few dozens or even a few hundred training images might very well be a feasible task during a semester or two.

A Note on the Perceptron and its Activation Function

NNs as discussed in biology consist of *neurons* and their interconnections called *synapses*. As we strive to learn new things, rewards in the form of dopamine kicks strengthen those interconnections which lead to desired outcomes. Consequently, learned behaviour and memory is stored as fine-tuned weighted nets of interconnected neurons in the brain.

In machine learning, neurons are modelled as linear classifiers called *perceptrons* and the synapses as weighted connections (see fig. 2.4). Adjacent perceptrons make up *layers*. The first layer of a NN is called the *input layer* which consists of one perceptron per input variable. For instance, one perceptron per pixel if the input is an image. The last layer is called the *output* layer and consists of one perceptron per output variable. For instance, *one* perceptron in a regression problem, one perceptron *per class* in a classification problem, or one perceptron *per output pixel* in a segmentation problem. Layers in-between the input- and the output layers are called *hidden layers*.

Interconnections between layers vary depending on architecture, and so does the number of perceptrons in the different hidden layers. If *every* perceptron in layer i is connected to *every* perceptron in layer $i+1$, the network is called

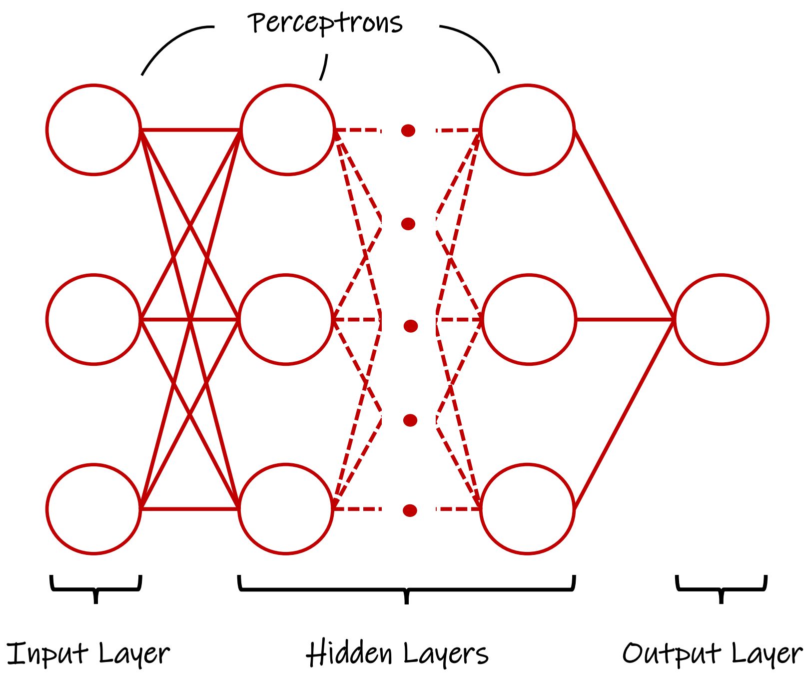


Figure 2.4: A neural network consists of interconnected linear classifiers called perceptrons. Adjacent perceptrons make up a *layer*, and layers between the input- and output layers are called *hidden layers*.

fully connected. However, perceptrons in one layer might very well be connected only to a subset of subsequent perceptrons, or even skip a layer or feed back to previous layers.

The perceptron's *activation function* determines the mapping between the its inputs (x), weights (w), internal bias (b) and the resulting output. The S-shaped *sigmoid* function (see fig. 2.6) which squashes its input between zero and one has seen wide application, but is currently ruled out by the better-performing ReLu function in deep NNs [30]. However, it still makes sense to use a sigmoid in the output layer in [40] since the desired output is a map of probabilities.

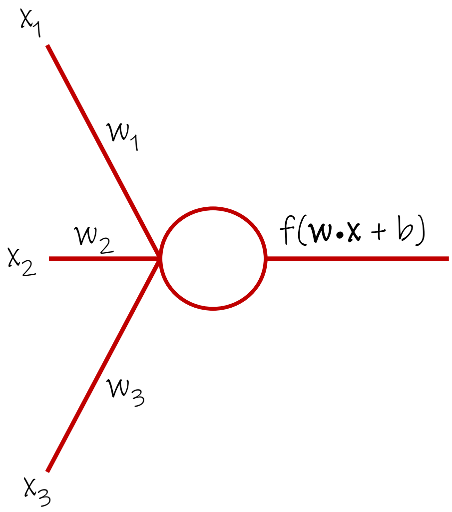
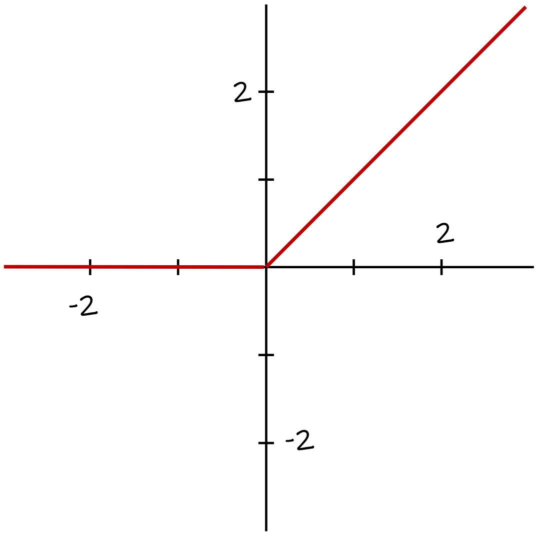
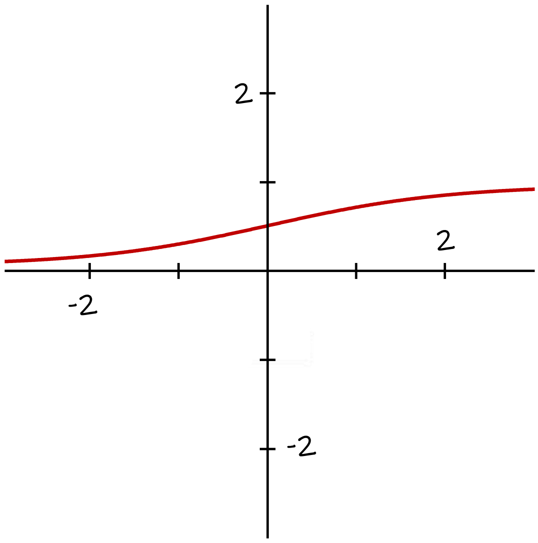


Figure 2.5: The perceptron is modelled as a linear classifier which receives inputs $\boldsymbol{x}$ from (typically) previous perceptrons which are weighted by their respective, trainable, weights $\boldsymbol{w}$. The perceptron adds a trainable bias b to the dot product of $\boldsymbol{w}$ and $\boldsymbol{x}$, and applies to it a non-linear activation function f.



(a) The ReLu function yields zero for negative inputs and a linear response to positive inputs. The x-axis describes input x, and the y-axis the response $f(x)$.



(b) The sigmoid function squashes its input between zero and one with its characteristic S-shape.The x-axis describes input x, and the y-axis the response $f(x)$.

Figure 2.6: Non-linear activation functions allow the networks to learn complex patterns. The sigmoid was traditionally favoured, but ReLus handle better through their simple nature and positive linearity the issue of vanishing gradients and forgetfulness in very deep networks [30].

Chapter 3

Video Material

Videos contributed to this project by Frøy gruppen were captured during net cleaning operations and inspections along the Norwegian coastline in 2018, 2019, and 2020. The videos vary in quality and appearance, as multiple vessels each with their distinct equipment were engaged in the different operations. In addition – weather and lighting conditions added a not negligible touch to each video. This chapter introduces, briefly, layout and properties of the different ROVs and their videos used in this work.

3.1 Flying Net Cleaner

The Flying Net Cleaner (FNC) [50] is produced by Sperre ROV Technology, part of AKVA group. The ROV manoeuvres along the net with six thrusters, carefully cleaning it with high-pressure seawater. The two videos as made accessible by Frøy gruppen (reffered to as FNC1 and FNC2) are filmed at 30 frames per second with a resolution of 1920 x 1080 pixels. Each frame shows parts of the ROV, watermarks, and frequent interactions with cleaner fish along the net wall, as can be seen in the sample frames in fig. 3.1. The cleaning operation is typically carried out at a speed of 1 m/s.

3.1.1 Challenges

Approximately 40% of each frame is useless with respect to net inspection, consisting of ROV parts. In addition, the FNC seems to attract, or, at least, not to frighten, fish to the degree as does the Manta robot (whose nature will soon be discussed). Interactions with cleaner fish is a pretty common sight in both cleaning operations with FNC analysed for this work, posing challenges which must be met when developing a robust algorithm for net tear detection. Furthermore – as the robot dives deep its lights turn on – which in turn alters the appearance of the net and its surroundings in the video.

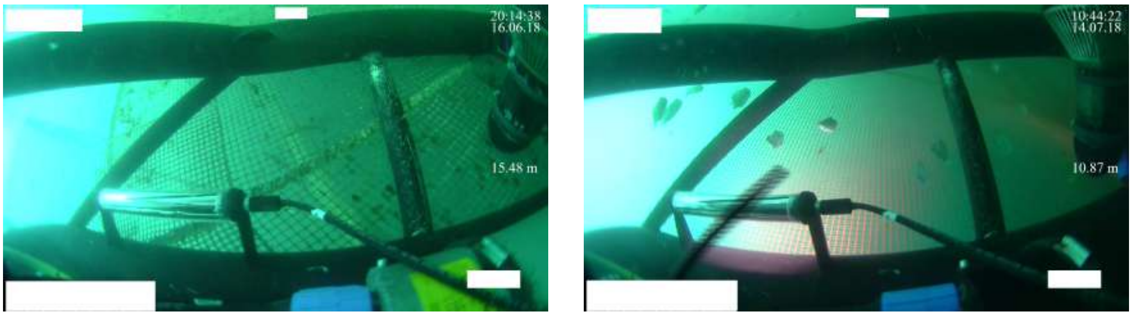


Figure 3.1: Typical scenes from the FNC include parts of the ROV and curious cleaner fish. The raw frame is tagged with date, time, and current depth. Certain watermarks have been censored for confidentiality reasons.

Figure 3.2: The Manta shows part of its construction in the port and starboard views, but usually clear shots in the fore view and – more often than not – turbulent waters in the aft view. Due to this three-views-a-row format each view measures approximately 600 x 500 pixels. The frames are usually tagged with date and time, but not always. Certain watermarks have been censored for confidentiality reasons.

3.2 Manta Net Cleaner

The Manta Net Cleaner (Manta) is produced by Stranda Prolog [51] and, like the FNC, uses thrusters for propulsion along the net, cleaning it with seawater jets. Unlike FNC, the Manta generates videos from four different views; fore, aft, starboard, and port. The raw frame format (see fig. 3.2) means each view is approximately 600 x 500 pixels whilst the entire frame measures 1920 x 1080 pixels, captured at 30 frames per second. The Manta moves as quickly as the FNC, but its footage is often clearer. The *four* videos granted by Frøy gruppen are referred to as MANTA1, MANTA2, MANTA3, and MANTA4.

3.2.1 Challenges

Amongst Manta's challenges is definitely the resolution of the input images. Whereas inspection type videos (yet to be discussed) and FNC videos provide 1920 x 1080 pixel resolution of a single view, the Manta views are one sixth the

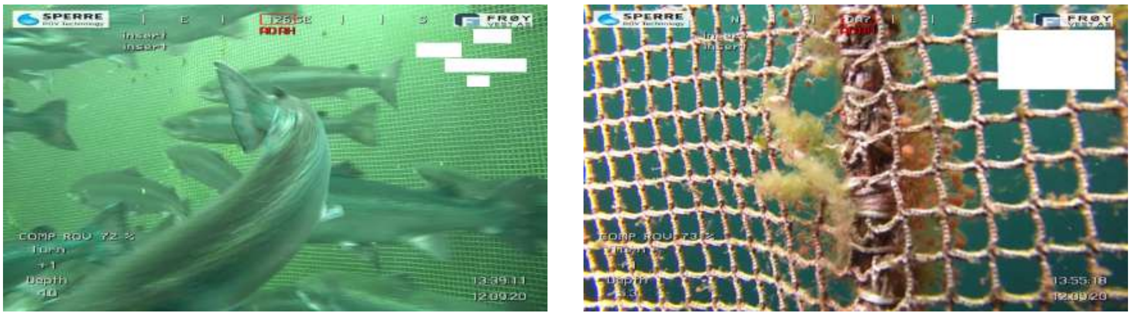


Figure 3.3: The inspection videos are primarily focused on irregularity detection and offer thus unique close-ups of suspicious areas. The inspection ROV is probably less invasive in the cage than the cleaner ROVs and the presence of schools of curious fish is more common than uncommon. Certain watermarks have been censored for confidentiality reasons.

size. Down-scaling and resizing of input images is indeed an important part of deep learning and image analysis, but larger input images offer in general greater flexibility in terms of methods. An image of larger resolution could be analysed with a sliding window making use of all information captured in that image, whilst smaller images might already have ruined reality by compressing large chunks of information into few pixels. This issue specifically manifested itself when exploring the possibility of evaluating irregular parts of an image with the scene interpreter (chapter 6), a CNN used to recognise fish or holes in small sections of the image. If the image itself is small, then a fraction of that image is necessarily smaller, and unnecessarily unclear.

Like the FNC, its construction occludes a substantial part of certain views, 35% in the port and starboard views. Unlike the FNC, fish rarely interact directly with the Manta.

3.3 Inspections

Whereas tear detection comes second to cleaning during a cleaning operation, it comes second to nothing during an inspection. The inspection video offers slow-paced shots of the net structure, and the ROV operator clearly take their time to investigate and zoom in on suspicious areas (see fig. 3.3). The ROV is produced, like FNC, by Sperre ROV Technology, and the resulting frames show no parts of the ROV, filmed at 30 frames per second. The resolution is 1920 x 1080 pixels. The single inspection video provided us is referred to as INSPECTION1.

3.3.1 Challenges

The Achilles' heel of the inspection videos is the ever-presence of fish occluding its view of the net. Disregarding every instance of a swimming fish could be a difficult mountain to climb. Contrary to cleaning videos (who more often than not have a rather stable distance to the net), the zoom level varies significantly within the inspection video; sometimes the net structure is extremely up-close, and then extremely far away at other times. This could pose challenges to segmentation modules and hole detection modules (NNs should at least be exposed to net structure with very diverse zoom ratio during training).

Its strengths, however, are many; the video quality seems to be higher than that of the cleaner robots, it is slow-paced with little motion blur, and the inspection shots tend to capture the net structure straight on, different from a cleaning operation where the ROV, as it were, *levitates* on the net structure, filming the net from the angle one would view the road when driving a car.

3.4 Usage

Certainly, access to several hours of video material caught by a wide range of ROVs is a luxury, but lack of constraint is not the best facilitator for progress. The Manta videos have been selected as prime target for scrutiny in this thesis. This choice is partly motivated by the idea of killing two birds with one stone – whereas inspection videos' main concern is with hole detection, applying an automatic hole detection module to a net cleaning operation effectively achieves *more*. Additionally, these videos pose several interesting challenges that slow-paced inspection videos do not. For instance – the presence of heavy algae growth and flying objects. Compared to the FNC, the Manta typically yields sharper footage.

Hence, hole detection has been tested on Manta videos, specifically on MANTA1 and MANTA4, whereas MANTA2 and MANTA3 and footage from other ROVs have been utilised to train the NNs. It is important to note that the ruling out of FNC and inspection type videos during testing does not mean the developed framework is not applicable to those operations in the future.

Chapter 4

Attention: The U-Net

The *near impossibility* of perception is a well-known headache in CV history. This mantra might eventually resemble a broken record, but it is absolutely fundamental as motivation for introducing the U-Net. Whereas the next chapter will introduce the rigid, rule-based, local irregularity detector (chapter 5), this chapter seeks to solve the much less concretisable problem of *attention*.

It is not indisputably evident that bottom-up approaches to object detection duplicate well what humans do. A body of optical illusions demonstrate how easily our brains are fooled – how it often seems to apply *top-down inference* rather than *bottom-up* analytical calculation. For instance – a famous experiment conducted by the cognitive scientist and psychologist Daniel J. Simons in 1999 pinpointed how detrimental lack of attention is to basic perception; Simons presented his Harvard students with a 25-second video displaying what seemed to be a game of basketball. Two groups of three, wearing white and black t-shirts, moved across the scene, passing the ball amongst team members. Simon's students' mission was seemingly simple: they were to count the number successful passes made by the white team. Surprisingly, whether or not the students got this number right was of minor importance. Half failed to notice the 6-feet gorilla that entered the scene, casually, strolling to the centre of action, beating his chest a few times, then walking off. The gorilla acquired himself eight seconds of screen time, but many busied themselves too much with counting passes to recognise the unexpected [52].

Likewise, ROV operators report to first and foremost focus on their primary job: the net cleaning operation. Their attention to algae covered cage net could actually be inhibitory to their hole detection capabilities in sections of clean net.

4.1 Where Holes Are At

Holes cannot be found within schools of fish. They cannot be found in video segments too pixelated to interpret, and they cannot be found within ROV parts. A hole detection scheme should therefore, arguably, pay attention to areas within which holes can appear, and not elsewhere. This aspect of the problem has not

Figure 4.1: The suspicious dark spot in the middle of intelligible net structure is very apparent to a humans. However, it is perhaps not well enough appreciated how different elements are intuitively disregarded as uninteresting during hole detection. Parts of the ROV, the blurry net floor, and the distant net wall, make up most of the image, but holes cannot be detected there.

been discussed well enough in previous research, all of which concerned with hole detection schemes, and not attention modules, implying the assumption that interpretable net structure covers the entire screen.

Disregarding nonsense in fig. 4.1 is a necessary precondition to identify the plausible hole in its net structure. An alien object, part of the ROV, covers more than a third of the view. The bottom net structure is too blurry to be analysed. Most of the net wall is too far away. Only a certain part of the image, in the leftmost centre, depicts net structure clearly enough to conclude that the dark spot represents an anomaly.

4.1.1 Introducing the MultiRes U-Net

Few traditional tools exist to successfully, and effectively, disregard everything that is irrelevant. Hypothetically – edge detectors can reveal regions of sharp transitions, which would effectively discard blurry regions. A binary mask could be used to scrape off any static ROV parts. But what about fish, algae, and picture corruption? These are anomalies intuitively discarded, perhaps top-down, without there being any rigid bottom-up rules that apply.

U-Nets have been successfully exploited in applications such as biomedical imagery to separate cancerous moles from healthy ones. If this task, apparently neither comprehensible in terms of rigid rules, is analogous to detecting analysable net detection in underwater scenes, then the U-Net could seamlessly solve the problem of attention. Having solved this problem, if the U-Net can tell us where to focus, the remaining task is entirely translatable to what has been tackled by previous research, where one can assume that everything in focus is of hole-detection relevance. This idea – this *paradigm* – is depicted in fig. 4.2.

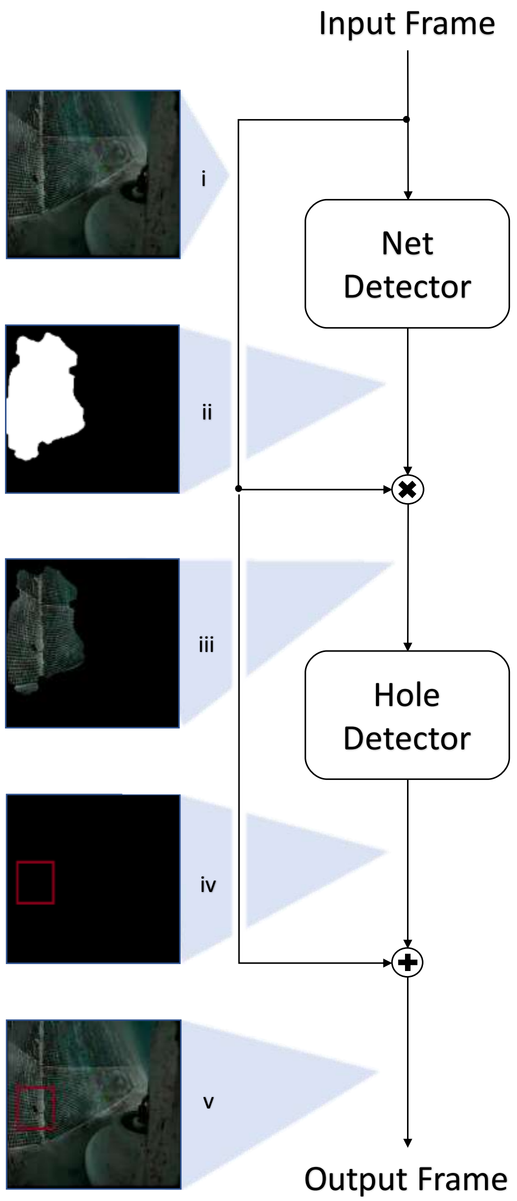


Figure 4.2: Detecting the hole in the Input Frame (i) can be achieved with traditional methods if one first designs a Net Detector, such as a specialised U-Net. The suggested attention mask (ii) can be combined with the Input Frame to produce a focused input (iii), an analysable base image for the Hole Detector (chapter 5). Ideally, the Hole Detector recognises the hole and produces some highlighting mask (iv) which is applied to the Input Frame to produce the Output Frame (v).

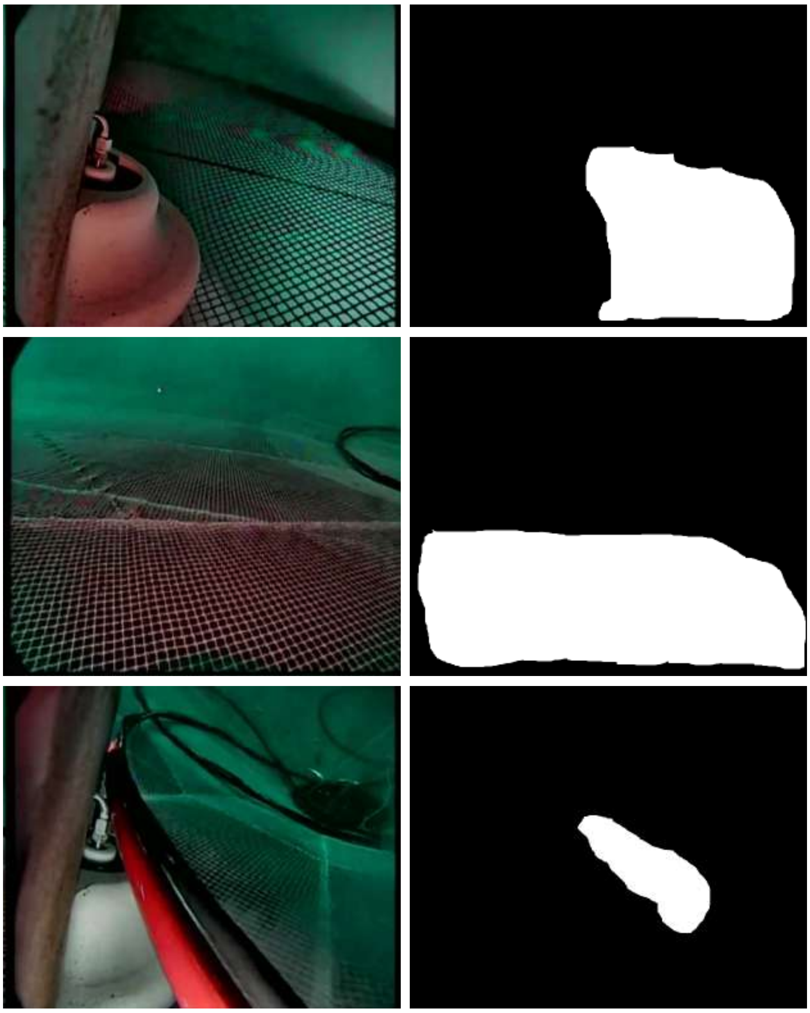


Figure 4.3: The dataset prepared for the U-Net to differentiate between Net and Nonsense consisted of 77 example views and binary masks highlighting what parts of the view contained assessable Net.

4.2 NeNoS: Net and Nonsense Segmentation

A proof-of-concept sized training dataset was constructed from one Manta video (MANTA2, see chapter 3). In total 77 images of size 600 x 500 pixels were extracted from all four available views (see fig. 3.2) and their binary attention mask counterparts were hand-crafted with the open-source graphics editor GIMP [53] (see fig. 4.3). The images were resized to 256 x 256 pixels, and three MultiRes U-Net models were trained in parallel in a K-Fold [33] manner. By dividing the relatively small dataset into three parts, and providing each model with two parts for training and the other for validation, one could easily deduce to what degree the problem was learnable, minimising the concern with lucky or unlucky validation splits[1].

[1]When validating the performance of a NN, it is common practice to withhold at least 20% of your data for validation purposes. If your data foundation is scarce, however, undesirable situations may occur where your small validation set is either very easy or very hard. K-fold attempts to solve this problem.

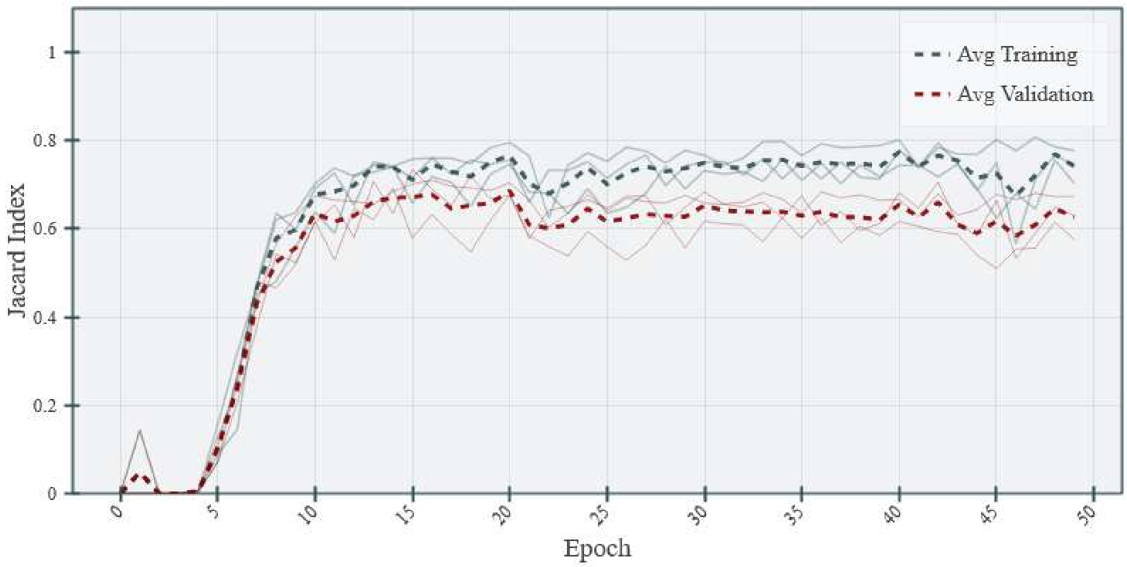


Figure 4.4: Convergence is reached relatively fast for the NeNoS 3-Fold models training on 77 Manta images. A Jaccard validation score of 60%-70% is not bad, considering that the ground truth is not necessarily precisely sketched by the author. Trends mapping inputs to output were however definitely picked up by the models, as learning and convergence is evident in the first 20 epochs.

All three models reached validation Jaccard indices of 60 - 70% after 15-20 epochs, before showing signs of overfitting (see fig. 4.4). These results were initially uplifting, and the models seemed to apply well to unseen scenes from separate videos. For instance, the suggested attention mask (ii) in fig. 4.2 is generated by one of these models, based on the Input Frame (i) which is from a separate video from the training video (tested on MANTA1, trained on MANTA2, chapter 3). Additionally, it is crucial to recognise the certain element of randomness and triviality that is introduced when a human produces ground truths for 77 images. Reaching close to 100% scores on validation data would imply some integrity leakage between train and test data, since the mapping between input and output cannot be unambiguously calculated for as long as a human decides whatever mask constitutes a reasonable ground truth for a certain image.

This procedure could very well have been scaled up and claim to solve the problem of attention completely. Perhaps it does, in some other application. However, the idea of a two-class attention mask did not work optimally with the proposed hole detection scheme of chapter 5, specifically, in its binarising scheme. Problems arose with what has been coined *the bright background phenomenon*.

4.2.1 The Bright Background Phenomenon

Without engaging too much in what is a yet to be discussed module, it should be quickly noted that the purpose of the attention module, in this application, is to assist future modules in representing Net Structure and Background as a binary image[2]. Binarising schemes will be thoroughly discussed, but the favoured scheme inevitably misrepresents the Net if it appears darker than the Background in a given scene. This is typical for cleaning operations where there is backlight, for instance from the surface waters. Whereas the U-Net can easily be trained to recognise areas of dark or bright net interchangeably, it is not necessarily a trivial task for traditional methods to effectively tell the difference.

Instead of relying on a two-class attention (NeNoS) module, two ideas arose; either the neural network could be set to provide a **three-class output**, namely Bright Net, Dark Net, and Nonsense, or, could it segment the net threads directly?

4.3 3CAS: Three-Class Attention Segmentation

By creating a U-Net capable of separating Bright Net from Dark Net, and Nonsense from either type of Net, one tackles well the problem posed from the NeNoS module. A few arguments also exist why this method *could* be preferable to direct net thread segmentation:

1. The 3CAS U-Nets can, perhaps, work on significantly smaller input images without losing too much information. This because net thread detection intuitively requires higher resolution than a rough sketch.
2. Traditional methods, such as adaptive thresholding [41], execute on large images within milliseconds on modern GPUs. The small attention mask can be applied to the original size image and the binarising algorithm can effectively work with full resolution.
3. Traditional approaches to binarisation are more transparent than neural networks. Whereas one cannot be 100% confident that the neural network represents net threads fairly, traditional methods will deliver predictable results on areas pointed to by the neural network.

However, one should be quite aware that the cost of such implementation is a significant increase in complexity. Net thread segmentation effectively combines attention and binarisation, whilst this method combines (in this application) a U-Net for attention segmentation, adaptive thresholding for binarisation, and adaptive morphological operations to decide parameters for the adaptive thresholding algorithm (chapter 5). It is, nevertheless, considered a useful experiment, and it is believed that aspects to this upcoming approach, including the temporal memory inclusion (yet to be discussed), can be useful also for other applications.

4.3.1 Class Encoding

The MultiRes U-Net was slightly modified to fit this application, introducing additional filters in the final layer to obtain 3-channel outputs. Thus, instead of

[2]This work capitalises hereafter words that represent *classes* handled by machine learning models.

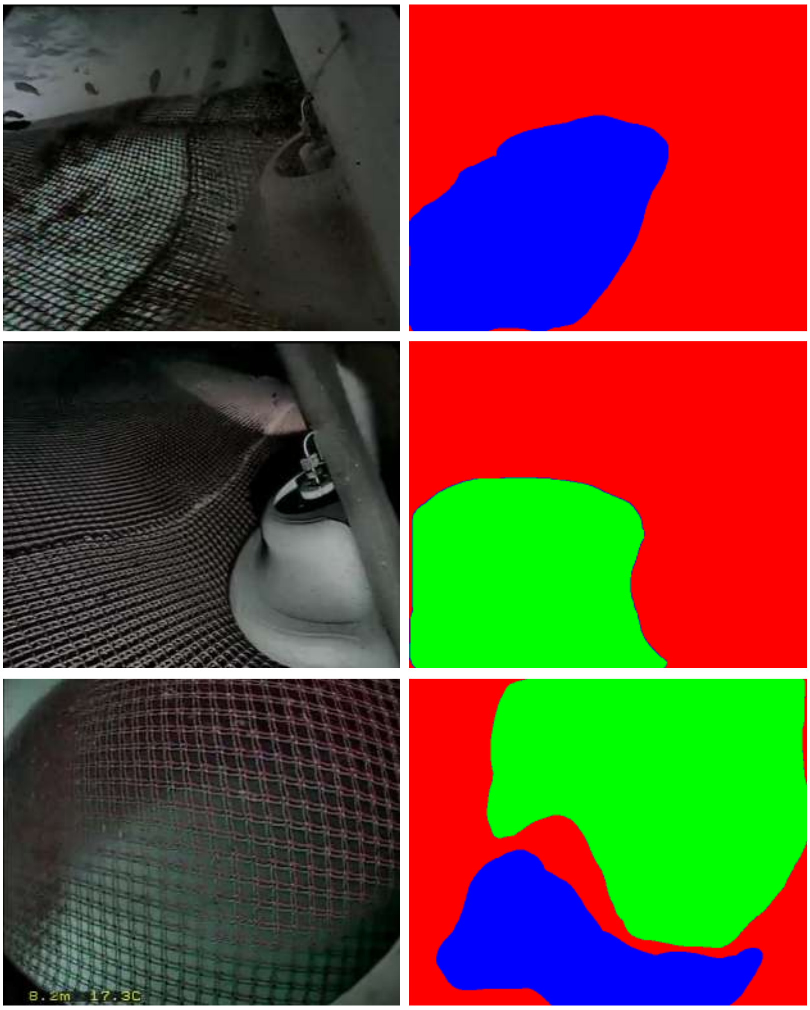


Figure 4.5: Approximately 250 images were constructed to train the 3CAS U-Net models. Each category was represented with an RGB channel; red areas indicate Nonsense, green Bright Net, and blue Dark Net.

binary attention masks, each pixel of the ground truth mask keeps three intensity values; a pixel belonging to a Nonsense area was encoded as [1,0,0] (all-red in fig. 4.5), Bright Net pixels as [0,1,0] (all-green), and Dark Net pixels as [0,0,1] (all-blue).

One could ask why three dimensions need be introduced instead of keeping one dimension where, say, Nonsense is encoded as -1, Bright Net as 0, and Dark Net as 1. Conventionally, such encoding is to be avoided when working on classification tasks (and this is indeed a classification task – where we classify each pixel of an image). This implementation is implying that Bright Net is somehow *closer* to Nonsense than Dark Net is to Nonsense since 0 is closer to -1 than 1 is to -1. By introducing three dimensions, one for each class, the three classes can be treated completely independently when training the neural network.

4.3.2 Class Imbalance

The production of 250 segmented images (from MANTA2 and MANTA3, see chapter 3) resulted in a dataset with significant class imbalance. Even though the (rather scarce) Dark Net class was disproportionately sought after during data acquisition, the total amount of pixels belonging to the Nonsense class accounted for approximately 70% of the data foundation, Dark Net pixels about 10%, and Bright Net pixels the remaining 20%.

Such class imbalance typically manifests itself in semantic segmentation. After all, most scenes in net cage cleaning operations are dominated by Nonsense, and, resultingly, models trained on raw data will have a bias towards the most dominant class. This could be beneficial if one wants to achieve a somewhat conservative model, but measures are often made to even the class distribution, or to elsewise emphasise scarce classes more by granting them higher importance in the loss function (see for instance [54]).

Solutions to the class imbalance present in this exploratory work has not been further investigated, and it remains unknown whether or not it is preferable to produce *less* conservative models. It may very well be the case that Nonsense is a fair *expected class*. However, to ensure a fair comparison of the K models trained (in a K-Fold manner), a greedy algorithm was developed to encourage class stratification (the algorithm is explained in fig. 4.6, and the class distributions shown in fig. 4.7).

In total nine 3CAS-models were trained, three on each of the input image sizes 64, 128, and 156, with datasets provided by the stratified K-Fold algorithm. Judging from fig. 4.8, it seems that models trained on different sizes perform almost equally well, although some improvements seem to accompany greater input sizes.

Class 0	Class 1	Class 2
Image 1	Image 2	Image 3
Image 4	Image 5	Image 6
Image 7	Image 8	Image 9
...	...	...

Highest Prevalence

Lowest Prevalence

First Iteration

Fold 0	Fold 1	Fold 2
Image 1	Image 4	Image 7

Second Iteration

Fold 0	Fold 1	Fold 2
Image 1	Image 4	Image 7
Image 8	Image 2	Image 5

Third Iteration

Fold 0	Fold 1	Fold 2
Image 1	Image 4	Image 7
Image 8	Image 2	Image 5
Image 6	Image 9	Image 3

Figure 4.6: The stratified K-Fold algorithm seeks to balance the class contents of each fold, such that all folds have a comparable data foundation. This specific version of the algorithm sorts first all images based on their total number of pixels belonging to each class. Here, Image 1 has the highest number of Class 0 pixels, and Image 4 the second highest number of Class 0 pixels. If K=3, then, in the first iteration, is Fold 0 dealt the Image with the highest prevalence of Class 0, Fold 1 the image with the second highest prevalence of Class 0, and Fold 2 the image with the third highest prevalence of Class 1. The next iteration, Fold 1 is dealt the image with the highest prevalence of Class 1, Fold 2 the image with the second highest prevalence of Class 1, and Fold 0 the image with the third highest prevalence of Class 2. So it goes – in a round-robin-manner – until all images have been assigned to a fold.

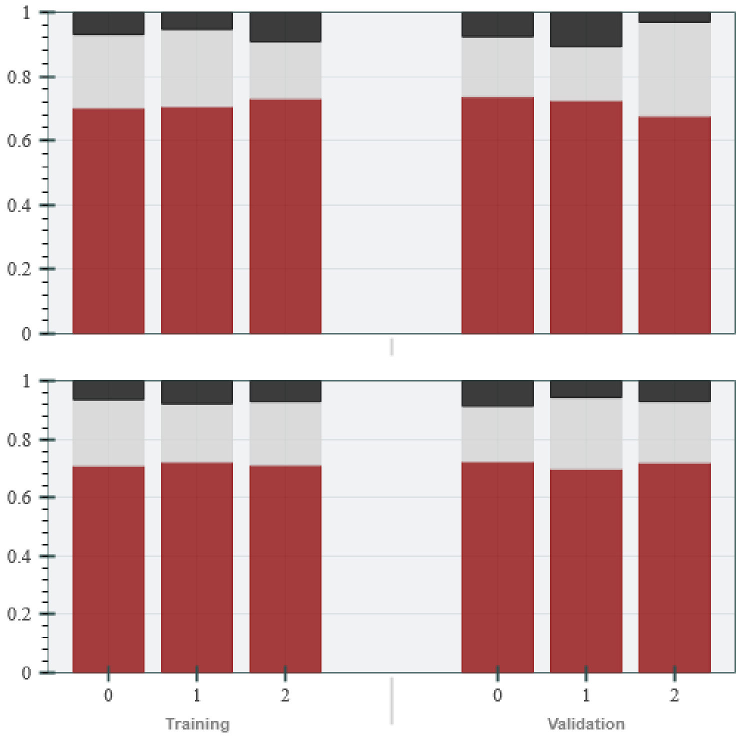


Figure 4.7: The class distribution for each fold can be evened with the greedy stratification algorithm proposed in fig. 4.6. The upper figure shows the folds' class distributions for training and validation, respectively, with a regular randomised K-Fold algorithm. The lower figure shows the resulting folds with stratification. Red colour is Nonsense, white is Bright Net, and black is Dark Net. Notice that, for instance, fold 2 is exposed to very little Dark Net in its randomised validation dataset, whilst exposed to more Dark Net than any other during training. With stratification, the proportion of each class is relatively even in both training and validation.

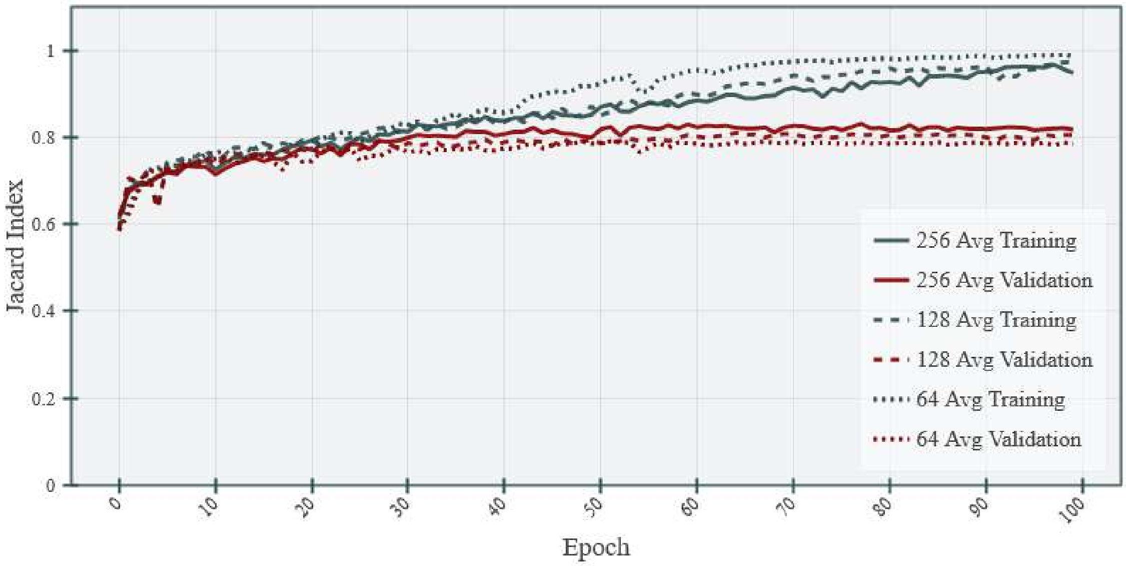


Figure 4.8: The average scores from 3-Fold 3CAS-training do not vary much depending on what input size the models have been training on. Seemingly, the models trained on 64 x 64 pixel images manage to grasp the complexity of the training data entirely, whilst abstracting that knowledge slightly worse than the other models to unseen data. Since the complexity of the models' hidden layers have not been modified, it makes sense that the models working on smaller images are relatively more complex given the task at hand than what is the case for the models working on larger images. More complex models typically yield better training scores, whilst worse validation scores could indicate that the compressed images are harder to interpret.

4.3.3 Input Image Size

Watching a few case studies from the test dataset (fig. 4.9) reveals, surprisingly, that 128 x 128 pixels could be a preferred input size to the larger 256 x 256 pixels, and the smaller 64 x 64 pixel images. It could be the case that, with very fine resolution, the models fail to include irregularities such as rather large holes in any Net class, but rather classify those areas as Nonsense. Models working on smaller images, however, generally provide segmentations of higher compacity than those working on large models (see for instance in fig. 4.9 the thin left wing of the green area in *256 px A*, or the thin ridge connecting the green segmentation in *256 px B*). This is indeed a favourable trait, since misclassification of holes as Nonsense is to be avoided at almost any cost. On the flip side, compressing images *too* much seems to provoke false segmentation in both examples involving 64 x 64 pixel inputs. Hence, a good case can be made that 128 x 128 pixel models should be further utilised.

4.3.4 Encouraging Temporal Continuity

One concern with frame-by-frame segmentation is the blindness to temporal continuity. Assumably, segmentation masks should not vary too much from one frame to the other – given that we are indeed analysing a video captured at 30 fps with an ROV moving approximately 1 m/s. One could, for instance, fear that an occurring irregularity within Net Structure is misclassified as Nonsense if the U-Net is only allowed to peek at that single frame, but if it were allowed to look *back* in time, it would be more inclined to include it in whichever Net class surrounds that irregularity.

The MultiRes U-Net authors have already created a three-dimensional counterpart to their U-Net[3]. By *stacking* a series of frames, creating a *time stack* of frames, one can apply convolutional operations between these frames by shifting filters across a third dimension, thus capturing the *motion feature space* of the images in the stack. This approach was initially implemented for *48, but at least 16 images per stack*, which eventually made the time complexity of the operation way too large to be feasibly implemented in a real-time application. Several seconds were required to perform a single segmentation. It could be thought that, since the initial application to which this architecture was designed, was biomedical-imaging, that it is better suited MRI (making up a 3D-scan) rather than fish cage net images stacked in time.

From available literature on spatiotemporal research, temporal consistency is usually accompanied by a considerable increase in time complexity. For instance, one interesting recent contribution to the field suggested pairing two CNN architectures; one for tracking the spatial coherence and one for the temporal [55]. The cost was 3.90 seconds per frame. Similar models include the *SegFlow* which utilises optical flow in parallel with CNN networks [56]. The execution

[3] https://github.com/nibtehaz/MultiResUNet/blob/master/MultiResUNet3D.py

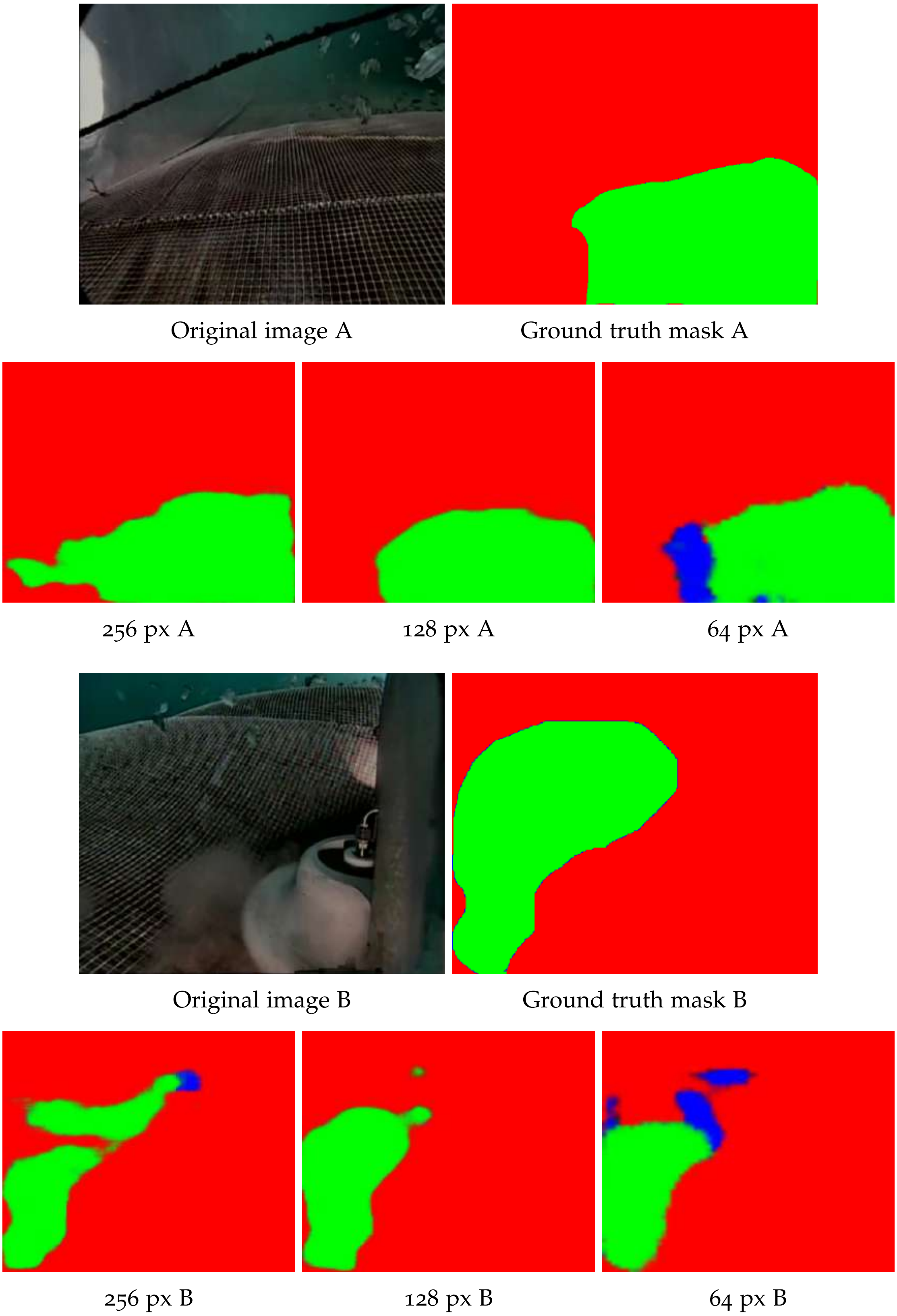


Figure 4.9: 3CAS models working on 256 x 256 pixel inputs, 128 x 128 pixel inputs, and 64 x 64 pixel inputs provide some different takes on segmentation. Although overall scores are quite similar, there is an argument to be made that models working on large images undersegment slightly, whereas models working on very small images oversegment, identifying Dark Net where there is none and vice versa.

time is 7.9 seconds per frame, working on the NVIDIA Titan X GPU[4]. All things considered, these operations are too heavy to be currently considered for real-time applications.

The Lag Mask: Expanding the Input Layer

A new and more cost-efficient solution is proposed in this work to encapsulate spatiotemporal information. This method allows the U-Net to *peek* at the previous segmentation, *the lag mask*, by concatenating, pixel-wise, the lag mask with the current input image, thus expanding the channels of a pixel in the input image by three, from 3 to 6. This alteration to the architecture of the U-Net, namely expanding the size of the input layer with 3 channels, seems not to impact the execution time of a prediction with more than half a millisecond, running on a standard CPU, compared to a regular 128 x 128 pixels 3CAS model.

There are, arguably, two significant pitfalls to this method. For one, the camera may or may not be in motion. This means that the previous, ideal, segmentation may or may not overlap completely with the ideal segmentation for the current frame. The mapping, therefore, between the lag mask and the ideal segmentation for the current frame varies depending on current movement. Secondly, the model might discover, during training, that the lag mask is a *very* good indicator of the current segmentation. However, in production, the model will necessarily make a few sloppy segmentations. If these are allowed to propagate to the next prediction, and the model assumes the lag mask *always* to be reliable, then one can quickly get stuck in a deadlock situation where every new segmentation replicates the previous, converging typically towards one class dominating the entire scene.

Stirring It Up: Countering Movement Ambiguity

A proposed method to counter the ambiguity caused by present (or not) camera movement, is to apply some significant blur to the lag mask before concatenation with the current frame. The severeness of this blur justifies also to what degree the U-Net can blindly replicate the lag mask, but it might still extract from it hints about what classes are present in what parts of the frame. Intuitively, the blurry borders compensate for minor movement that might have occurred from the previous frame to the next. An example of this method is visualised in fig. 4.10.

Deliberate Corruption: Circumventing Blindness to the World

Children, and robust machine learning models, whilst learning what patterns govern the world, both arguably benefit from exposure to a certain fraction of corruption. The story of Siddhartha Gautama – the Buddha – is one of a sheltered young man whose perception of reality crumbles when witnessing poverty, illness, and old age beyond the protective walls of his father's palace [57]. Likewise, U-Nets always provided with rock-solid lag masks during training

[4] https://www.nvidia.com/en-us/geforce/products/10series/titan-x-pascal/

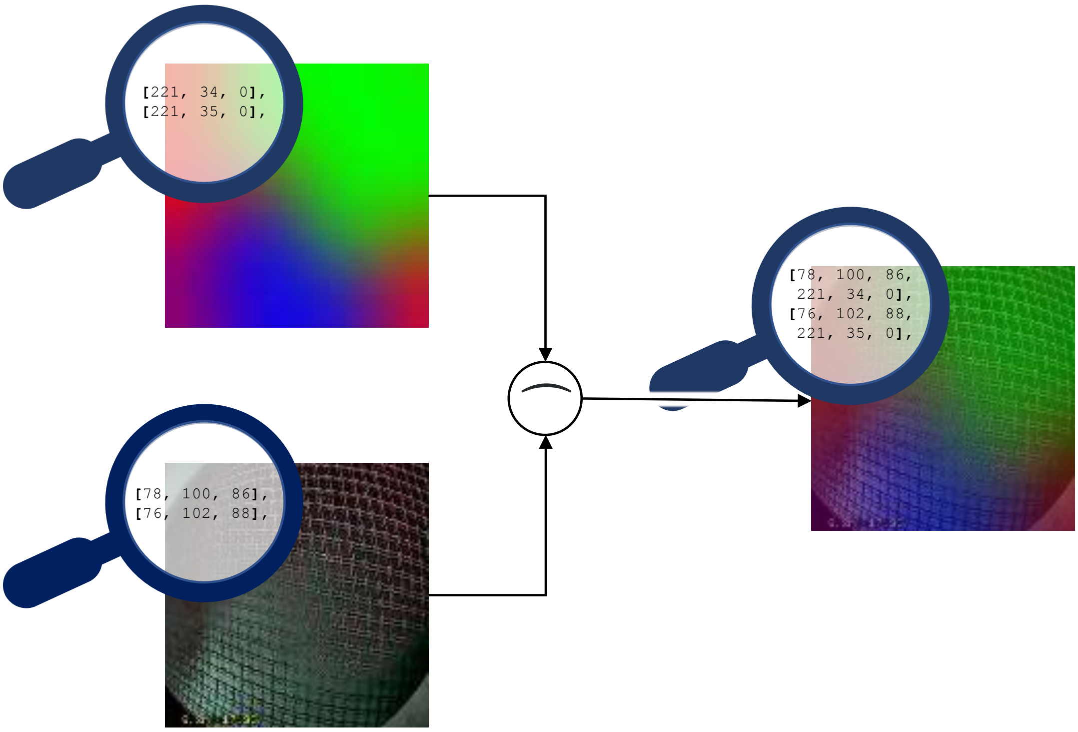


Figure 4.10: Pixel-wise concatenation of the lag mask (upper left) with the current frame (lower left) yields an output image with six channels, where each pixel encapsulates the RGB values of both the current frame and the lag mask. Notice that these RGB values are visualised on a [0, 255] interval, and not the normalised [0,1] interval which is utilised for all training.

can fail miserably when entering the real world where lag masks can be all but perfect. By injecting some corrupted masks during training, however, the models can learn to balance the trust put in both the current frame, *the world*, and the lag mask at hand.

In practice, a certain fraction of the training images were, for each epoch, corrupted by replacing the concatenated lag mask with a mono-class mask. That is, each pixel of the lag mask was replaced by either [0,0,1], [0,1,0], or [1,0,0]. The idea is that, for these training instances, the models *have* to rely entirely on the pixel intensities of the input image related to the current frame, and to disregard the pixel intensities related to the lag mask, since these are nothing but noise. This technique is a form of *regularisation*. The higher the fraction – the less trust is put in the lag mask, and vice versa.

4.3.5 Finding the Ultimate Combination

Having established that 128 x 128 pixels provides a good representation for the original 500 x 600 pixels images, it remains still unclear

1. if lag masks improve the segmentation, and, if so,
2. to what degree the lag mask should be blurred, and
3. if the training data should be regularised, and if so, to what degree.

Keeping in mind that the purpose of the lag mask is to *stabilise* the segmentation, temporally, it seems that at least one metric which could monitor success is the degree of fluctuation in the models' segmentations during a video sequence. However, the optimal degree of fluctuation is decisively neither **zero** nor **one hundred** percent; the segmentations should be flexible enough to respond to occurrences of alien objects, such as fish, and shifting scenes, but stable enough not to flicker unnecessarily when the scene is rather stationary. Additionally, the models should be able to pick up both Dark and Bright Net if both classes occur within the same frame.

All these aspects have been carefully studied within the scope of **four** 150 frames-long test videos, on which in total 24 individual U-Nets proposed their segmentations. Consult fig. 4.11 to properly understand the structure of the score-sheets presented in the upcoming evaluations.

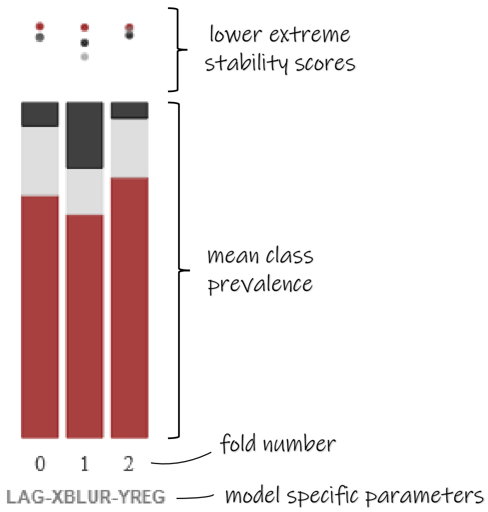


Figure 4.11: The scoreboards visualise the intra-class and inter-class dynamics of the different U-Net models, executed on a single video.

The name, LAG-XBLUR-YREG, reveals three parameters, dash-separated: the first whether or not the model is allowed to peek at the lag mask. This section can take on two values; L indicates that it *does* peek (lag) and NL that it does not (no lag). Subsequently, XBLUR tells the degree of blur applied to the lag mask. E.g., 25B means the breadth of the Gaussian kernel used to blur is 25% the breadth of the lag mask. The YREG section explains to what degree regularisation has been applied when training the model. Eg., 50R means 50% of lag masks have been swapped for noise during training.

The *fold numbers* reflect that for each configuration were trained three models in a K-Fold manner. When comparing the different configurations, it can be useful to compare fold numbers 0 to 0, 1 to 1, and 2 to 2 since these models have utilised the same training data, thus leaving most discrepancy to configuration differences.

The barplot shows the mean proportion of that video classified as Nonsense (red), Bright Net (white), and Dark Net (black).

The scatter plot displays the intra-class dynamics for each class for that video. For each class is measured the Jaccard index between the current segmentation and the previous, where a low score indicates little overlap, and therefore a high degree of frame-by-frame fluctuation. The dots mark the mean Jaccard score minus **three** standard deviations, meaning scores near 1 indicate extreme stability certainly typical of deadlocking segmentations.

Above the figures are also QR-codes leading to a video showing the segmentations performed the models. The segmentations have been coupled with a binarising scheme for clarity (see chapter 5).

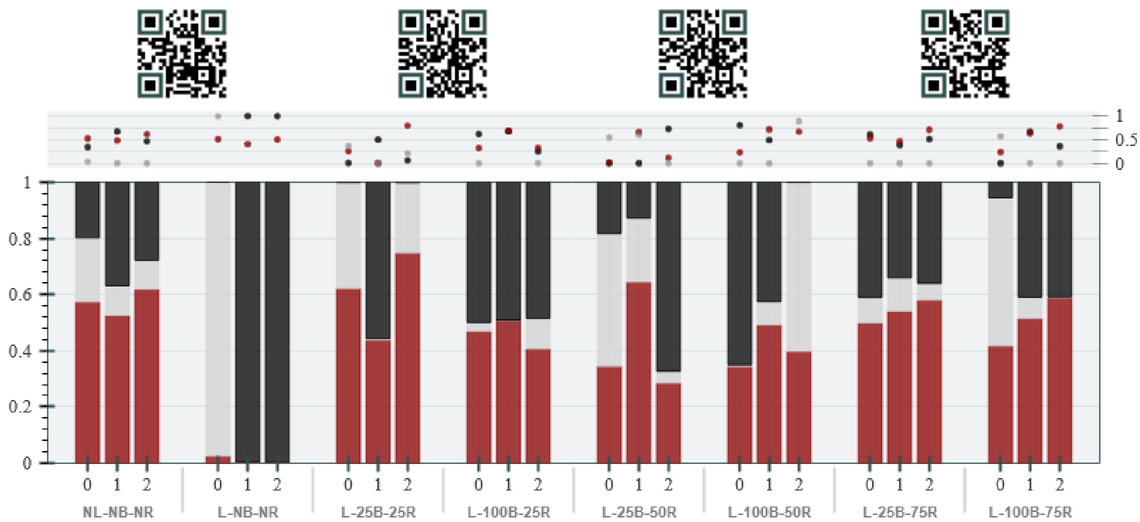


Figure 4.12: Test video 1 segmentation scores. Remember that hyperlink-equivalents to QR-codes are available in appendix A.

Segmentation Test Video 1: A Dynamic Scene

The first test video is arguably a challenging one; the first second we observe an equal amount of each class – then the camera moves, introducing severe motion blur which should result in predominantly Nonsense segmentations. Then, for approximately two seconds, we observe, yet again, all three classes. This rules out, obviously, each fold of L-NB-NR, each fold of L-25B-25R, fold 1 (and most likely 0) of L-100B-25R, and folds 0 and 2 of L-100B-50R (see fig. 4.12) since they fail to include all three classes.

Judging from visual impressions (videos are available with QR-code scanners), NL-NB-NR provides the most convincing result in fold 1, where the segmentations are mostly good, albeit more fluctuating than desirable. Amongst the six models with 50% regularisation does L-100-50R fold 1 stand out. Generally, the models with severe blur seem to provide more stable segmentations during jerky camera movement.

Even though stability scores (in the scatter plot of fig. 4.12) seem not to improve from the lag-less model (NL-NB-NR), a striking edge to the lagful models is that they build *clusters of classes*. Whereas models with no lag could provide segmentations that overlapped to the same degree, the shape of those segmentations could be significantly different from the previous (often identifying a second class in the midst of another), but the models *with lag* shrank or grew the clusters at contour level, preserving the *core* of the cluster.

With 75% regularisation, the models seem to undersegment too much, leaving out perfectly intelligible parts of the scene. Additionally, the segmentations were perceived as better with 50% regularisation.

The best-performing model was, comfortably, fold 1 of L-100B-50R.

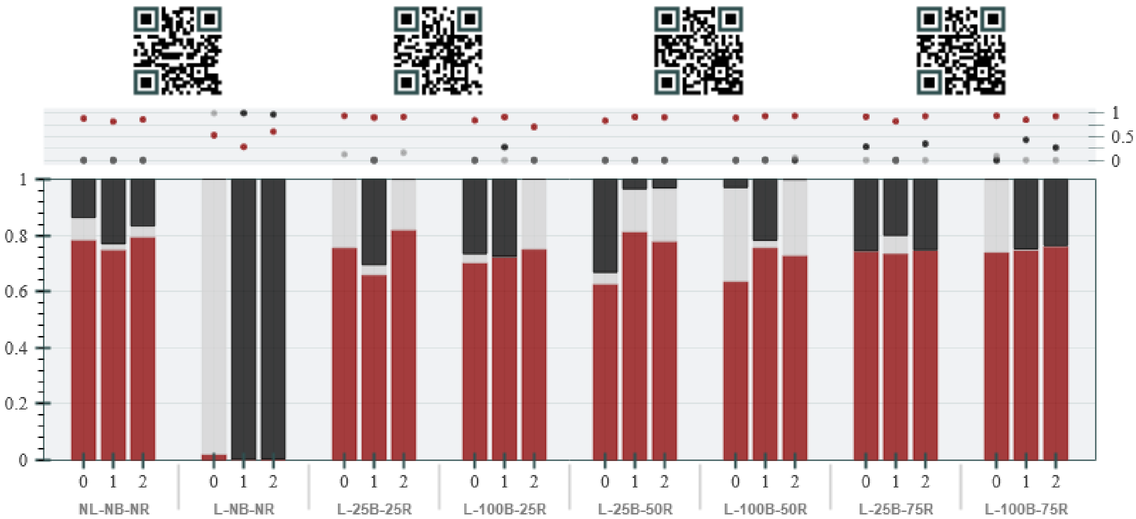


Figure 4.13: Test video 2 segmentation scores.

Segmentation Test Video 2: Mostly Nonsense and Dark Net

The second test video looks first at Dark Net, then surface waters, before returning to a scene with mostly Nonsense, Dark Net and a fraction Bright Net. All folds of L-NB-NR are therefore ruled out, and so are folds 0 and 2 of L-25B-25R, fold 2 of L-100B-25R, and most likely fold 1 and 2 of L-25B-50R, fold 0 and 2 of L-100B-50R, and fold 0 of L-100B-75R, due to little Dark Net content (fig. 4.13).

From visual impression, fold 1 of NL-NB-NR does a decent job, managing to pick up the Bright Net fraction. Introducing lag, with 25% regularisation, effectively removes tendencies to sporadic oversegmentation when analysing surface waters. The reluctance of these models to establish segmentation clusters where there yet are none seems to be a benefit to the lag-mask-paradigm. However, neither model manages to pick up both Dark and Bright Net when both are present, indicating that forming a break-out cluster might be *too* hard.

With 50% regularisation, four out of six models present inverted binary representations (also very clear from fig. 4.13) and the only two potential candidates are therefore fold 0 of L-25B-50R and fold 1 of L-100B-50R, the latter of which clearly outperforms the former. However, neither of these models manage to identify the tiny area of Bright Net, but L-100B-50R disregards more of that as Nonsense than the models with 25% regularisation, therefore providing a scene less prone to false positive hole detection.

Neither did the models with 75% regularisation manage to identify the tiny piece of Bright Net. The segmentations are, overall, decent, but more unstable than those with 50% regularisation.

The best model is, yet again, fold 1 of L-100B-50R. Even if failing to pick up the small piece of Bright Net, it provided a stable and intelligible segmentation of the other classes.

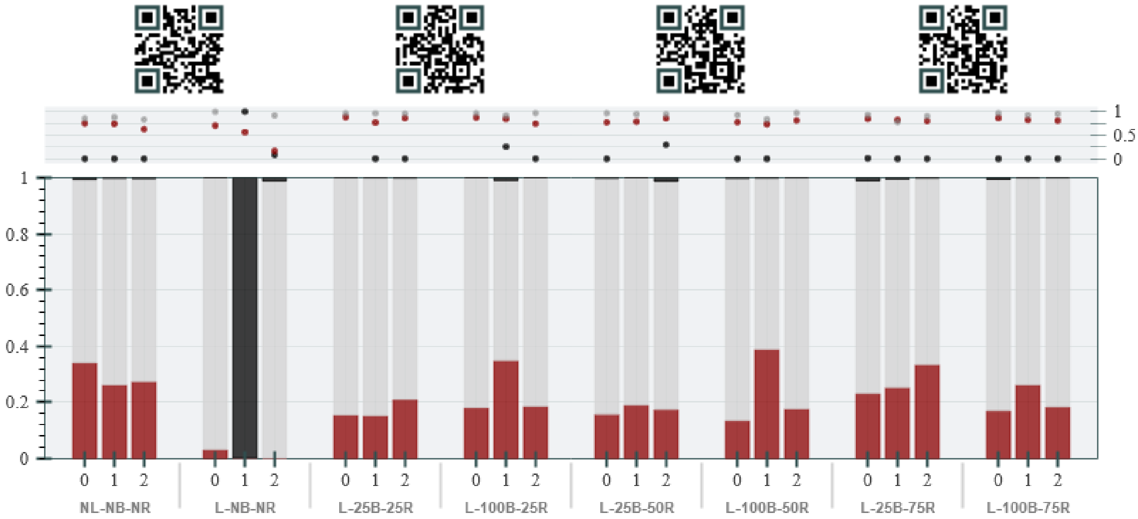


Figure 4.14: Test video 3 segmentation scores.

Segmentation Test Video 3: Stable Bright Net

The third test video shows a steady scene mostly consisting of Bright Net. This reflects well in most models of fig. 4.14, where there is typically a lot of Bright Net with very high stability scores. The only set of models to fail, miserably, are those of L-NB-NR.

The lag-less models provide decent segmentations, but with more fluctuation in the contours than the stable scene should imply. There are also some instances of false Dark Net reports. With lag and 25% regularisation, the contours stabilise significantly for all models. However, this video shows some interesting lighting conditions on parts of the net which are disregarded by fold 1 of L-100B-25R (and, to some degree, by fold 2 of L-25B-25R) as Nonsense. This could benefit the hole detection algorithms as the binary representation of this area elsewise could resemble holes. With 50% regularisation, all models perform quite similarly, but fold 1 of L-100B-50R is more conservative in its approach (which manifests itself in larger Nonsense proportions in fig. 4.14). Arguably, this model chooses to segment only net that is clear enough to be intelligibly analysed, whereas other models include net that is too far away for reliable analysis. With 75% regularisation, all models perform decently, albeit with some more contour flickering.

Most models performed well on test video 3. However, models with lag masks outperform lag-less models with their stable contours.

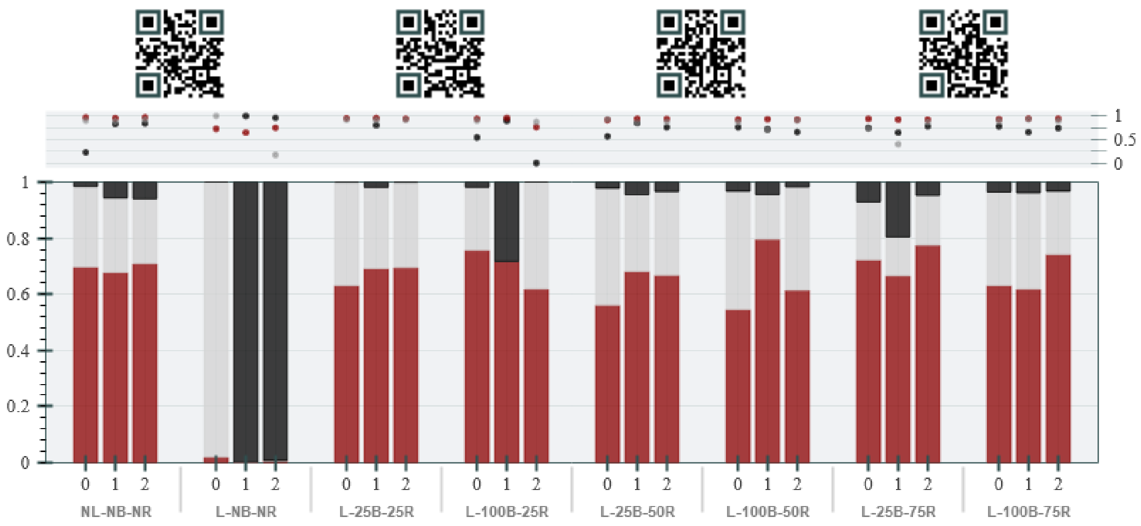


Figure 4.15: Test video 4 segmentation scores.

Segmentation Test Video 4: 3 Stable Classes

The last segmentation test video is almost as stable as a still-video; containing all three classes, but predominantly Nonsense and Bright Net. This means that all folds of L-NB-NR, fold 0 and 2 of L-25B-25R, and fold 1 and 2 of L-100B-25R can be immediately ruled out from observing in fig. 4.15 that they lack either one of two classes.

Amongst lag-less models are folds 1 and 2 of NL-NB-NR actually very good, judging from visual analysis of the videos. The contours do indeed flicker some, but the border between Bright and Dark Net is very precise. The two models not yet discarded amongst those of 25% regularisation both perform very well on this task. Fold 0 of L-100B-25R is slightly more conservative than fold 1 of L-25B-25R, but overall, both are stable and good. Strikingly, *all* models with 50% regularisation correctly identify the three classes. One peculiarity with fold 1 of both, however, is that they disregard the transitional area between the classes as Nonsense. This is actually quite clever, since this area is somewhat uninterpretable and tends to corrupt the segmentations of the other models. All models with 75% regularisation also manage to identify the three classes, but they introduce more noise, not suppressing well enough sporadic class clusters. There might be some advantage to L-25R-50R over L-100B-50R due to its higher degree of stability. This observation is confirmed by the scatter plot in fig. 4.15, showing some more stability for Bright and Dark Net in one than the other.

Specifically favourable segmentations of test video 4 are those proposed by fold 1 of either lagful model with 50% regularisation, with a tiny advantage to L-25B-50R.

Verdict

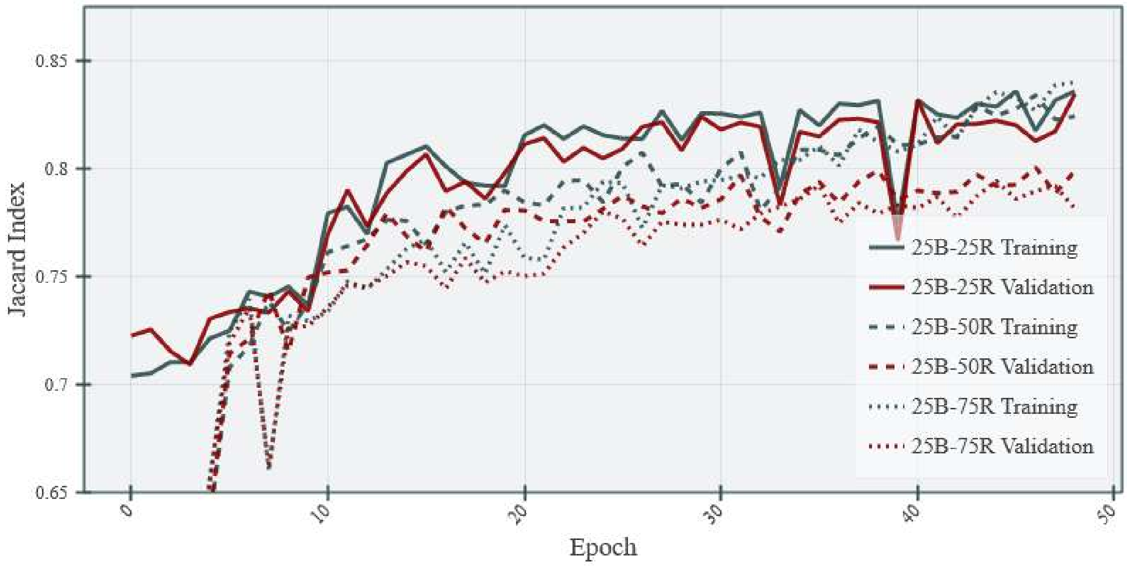
The test videos confirm that lag masks introduce temporal continuity – specifically by forcing classes to form in clusters which grow or shrink at contours, and new classes rarely occur within already formed clusters. Without such lag masks – the segmentation of one frame is completely decoupled from previous segmentations. However, the clusters formed tend to be over-conservative, and a proper way to counter this is through regularisation, incentivising confidence in the current frame and not only the previous mask.

Granted, more regularisation leads to worsening validation scores during training (see fig. 4.16) and indeed also more flickering contours. Nevertheless, it seems to be absolutely vital to achieve viable results in a realistic mileu. Be it 25% regularisation or 50% regularisation, in scenes with a single type of Net both performed well, but in complex scenes, 50% regularisation outperformed 25%, enabling break-away class clusters where needed.

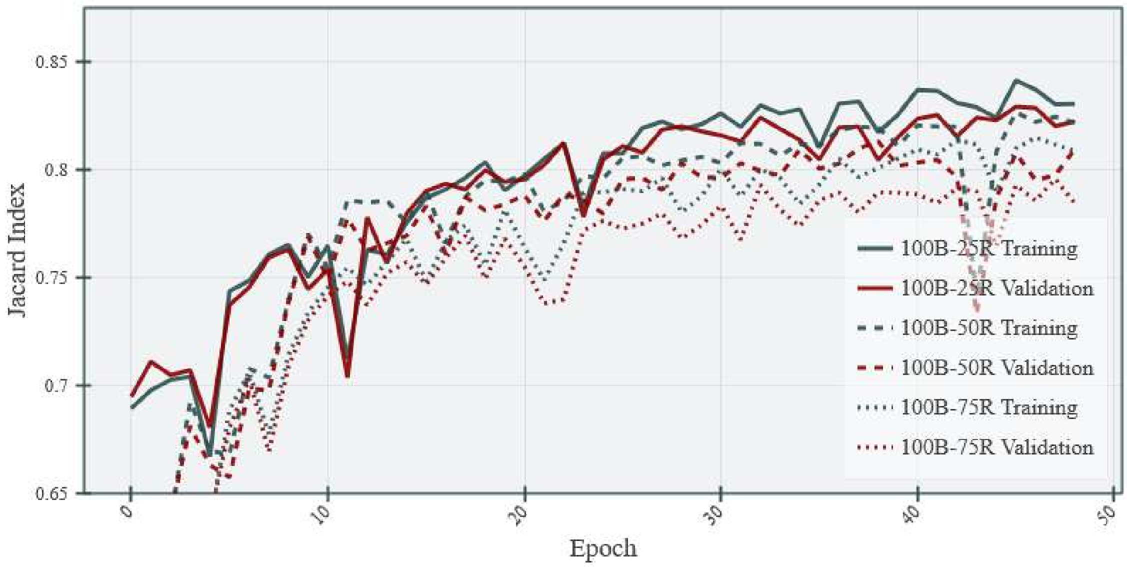
Severe or modest blur did not seem to impact validation scores to a large degree (see fig. 4.16). From watching the videos with little motion (test videos 3 and 4), deciding which level of blur outperformed the other was not really feasible, but, interestingly, when there *was* motion involved (test videos 1 and 2), the models with severe blur typically yielded more stable segmentation. This observation substantiates the claim that blurring assists the segmentation process during motion (which does make the lag mask a less precise predictor of the current).

Based on this investigation, fold 1 of L-100B-50R, a model working on 128 x 128 pixels input images – with lag masks – blurred with a Gaussian kernel of breadth 129, trained for 50 epochs where 50% of lag masks have been swapped for noise, is chosen as the best 3CAS model. It executes on a GPU on a typical image in less than 7 milliseconds, which makes it very suitable for real-time applications, depending on binarisation scheme effectiveness.

In production, the life cycle of predictions and lag mask propagation can be seen in fig. 4.17.



(a) Average scores during L-25B-YR training.



(b) Average scores during L-100B-YR training.

Figure 4.16: All models were trained for 50 epochs, after which training and validation scores diverged significantly. More regularisation worsens validation scores, whereas more blur (upper figure being modestly blurred and the lower, severely) has an ambiguous impact.

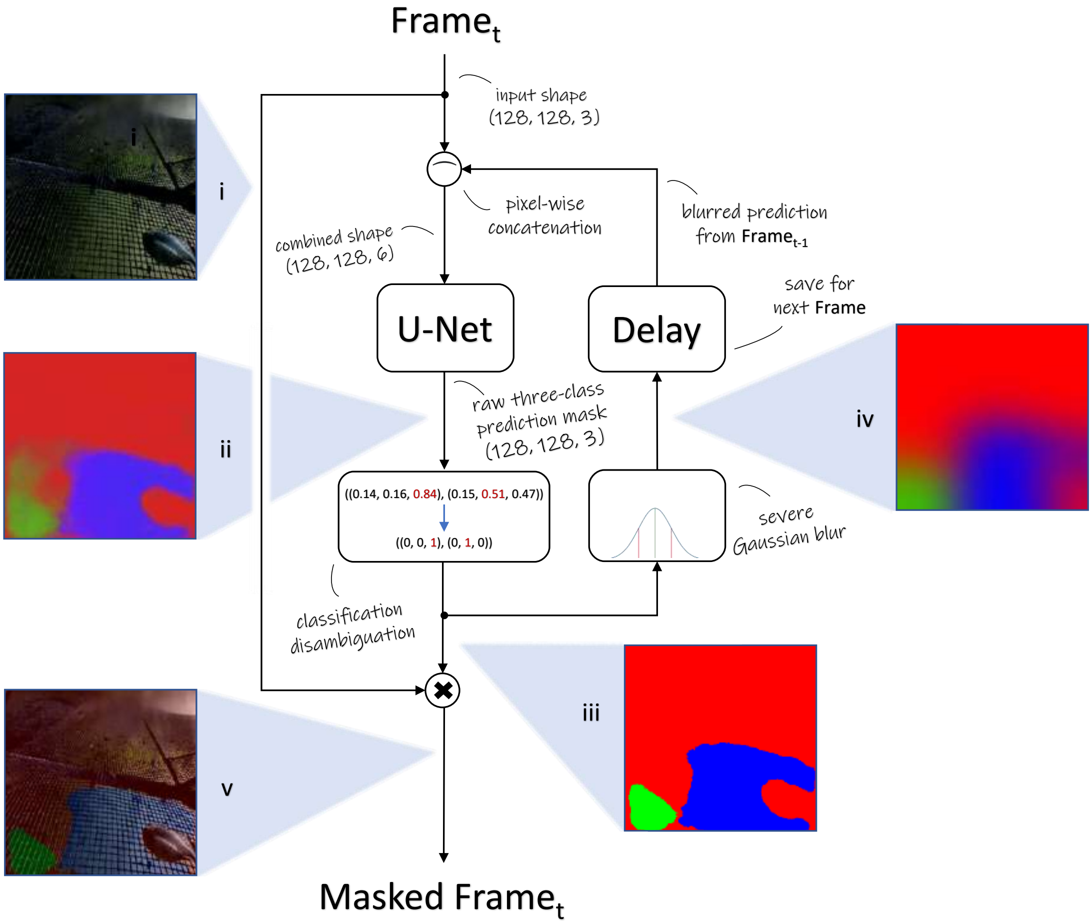


Figure 4.17: Whilst running in an unknown environment, the U-Net utilises the input image (i) in combination with the delayed lag mask (iv) to make the raw prediction (iii). The raw prediction is clarified by suppressing all pixel values but the channel with the maximum value, generating the final prediction mask (iii). This mask can be applied to the original image to create a masked frame (v) to assist future binarising schemes.

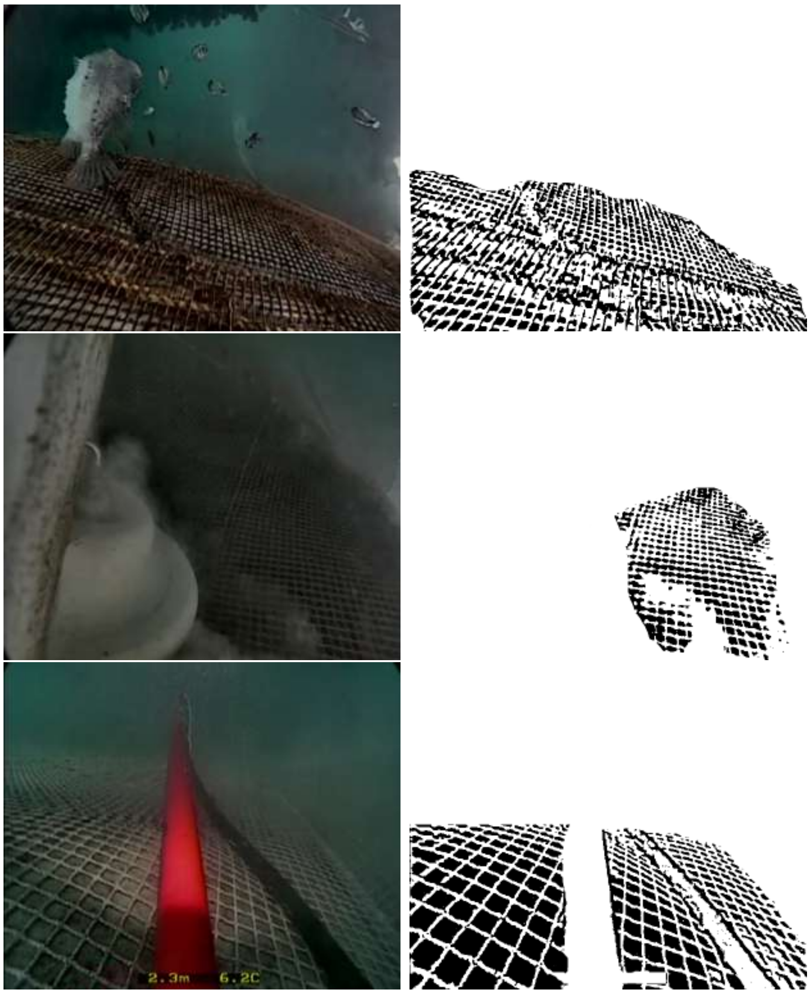


Figure 4.18: Net Thread Segmentation makes the subsequent binarising scheme redundant, if it can produce coherent binary image of satisfactory resolution. 200 training images were constructed from the very same video as was the 3CAS dataset.

4.4 NTS: Net Thread Segmentation

Seamlessly countering the problems of occasional Dark Net, occasional Bright Net, and higher or lower prevalence of Nonsense, can be achieved by training an NTS MultiRes U-Net. In principle, it utilises the same architecture as the NeNoS network, but with finer resolution output segmentations. Instead of having the U-Net propose a sketched area of attention, and implementing separate modules to properly binarise that area, the U-Net is suggested to itself deliver a coherent binarisation of the input.

For this experiment were constructed 200 images (see fig. 4.18) from the MANTA2 and MANTA3 videos. Yet again, all four views were utilised, equally.

One very specific concern might hinder the further development of this method. Whereas proposing a sketched area of attention can supposedly be achieved from very compressed images, segmenting *net threads* intuitively requires a higher resolution on the input images. On the other hand, sketched attention masks can be produced in small sizes and enlarged to fit the original frame without losing much information. In a sense, it requires *more* from NTS to succeed; it needs to produce a higher quality segmentation if the hole detector is to perform reliably.

4.4.1 K-Fold Training

Firstly, three models were trained in a K-Fold manner with resolutions 256 x 256 pixels, similarly to the NeNoS and 3CAS models. Afterwards, three models were trained, likewise, on 512 x 512 pixel images. Judging from fig. 4.19, the performances of the different models were not strikingly different. However, when enlarging the prediction masks and comparing them to ground truths of original size (500 x 600 pixels), the models trained on small images were able to reconstruct the original masks with 94% accuracy (in terms of Jaccard index), and the larger models hit 96%. These scores were produced by having each fold predict their training and validation data, then enlarging those predictions to fit the size of the original masks. Hence, they performed so well because they had already been exposed to the training data, but models working on smaller images managed not to represent this knowledge with the same accuracy as models working on larger images. The numbers might not seem too different, either, but considering the fact that the average mask is 88% white pixels, hitting large numbers is relatively easy, and a few percentages of score might be exactly what separates the models which manage to represent small broken net threads, and those that do not. Judging from case studies such as fig. 4.20, one can also notice how the models trained on larger image capture nuances which can be crucial when holes are present as broken meshes not directly in front of the camera.

4.4.2 Data Acquisition and Production Quality Concerns

Knowing that the 3CAS system can engage in a very powerful relationship with subsequent binarising schemes, it is absolutely crucial that NTS proves to match that performance. However, collecting NTS data is much more time consuming than collecting 3CAS data (notice the complexity difference between the masks in fig. 4.5 and fig. 4.18). It might be the case that NTS is vastly preferable to other strategies, but only if thousands of training images are generated.

Two strategies were tested to boost the potential of NTS;

1. Transfer learning[5] with base models trained on ROV images from FNC1, FNC2, and INSPECTION0.
2. Blurred and regularised lag masks in the same manner as 3CAS.

[5]Transfer learning implies specialising a pre-trained NN as opposed to training a NN with randomly initialised weights. It is further explained in chapter 6.

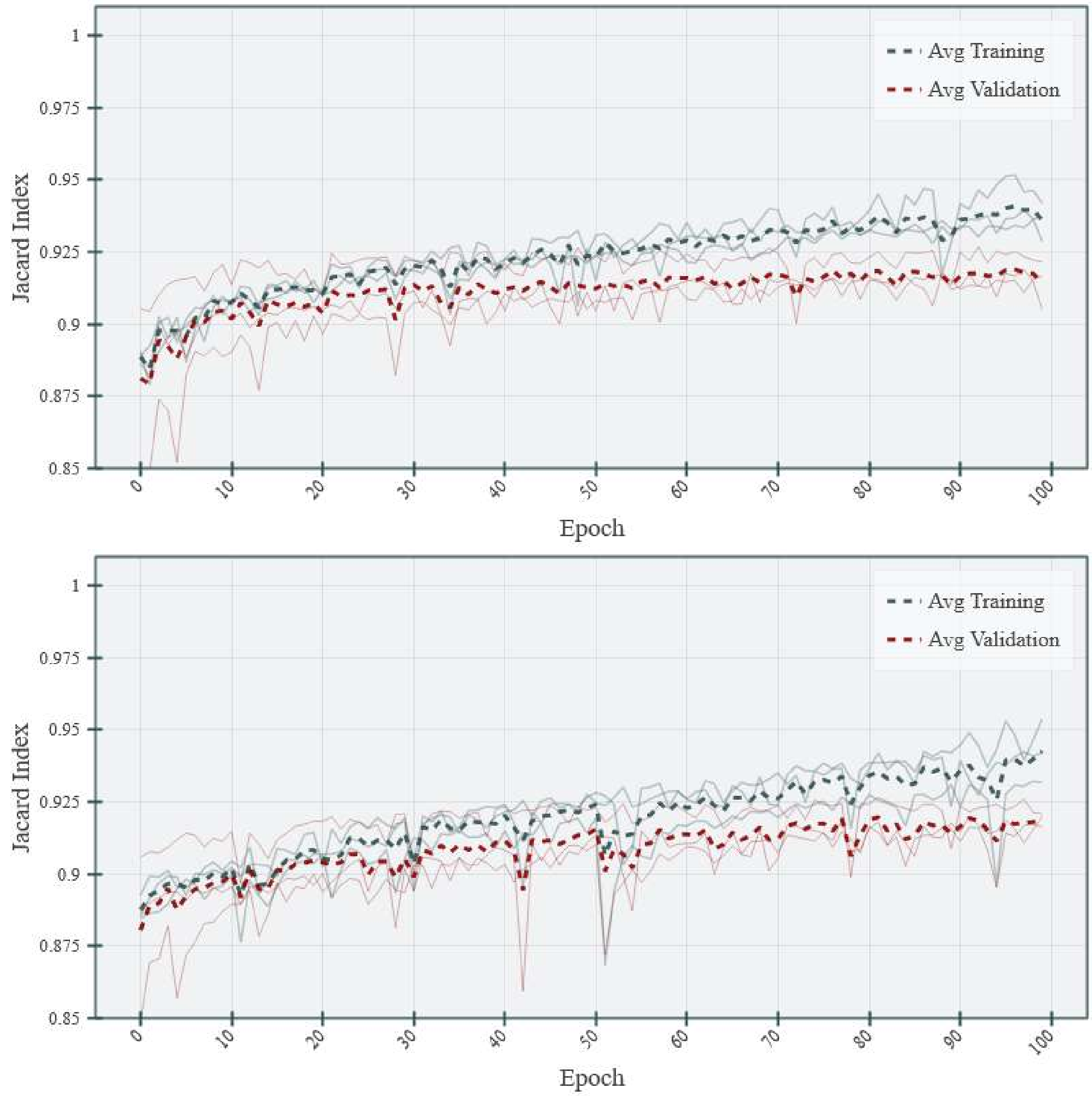


Figure 4.19: The two NTS models, trained on 256 x 256 pixel images (upper scoreboard) and on 512 x 512 pixels (lower scoreboard) performed similarly during training. Notice that, compared to fig. 4.3, fig. 4.18 have a larger proportion of white pixels. On average, 88% of the dataset masks contain white pixels, revealing why Jaccard scores start very high. Even though the models trained on 512 x 512 pixels seem to still be on the rise, indicating there is more to learn, the validation scores did not improve over the next 150 epochs following the 100 first.

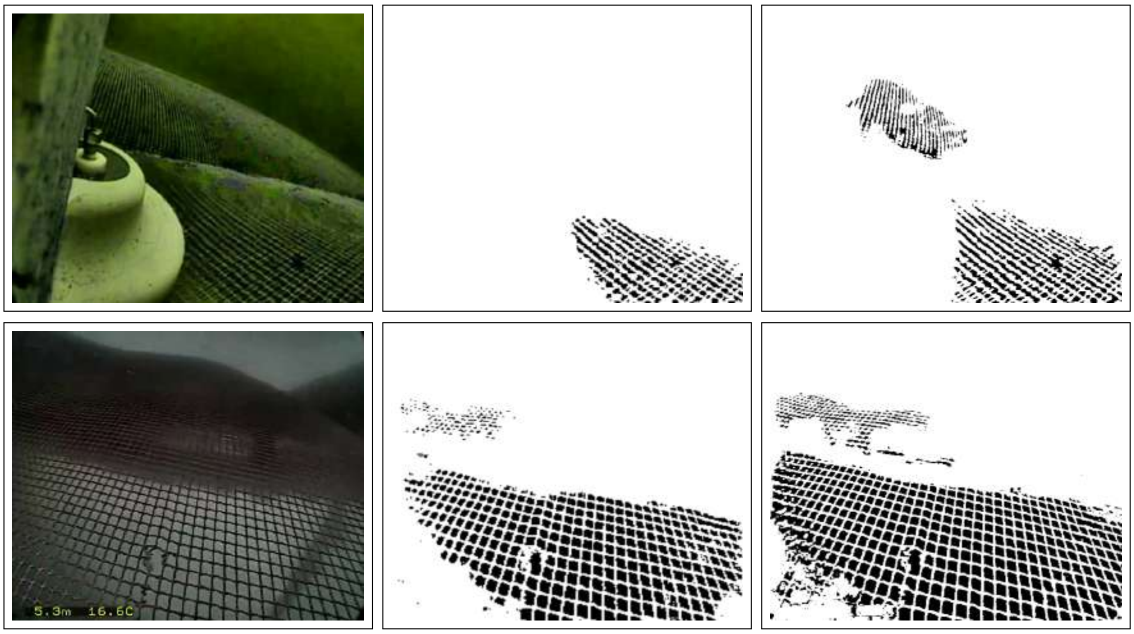


Figure 4.20: Leftmost images show the input, the the middle column a 256 x 256 pixel NTS model segmentation, and the rightmost column, a 512 x 512 pixel NTS model segmentation. The typical output from the models working on smaller input images represents the input well when meshes are large, but nuances are lost when the Net Structure is originally low resolution.

The main idea behind the transfer learning approach is to see whether or not intelligible knowledge can be transferred from one type of ROV to another. If possible, then the burden of creating training data for each ROV is lightened, since previously created models (for other ROVs) can be reused and fine-tuned on a limited number of training instances for the new ROV.

Introducing lag masks in the 3CAS system yielded favourable stabilisation in the segmentation. The acquired parameters for lag and regularisation will be reused, and, since the masks in this case are binary, we will only introduce one additional input channel instead of three. We will also assume that the lag mask is sufficiently similar to the current, and thus not engage in the labour of generating brand new lag masks, but simply blur and regularise the current.

4.4.3 Transferring Knowledge

Since multiple ROVs (with different cameras and views) can be used to perform a cleaning operation, a transfer learning approach could perhaps be a good way to make use of hard-earned data from one model by passing it on to another. Intuitively, all models have in common *the segmentation of pixels that look like Net Structure*, so there should be a fair chance that some information acquired when training on, say, FNC videos could be of interest to a model training on MANTA videos.

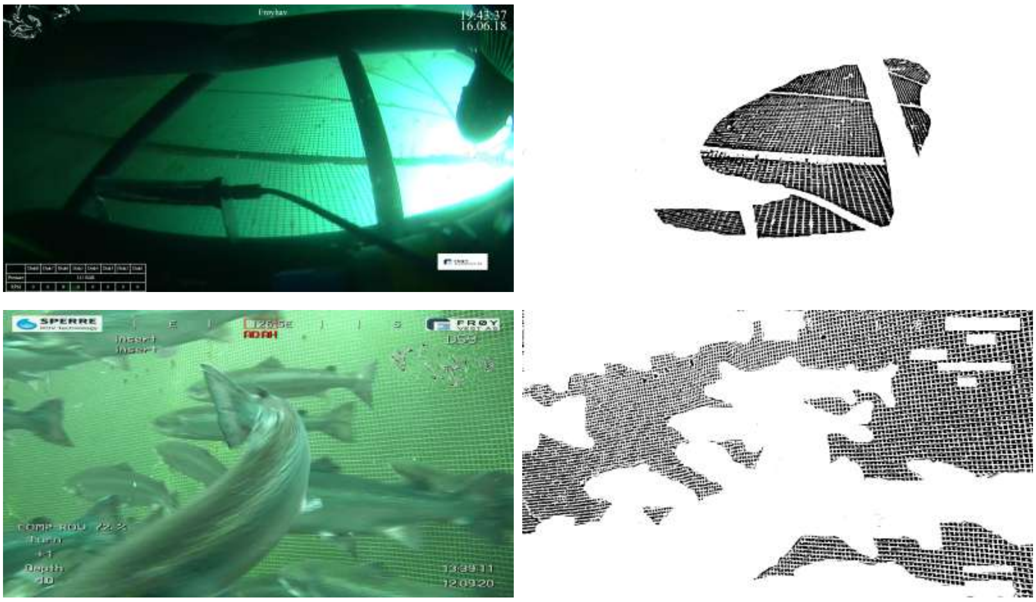


Figure 4.21: 190 additional images were made from FNC videos (upper row) and an INSPECTION video (lower row) to investigate if knowledge could be transferred from one set of data to another.

Practically, a total of 120 test images were generated from INSPECTION0, 30 from FNC1, and 40 from FNC2, in addition to the 200 MANTA images. Three folds of MultiRes U-Nets were trained on INSPECTION0 images for the first 100 epochs with default learning rate (0.01) like one would if one were to build a network specialised on INSPECTION videos. The learning rate was then lowered to 0.00001 and training- and validation data swapped to FNC1 for 25 epochs. Subsequently, training- and validation data changed to that of FNC2 for the next 25 epochs before fine-tuning with the MANTA data foundation between epochs 150 and 200.

Score-wise, judging from fig. 4.22, this approach did not add any edge to the original MANTA performance (see fig. 4.19). It could, however, be the case that the visual impression is better.

4.4.4 Finding the Ultimate Combination

Eventually, settling with 512 x 512 pixel input images, four model philosophies were investigated in a three-fold manner; NTS with no lag mask, NTS with blurred lag mask and regularisation, NTS with transfer learning and no lag mask, and NTS with transfer learning, blurred lag mask and regularisation. The best parameters for regularisation and blur from 3CAS were utilised, and the models were tested on the same four segmentations test videos as previously investigated with 3CAS.

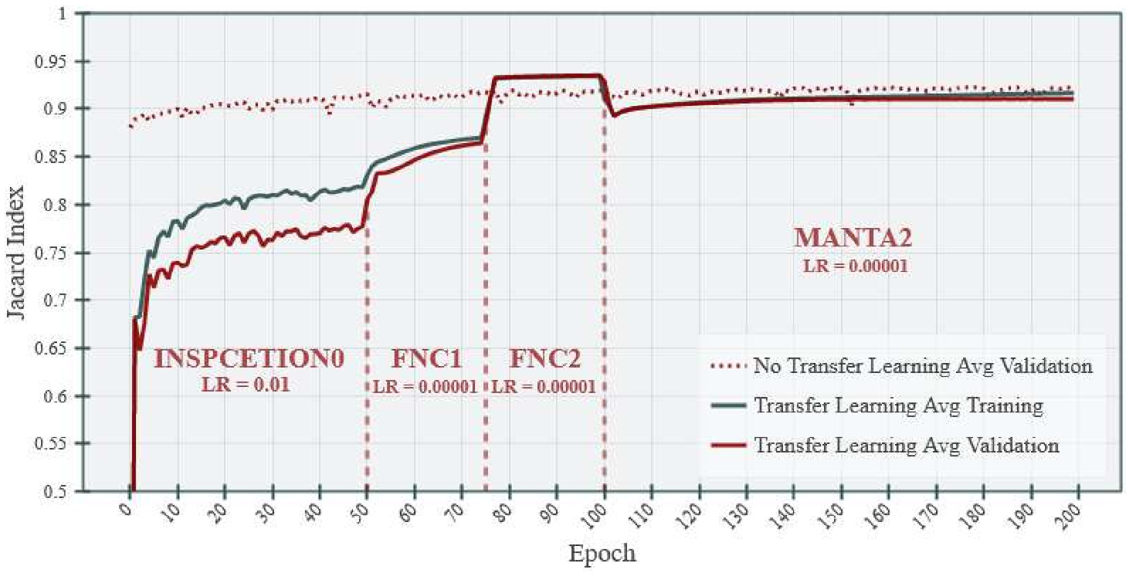


Figure 4.22: Three MultiRes U-Nets were trained in a K-Fold manner first on INSPECTION images, then, lowering learning rate, on two FNC datasets, before fine-tuning on MANTA images. The scores vary depending on the complexity of the datasets, so the sole purpose of this experiment was to see if the latter part of training, on MANTA2 images, could get the better of the previous models which trained on MANTA images only (dotted line). They seemingly did not.

Segmentation Test Video 1: A Dynamic Scene

This video is described in the 3CAS investigation (section 4.3.5), however, briefly, its main components are *movement* and *Bright and Dark Net*. The models with no transfer learning applied are initially uplifting; both fold 0 and fold 2 of NTS-NT-NB-NR represent the Bright and Dark Net structures surprisingly well. Both representations can be said to be favourable to the best-performing 3CAS-implementation, reacting instantaneously to the presence of both classes of Net. Especially fold 2 performs well, but it might be too eager to segment Background where there is in fact ambiguity (see from fig. 4.23 that the proportion of Black is larger for fold 2 than fold 0). All folds of NTS-T-NB-NR perform poorly; fold 0 segments way too much Background (and is also very unstable, reflected in its scatter plot), and folds 1 and 2 partly invert their segmentation, representing most of what is truly Background as Net or Nonsense (reflected in their large proportion of White class in the barplot).

With lag masks, fold 0 of NTS-NT-100B-50R stands out – clearly – segmenting the difficult transitions in net colour perfectly (see fig. 4.24). Its main challenge might be an overeagerness to segment ambiguous parts of the frame, thus risking false positive detections in blurry areas. Yet again, neither model with transfer learning manage to bring anything of interest to the table.

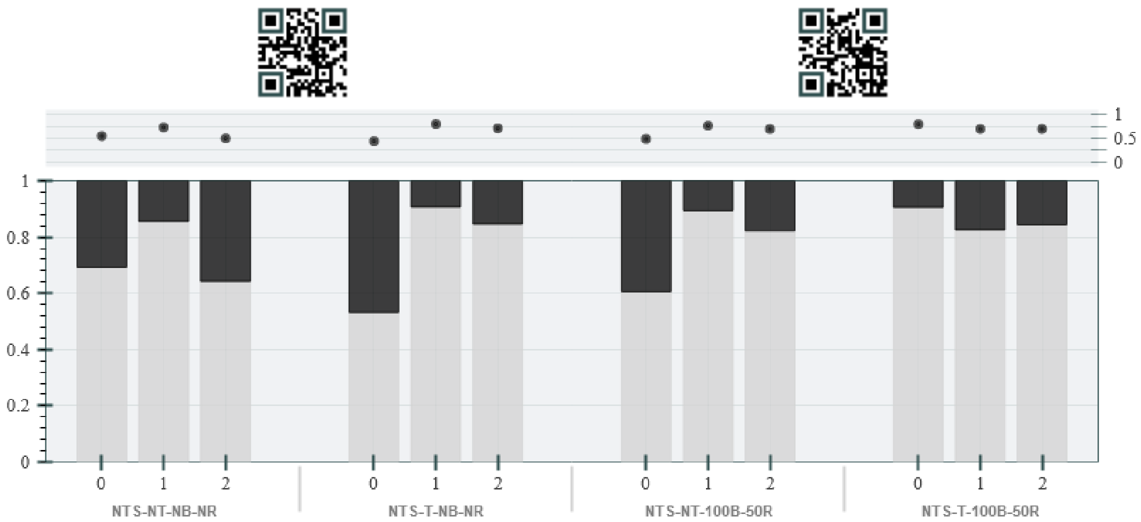


Figure 4.23: Test video 1 NTS segmentation scores. The segmentation is binary, including Dark Net, Bright Net and Nonsense in a single class (white bars). The black bars indicate the proportion of Background in the video (which is the area in-between net threads, and therefore potential holes). The scatter-plot shows the lower extreme stability score of the white class. T or NT translate to *transfer* or *no transfer*. Otherwise the plot is of similar nature to fig. 4.11.

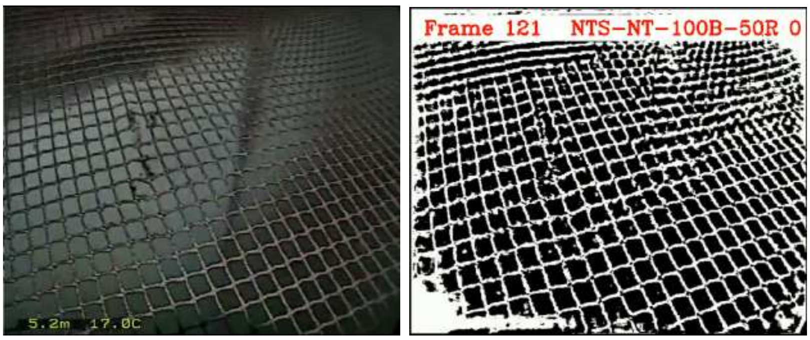


Figure 4.24: Fold 0 of NTS-NT-100B-50R segments the net structure in test video 1 almost perfectly, but segments also ambiguous areas.

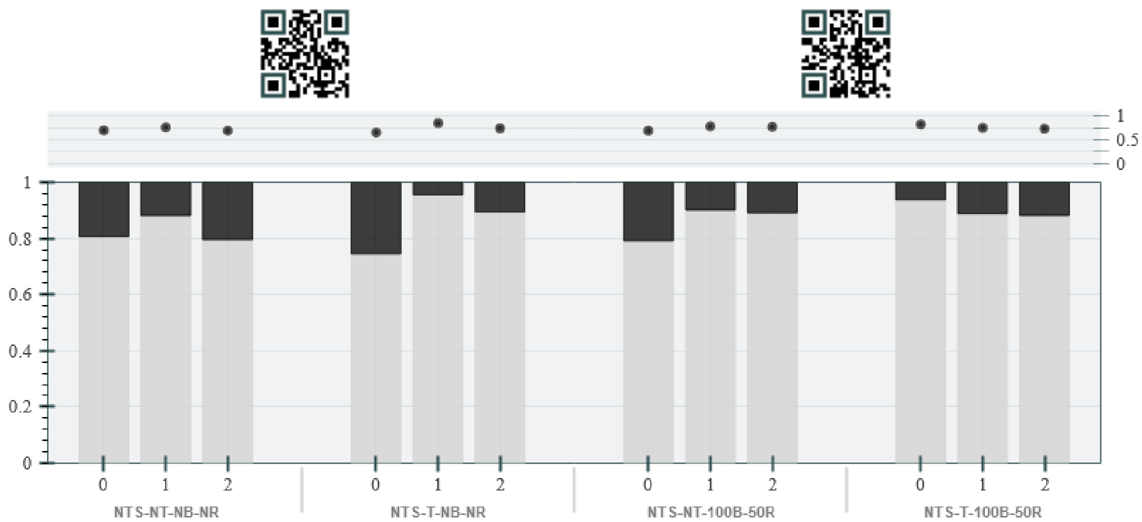


Figure 4.25: Test video 2 NTS segmentation scores.

Test Video 2: Mostly Nonsense and Dark Net

Neither 3CAS model managed to segment the tiny piece of Bright Net present at the end of the second test video. This is seamlessly handled, however, by folds 0 and 2 of NTS-NT-NB-NR. Both folds could, however, be criticised for including too much ambiguity in the Background segmentation, potentially triggering false hole reports. Fold 1 is too conservative in its segmentation, and the transfer learning models replicate their behaviour in video 1, with fold 0 producing too much Background, and the other folds tending towards inverted segmentations.

With lag masks, fold 0 of NTS-NT100B-50R yet again produces the highest quality segmentation, and the transfer learning models yet again fall short. The main concern with the best-performing segmentation is its tendency to segment areas far away or in other ways too blurry to be analysed. The segmentation in the transition area between Dark and Bright net is, anyhow, impeccable and unmatched by any other model, with fold 2 being its closest contestant. Fold 2 can though be said to be too conservative, segmenting too little Background, but could be a viable option if fold 0 eventually raises too many false alarms.

Figure 4.26: The compensation for the light reflections is fascinating but could raise false alarms, with broken net threads reported where there probably are none.

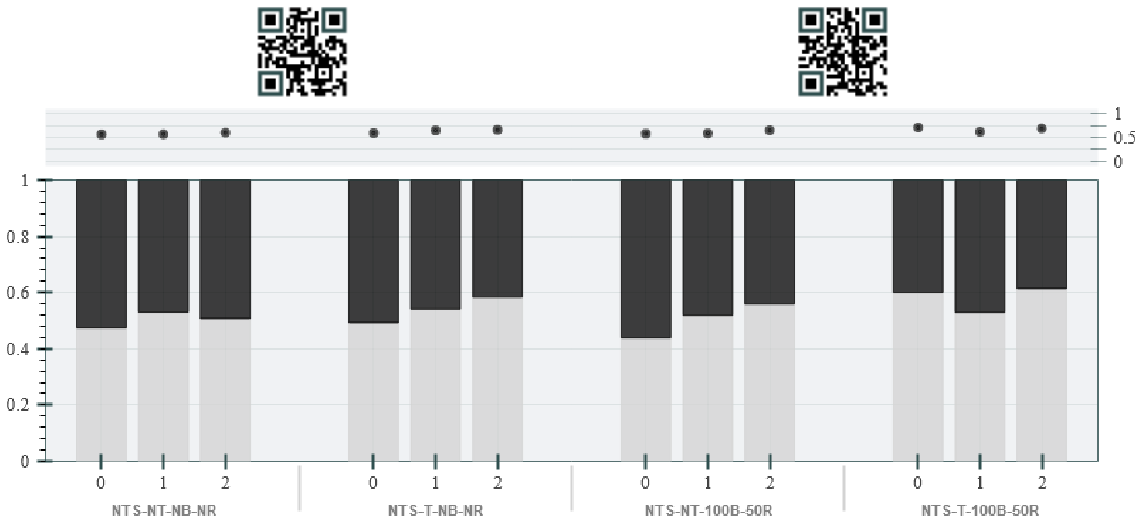


Figure 4.27: Test video 3 NTS segmentation scores.

Segmentation Test Video 3: Stable Bright Net

Either lag-less model but fold 0 of NTS-T-NB-NR performs rather well on this scene, with fold 1 of NTS-T-NB-NR perhaps gaining the upper hand, marginally, excluding some more unintelligible structure in its segmentations than the others.

With lag masks, fold 0 of NTS-NT-100B-50R stands out – remarkably – by compensating completely for a very intriguing reflection of light off the net structure (see fig. 4.26). This phenomenon was not dealt with by any other model researched, and serves to prove a remarkable asset to deep learning approaches contrary to traditional methods; the neural networks can, implicitly, learn patterns from the training data which one could never be able to describe with a rigid set of rules.

Overall, segmentations are very stable, and any model with lag masks could be set to replace fold 0 of NTS-NT-100B-50R should a more conservative model be desired.

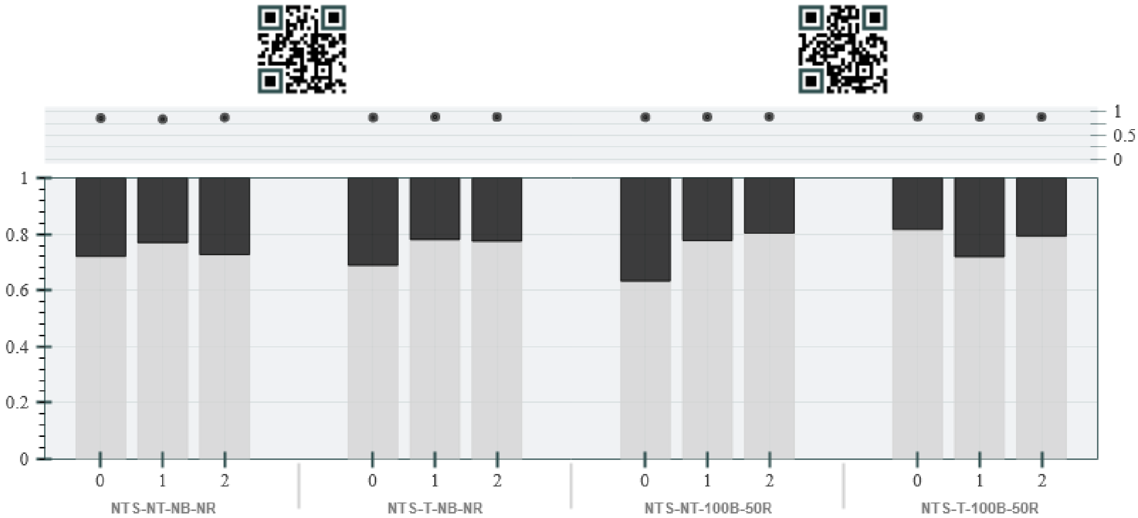


Figure 4.28: Test video 4 NTS segmentation scores.

Segmentation Test Video 4: Stable Classes

Neither NTS model matches the best-performing 3CAS model on this particular scene. With no lag mask, folds 0 and 2 of NTS-NT-NB-NR come arguably closer than the others, but their segmentations are quite more unstable than the nature of scene implies. They also report large pieces of Background in the transitional area between Bright an Dark Net.

Introducing the lag mask, fold 0 of NTS-NT-100B-50R produces, again, a very good segmentation of the net threads. Its drawback is, however, its inclusion of ambiguous areas in the segmentation, yielding results which will most certainly trigger false positive reports.

Verdict

Segmenting net threads directly facilitates programs with less overhead and sleeker information flow than counterparts relying on 3CAS segmentation interweaved with traditional binarisation schemes.

Introducing lag masks, with identical parameters with regards to blur and regularisation as discussed with 3CAS, seemed also to benefit the NTS models, with fold 0 of NTS-NT-100B-50R almost certainly being the better of the lot in all test videos. This model proved repeatedly to deliver segmentations better than those achieved with 3CAS in all videos but the last, falling only short because of its generosity in terms of including parts of the frame that are really *too blurry to be analysed* in the segmentation.

It is believed that, with such uplifting results achieved with a very small training foundation of 200 images, an effort to construct a larger dataset with NTS images will pay off in extremely robust net segmenting machine learning models. It is most certainly not necessary hereafter to pursue traditional methods to achieve net segmentations, if sufficient hardware can be acquired to back the computational demands of a MultiRes U-Net or similar neural networks, as the results achieved with deep learning far supersede those of traditional methods.

However, traditional methods and 3CAS were essential components of this work for a substantial amount of time. They will therefore be discussed in detail in upcoming chapters. Implementational details of such systems can perhaps be of interest for some other research, but – concerning net structure segmentation – it seems that deep learning outperforms the traditional techniques by a significant margin.

Complexity-wise, the 512 x 512 pixel NTS models with lag masks execute in approximately 31 milliseconds on the NVIDIA Titan X GPU, which qualifies for real-time, barely. However, with the additional hole detection scheme (yet to be discussed) real-time usage is not yet an option. Considering the proof-of-concept nature of this work, however, the numbers are not too far from the desired threshold, and it is believed that optimisations can be made to attain this.

Chapter 5

The Local Irregularity Detector

What humans perceive as video is in fact an array of static images, *frames*, each on display for the fraction of a second before replaced by the next. Video analysis can simply be analogous to *image analysis* if one analyses each frame individually. However, our human flesh-and-bone video analysis capacities reach beyond frame-by-frame analysis. The richness of the information more or less hidden in-between frames, in the flow of such, is literally beyond measure but equally harder to capture.

What is a hole? On a frame-by-frame basis, it is at least a deviation from the regular pattern within an area of elsewise regular net structure. But so are fish, so is floating algae, and so is temporary flickering in the midst of a perfectly intact piece of net. Just like the U-Net in chapter 4 was tweaked to encapsulate spatial information, so should, intuitively, the hole detection module also be able to separate swimming fish and floating algae from actual holes.

At some point, however, needs the hole be detected, if not verified, for the first time. This detection should take place on a frame-by-frame basis, that a lightweight module should analyse each frame and mindlessly report sufficient deviations from the norm. Acknowledging the inevitability of false reports from flickering frames and alien objects such as fish and algae, additional, smarter, modules, should be designed to put the local reports under scrutiny, but the process of detecting and eventually verifying holes naturally springs from bottom-up, starting on a single frame, with the Local Irregularity Detector.

5.1 A Binary Reality

Local irregularity detectors have already been investigated by Haugene [17], Jakobsen [19], Betancourt et al. [20], and Paspalakis et al. [21], and Zhao et al. [22], none of whom termed their algorithms *local detectors*, but their damage assessment algorithms were in essence that: local frame-by-frame detectors. Haugene's *damage detection algorithm* found large connex pieces of Background pixels in a binary image using morphological operations with a disk-shaped kernel of user-defined size. Jakobsen's *mesh detection* looked for consistent white (mesh) lines spanning the entire black-and-white frame using depth- and breadth-first searches, and then compared the relative distances between the horizontal and vertical lines. Betancourt's *damage detection process* reconstructed the net pattern from a binary image, eventually recognising knot points connections and identifying damage as missing connections. Paspalakis et al. proposed two algorithms: one based on white mesh pixel counting, and one based on Hough line detection in a binary image, and relative distance calculation (similar to Jakobsen's approach). Zhao et al. compared mesh hole areas and the distribution of knot points. The common denominator? The binary representation of the original frame.

Given the nature of the problem – detecting irregular holes in net structure – it seems reasonable to reduce the complexity of the original frame down to the two primary components needed for solving the problem, namely Net Structure and Background. These terms are in a binary sense analogous to Foreground and Background, and the *least complex representation* of the original frame in which holes can be recognised is thus an image consisting of zeros and ones – in this work representing *Background* and *anything else*, respectively. If a binary image can be successfully created, then formulating what constitutes an irregularity can be more easily deduced:

1. An irregularity is a deviation from the ordinary structure within Foreground pixels, or
2. an irregular piece of Background, or
3. an irregular interplay between the two

where, generally, Olsen's line search, Betancourt's knot point reconstruction, and Paspalakis' Hough line distance comparison belong to the first school of thought, and Zhao and Haugene's detection of large pieces of Background to the second, and Paspalakis' pixel count method to the latter. Even though the pixel count method did not count Background pixels per se, it deemed areas containing *too little* Foreground irregular, and such worked with the hypothesis that each area should contain a certain amount of Foreground, and a certain amount of Background.

Granted, the binary image is a proper starting point for a traditional irregularity detection algorithm. We have already shown that NTS can be implemented with U-Nets to provide an excellent binary equivalence of reality. However, traditional methods have been thoroughly investigated in this work, and it is worthwhile understanding properly how traditional methods have been implemented in earlier works (and – in fact, in this work, coupled with 3CAS) to fully appreciate what challenges are circumvented with deep learning assistance.

How does one proceed to create the optimal binary image? The aforementioned authors favoured not the same approaches, and neither utilised the approach favoured in this work. Some of the most popular algorithms to achieve binarisation have already been discussed in 2 (thresholding algorithms and edge detectors), but their strengths and weaknesses as related to this topic deserve investigation.

5.1.1 A Preliminary Sidetrack: The Grayscale Image

All traditional binarisation approaches to be discussed originate in a grayscale image, and not the entire colour space of the original frame, so a brief mention of colour to grayscale image conversion is needed.

Haugene decided to utilise *all* three available colour channels, in addition to *hue* and *saturation*. His binarisation schemes were run on each channel, individually, and later combined in a single, coherent, binary image using a binary OR operation. Jakobsen, on the other hand, attempted to work on each channel, but wound up favouring the Red channel after trial and error.

Apart from working on single channels, one might *blend* the RGB-channels to create an image I with single grayscale channel which encapsulates all in one. OpenCV's implementation[1] is:

$$I_{gray} = 0.299 \cdot I_{red} + 0.587 \cdot I_{green} + 0.114 \cdot I_{blue} \tag{5.1}$$

Neither Paspalakis [21] nor Betancourt [20] describe their grayscale conversion, so whether or not they operate on mixed or separate channels is down to pure speculation. This work, however, will *exclusively* work on the mixed grayscale image. The reason for which is mainly two-fold.

Firstly – analysing a single, mixed, channel, is three times faster than analysing three channels, and six times faster than analysing six channels. If it can be reasonably assumed that the mixed channel provides a *sufficient* representation of the three basic channels, then it is favourable from a computational perspective. Secondly – since this work implements the local irregularity detector as a low-cost module which can be allowed to produce false positives (later filtered by other modules) then it is simply not a problem if a few frames over-report irregularities. Haugene's [17] reasoning for utilising as many channels as possible was indeed to attain conservatism with regard to Foreground segmentation – given his binary OR combination of all channel evaluation outputs. And, since eq. (5.1) suppresses the intensity of all channels, then, necessarily, will the resulting mix yield high intensity values for only those pixels in which all channels *combined* yield a high response, whereas Haugene's combinatorial approach allowed high responses in single channels to manifest themselves in the resulting binary image.

One could, rightly, criticise eq. (5.1) and wonder why not all channels have been granted an equal share of the mix:

$$I_{gray} = \frac{1}{3} \sum I_c, \quad c = \{red, green, blue\} \tag{5.2}$$

[1] https://docs.opencv.org/3.4/de/d25/imgproc_color_conversions.html

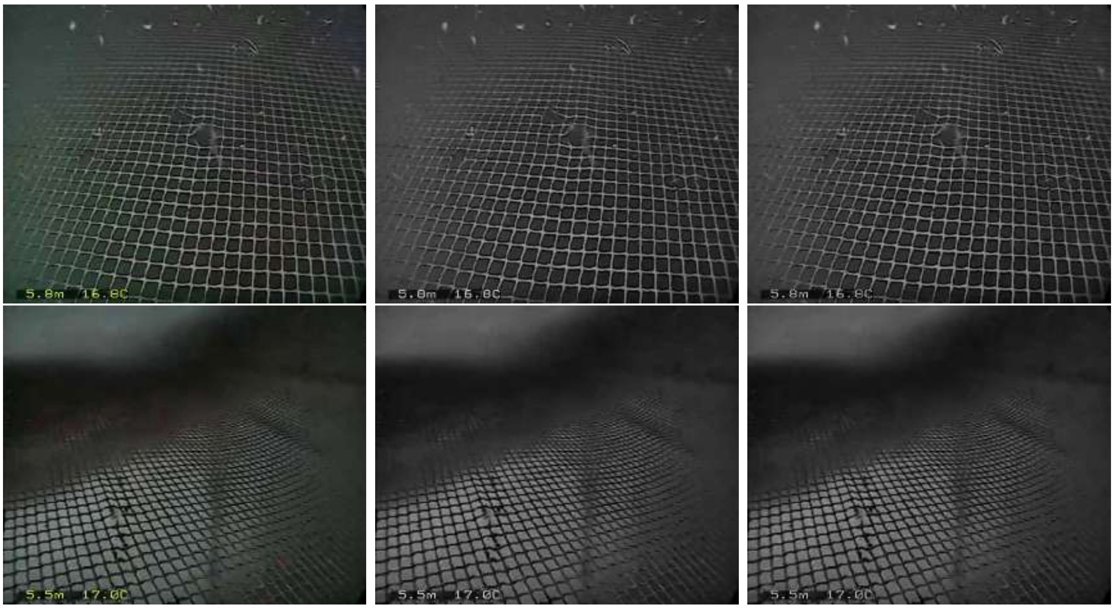


Figure 5.1: The leftmost images are original RGB frames, followed by grayscale representations achieved with eq. (5.1) and eq. (5.2), respectively. Chances are, you can not spot the difference with the naked eye. However, a computer could.

After all, the mix is to be exploited by a computer and not to be viewed by a human. The difference is not grand, but there might be a certain advantage to utilising all channels equally, and given shifting underwater scenes and conditions, the different colour channels will necessarily fall in and out of favour. Monitoring which channel should be more dominant given a certain scenery has not been investigated in this work, and it seems not perfectly clear whether such endeavour is feasible nor significantly rewarding.

Finally, concluding this brief discussion of grayscale conversion, this work has utilised eq. (5.2) to convert the original RGB frame to a single-channel grayscale image. The effect of eq. (5.2) versus eq. (5.1) is marginal, as visualised in fig. 5.1.

5.1.2 Binarisation through Manual Pixel Intensity Thresholding

The simplest road from grayscale to binary image is indisputably through pixel intensity thresholding:

$$I_b = \begin{cases} 0, & \text{where } I_{gray} < T \\ 255, & \text{where } I_{gray} \geqslant T \end{cases} \tag{5.3}$$

where I_b is the resulting binary image, I_{gray} the grayscale image, and T a pixel intensity threshold in $[0, 255]$. It should be mentioned as a side-note that a *binary* image can be addressed as an image consisting of zeros and ones, or zeros and 255s, interchangeably.

Jakobsen experimented first with setting T manually, an approach which can work very well in certain scenarios, but not so much in others. For instance,

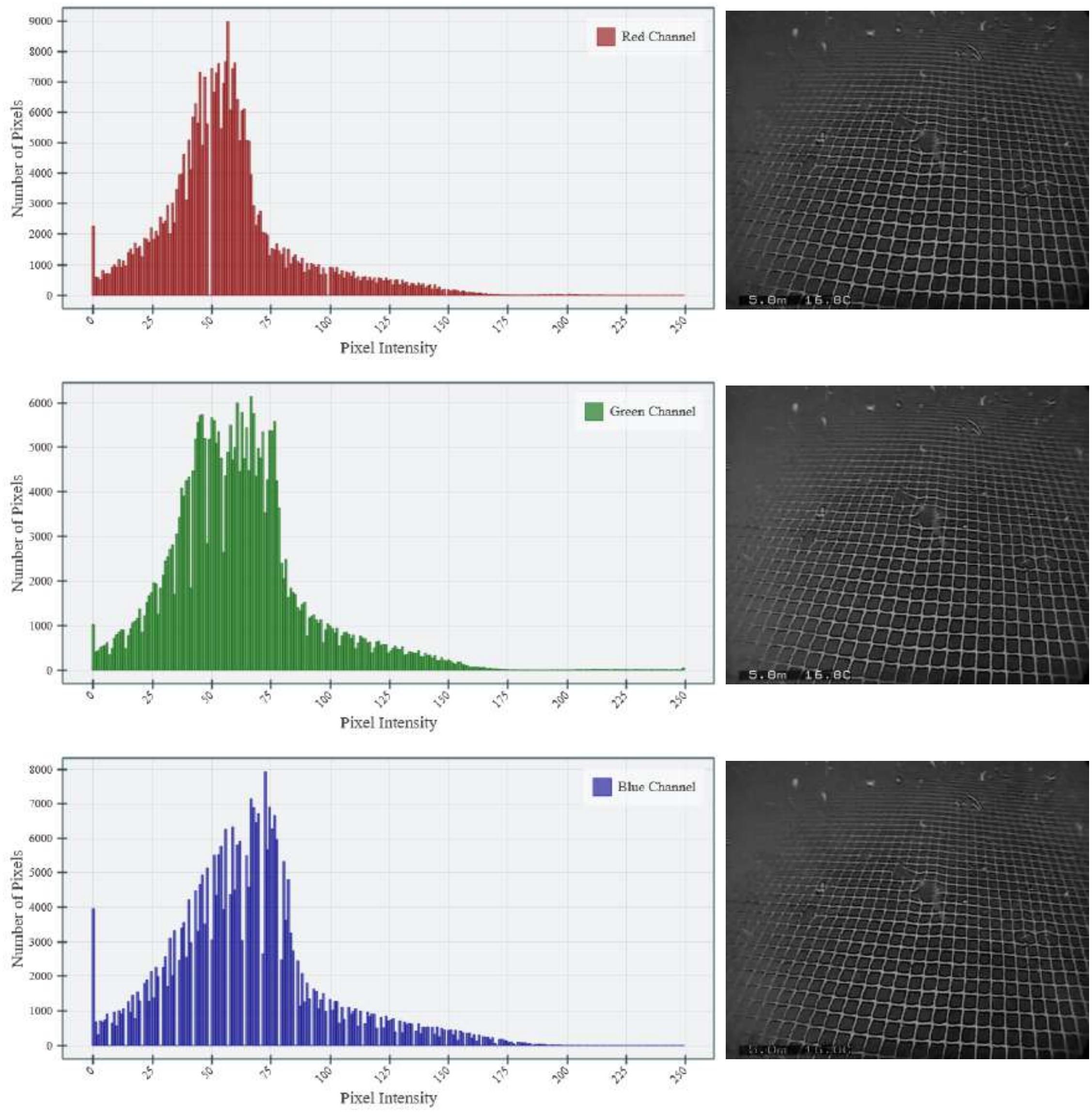


Figure 5.2: The histograms of the three channels (Red, Green, Blue) and their respective single-channel grayscale image, can be analysed to identify an optimal binarisation intensity threshold.

the pixel intensity values in fig. 5.2 indicate that the threshold exists somewhere between 0 and 150, whereas a brighter scenery would skew the distribution further to the right. Deducing an intelligible optimal threshold value from these histograms is not really feasible, since the two hypothetically present classes (Background and Net Structure) do not separate themselves into well-behaving clusters in the histograms. The value must therefore be approximated, then optimised based on visual impression. See for instance how the quality of the binary images in fig. 5.3 changes radically with perturbations to the thresholding value. Constantly updating the thresholding value like this is very inconvenient.

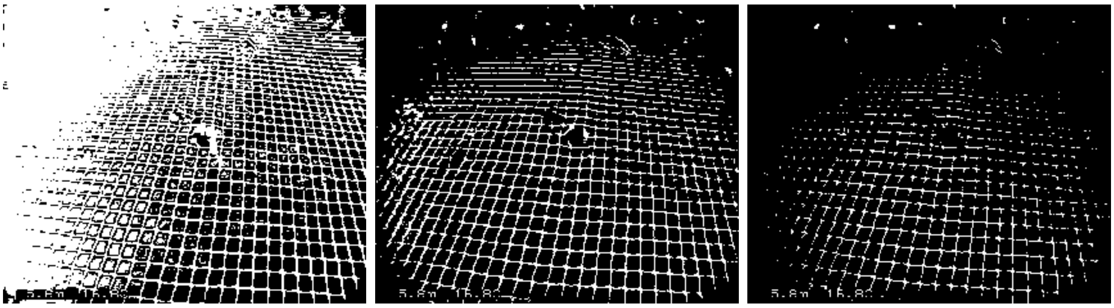


Figure 5.3: With global thresholding, the threshold value must be carefully selected in order to achieve superior binary results. Here, starting from the left, the threshold T has been set at 50, 75, and 100. The original grayscale image subject to thresholding is located in the upper right corner of fig. 5.1.

5.1.3 Binarisation through Otsu's Method

Otsu's method, as discussed in chapter 1, yields an effective way of automatically choosing the optimal threshold T based on the pixel intensity histogram of a grayscale image. However, complex scenes arise underwater where a single threshold simply does not suffice. For instance, in fig. 5.4 it is strikingly apparent how the binarisation succeeds in the upper row, but fails in the lower row. Most of the image depicts Net Structure, but a global thresholding value fails to include large parts of the net in the binarisation. This tendency will be fatal to an irregularity-detection algorithm, since black areas in the binarised image assumably indicate holes.

Otsu's method brings to the table the very desirable unsupervised optimal threshold selection, but falls short in complex scenes where the homogeneity assumption of the classes is not true. Due to the observed inadequacy of Otsu's method in complex scenes, it is dismissed as a binarisation candidate in this work. Otsu's method was nevertheless favoured by Betancourt [20], Paspalakis [21], and Zhao [22].

5.1.4 Binarisation through Adaptive Thresholding

Optimal global threshold selection, be it manual or automatic, fails when there exists no global threshold to effectively separate Net and Background across the entire image. So, could one achieve binarisation through *local* optimal thresholding? The answer is *yes*, through adaptive thresholding [41].

With adaptive thresholding, each pixel is evaluated based on the pixel intensity distribution of its neighbourhood. This is obviously an extremely desirable trait to counter the aforementioned problems posed by complex gradients. The block size of the local neighbourhood needs to be carefully selected, to make sure it encapsulates local instances of Net Structure and Background.

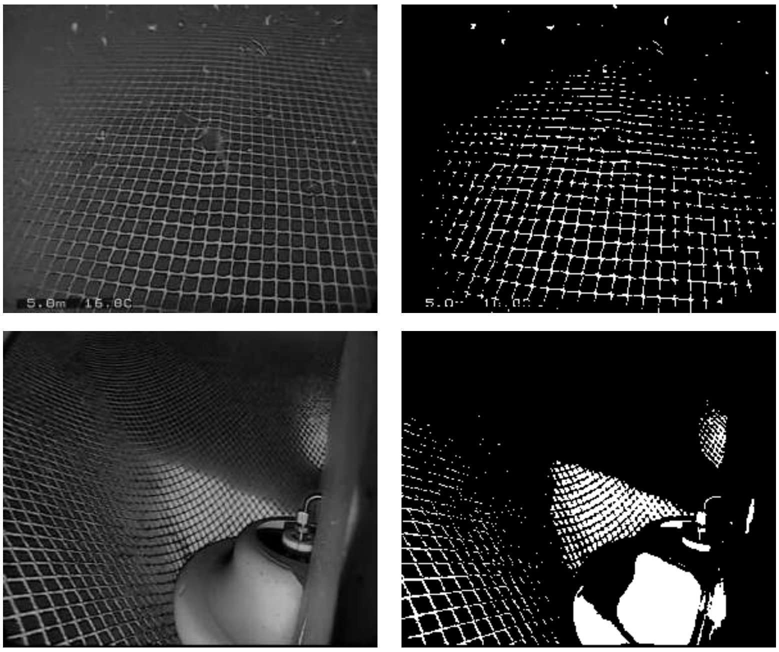


Figure 5.4: Otsu's method effectively finds the *single* optimal binarising threshold value T. The problem, however, is that a single T is not sufficient if the image is complex. In the upper row, the Net Structure and the Background is rather homogeneous across the image, whereas the bottom image has a complex lighting gradient which causes the net to appear both bright, dark, and everything in between, in the same image. This caused much of the bottom right binary image to appear black, even though most of the grayscale image displays Net Structure.

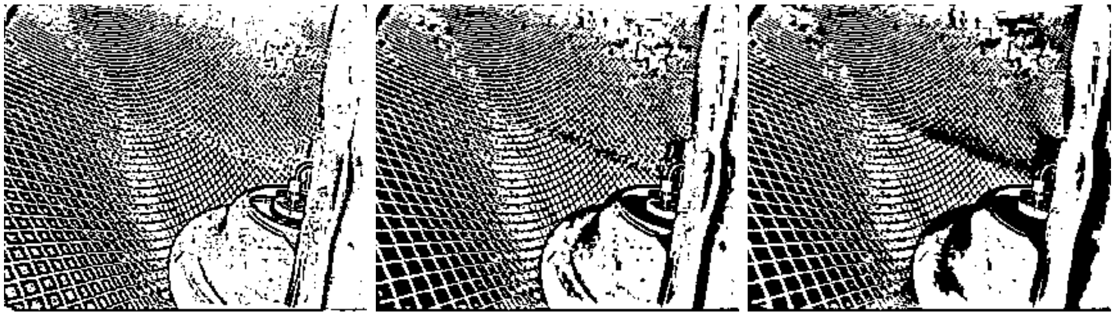


Figure 5.5: The adaptive thresholding algorithms classifies each pixels based on the distribution of its local neighbourhood. The neighbourhood block sizes utilised here were 21, 51, and 101, respectively.

In fig. 5.5 the leftmost image has too small a neighbourhood, whereas the rightmost arguably has too large a neighbourhood. Too small a neighbourhood manifests itself in the characteristic *filled holes* in the leftmost image: when evaluating Background pixels in the middle of a mesh, the neighbourhood does not include any Net Structure pixels and so the pixel subject to classification merely compares itself to other Background pixels. In the rightmost image, a diagonal dark area in the middle of the net appears since the very local nuances between Net Structure and Background fail to be recognised. Since each pixel is evaluated based on the neighbourhood, both Net pixels and Background pixels were, in that area, both relatively dark when expanding the neighbourhood.

Intuitively, the size of the neighbourhood should also, ideally, not be global, but rather derived as a function of the local mesh size. This will be discussed in section 5.4.1. .

5.1.5 Binarisation through Edge Detection

The binarisation schemes discussed thus far all revolve around pixel intensity thresholding, but additional classes of such schemes exist. Edge detectors, for instance, segment pixels based on intensity *variation* and transition instead of pixel value per se. The edge detectors have one certain advantage to the thresholding algorithms: notice that the Net Structures in the upper and lower row of fig. 5.1 are bright and dark, respectively. A thresholding algorithm with no knowledge of *what* it was evaluating would necessarily represent the Net as *white binary class* in the upper row, and as *dark binary class* in the lower row. In more complex scenarios, such as that of fig. 5.5, the Net is *both* dark and bright within the very same image, and the binarising scheme would need help from other modules in order to correct the binary representation, and make sure all Net Structure was handed the correct white class label. An edge detector, however, would disregard the fact that the Net was either dark or bright. It simply yields response to the fact that there is a sharp transition in pixel intensity value where the Net Structure occludes the Background.

Three different edge detection algorithms were discussed in researched work: Jakobsen and Haugene both explored Canny's algorithm, but neither used it. Jakobsen stuck with intensity thresholding and Haugene introduced his own edge detector, the Local Sharpness Detector. Jakobsen also investigated the Laplacian of Gaussian (Marr-Hildreth), but it, also, fell out of favour with him .

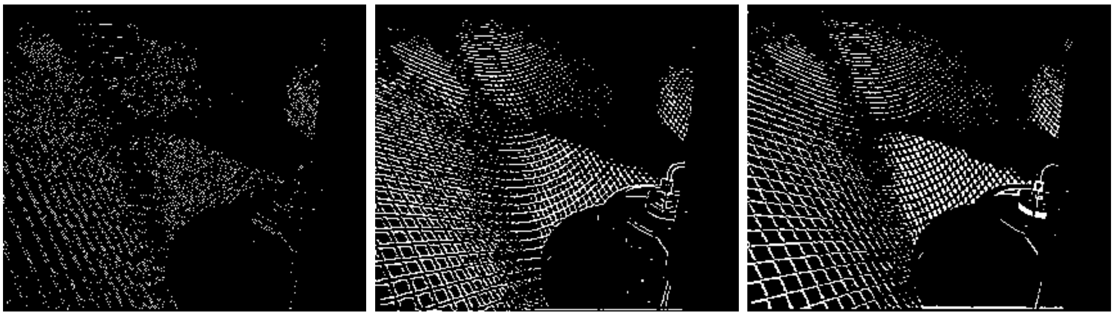
Canny's Algorithm

Canny's algorithm [39] is an extremely popular edge detector widely used in various application since its advent in the 1980s. The algorithm aims to provide robust edge detection, in terms of suppressing weak or false edges, but still incorporating edge parts that might suffer from varying illumination. However, its reliance on edge intensity thresholds introduces need for human fine-tuning.

Such tuning might provide decent edge detection for certain scenes, whereas shifting scene conditions, conversely, render the edge detection poor or even useless. For instance – conditions which suffer from very low illumination will limit the edge intensities, in which case the Canny parameters should be set to detect very fine intensities. These thresholds ultimately lead to false edge detections under of abundant illumination. In fig. 5.6a the parameters have been fine-tuned for that very scenario. Apart from yielding the same response to the (originally) dark and bright Net, it fails to segment as much Net Structure as did the adaptive thresholding algorithm (fig. 5.5). Since Canny's algorithm apparently yields worse results than achievable with adaptive thresholding, in addition to requiring very specific fine-tuning, it was not considered further in this work.

Marr-Hildreth's Algorithm

The Laplacian of Gaussian (also called Marr-Hildreth's algorithm) is an image analysis classic, proposed in the 1980s by David C. Marr and Ellen C. Hildreth [46]. Its performance on the sample scene in fig. 5.6b is quite similar to that of Canny's algorithm, which was generally decent. However, it also fails to include large parts of the Net Structure, specifically, those parts which are somewhat blurry. The algorithm itself is designed not to respond to blurriness (given the *Gaussian* part), so it will naturally respond only to the sharper parts of the image. The severity of the blurring can be fine-tuned by humans, and the algorithm was in this work perceived as hyper-responsive to perturbations in its parameters. As was the case with Canny's algorithm, shifting sceneries require shifting parameters to achieve acceptable binarisation results. This works aims to avoid what Haugene called his *myriad of tuning parameters*, so Marr-Hildreth's algorithm was also left aside due to better options.



(a) Canny's algorithm. (b) Laplacian of Gaussian. (c) Haugene's edge detector.

Figure 5.6: The edge detectors offer a new take on binarisation. If perfectly tuned, then both Canny's algorithm and the Laplacian of Gaussian detect net threads as white responses indifferently of the net pixel intensities being dark of bright. However, Haugene's edge detector highlights sharp and bright pixels by nature.

Haugene's Algorithm

The local sharpness detector proposed by Haugene [17] subtracts from the original image its blurred equivalence. Albeit not concerned with derivatives like the previous two edge detectors, it is an edge detector in the sense that it locates brief instances of *sharpness*. In fig. 5.6c Haugene's algorithm is, impressively, well on par with the aforementioned edge detectors, albeit with one drawback: his edge detector yields inverse response for dark Net Structure. His edge detector effectively retains only those pixels in the original image which are bright, and surrounded by a less bright neighbourhood. This is not to discard his approach completely, the response is very similar to that of the adaptive thresholding algorithm. However, his approach struggles when there is little sharpness in the original image. If an image is blurry, then its blurry equivalence is, well, also blurry. The difference is zero. These scenarios are simply better handled by adaptive thresholding.

5.2 Binary Correction: Coupling U-Net and Binariser

An extraordinaire component of this work is the integration of the U-Net to filter non-relevant parts of the frame. Whereas previous work on the field has typically assumed the entire frame to be of interest to the hole detector, this work acknowledges the fact that a real-life application needs to handle realistic scenarios.

Recall from chapter 4 that the 3CAS U-Net identifies three classes: Bright Net, Dark Net, and Nonsense (see for instance fig. 4.5). These three classes make perfect sense when recognising in fig. 5.5 that parts of the binarised image are inverted due to the appearance of Dark Net in the original image.

The first version, say, of the U-Net produced a two-class output, namely *Net* and *Nonsense*. This type of output demanded a significant amount of work to detect whether or not the scene in question showed Dark or Bright Net Structure. With traditional computer vision techniques, this was effectively solved by assuming there to be *more disjoint Background regions than disjoint Net Structure regions*. If indeed there were more *white pixel* regions than *black pixel* regions, the binary image was inverted. Ideally, one is faced with a single Net region and several, smaller, Background regions. However, motion blur and poor image quality often caused Net Structure connections to appear broken in single frames, and hence there could exist frames in which there were in fact fewer Background regions than there were Net Structure regions. To counter this was included a consensus module which demanded a certain continuity in the call for binary inversion before its application.

Other statistical attempts were implemented to make a better binary correction module. However, even if they would succeed in scenes in which the entire binary representation was inverted (i.e. the entire Net was dark such as in the lower row of fig. 5.1), they would not be able to handle situations where parts of the image contained Dark Net, and the other half Bright.

By adding an extra class to the output layer of the neural network (going from NeNoS to 3CAS) it turned out to be largely learnable by the U-Net whether or not the Net in question was Bright or Dark. In fig. 5.7 it is visualised how the Binary Correction module operates with the U-Net: The three-class prediction mask shown in fig. 4.5 is split into three separate masks; White Net Mask, Dark Net Mask, and Nonsense Mask. Summing the product of White Net Mask and the original binary image, the Nonsense Mask, and the product of Dark Net Mask and the inverted binary image yields binary coherent representation where *any black pixel region* can be treated as proper Background.

5.3 Detecting Irregularities

With a coherent binary image in place, regardless of its origin (either from 3CAS and subsequent binarisation, or from NTS), irregularity detection can finally take place. This work has specifically aimed to develop Haugene's [17] damage detection algorithm, and thus detects irregularities as *an irregular piece of Background*.

Choosing *one* necessarily implies rejecting *all else*. Paspalakis' [21] pixel counting approach and Zhao's [22] area comparison algorithm were abandoned due to concerns with their naïveté. Broken connections in the Net Structure do not necessarily alter the number of Background pixels in a sub-part of the image significantly, causing significant worry for Paspalakis' method. In addition, algae growth and such could make the Net Structure appear *fatter* in the binary image, altering the relationship between Net and Background pixels in that part of the image.

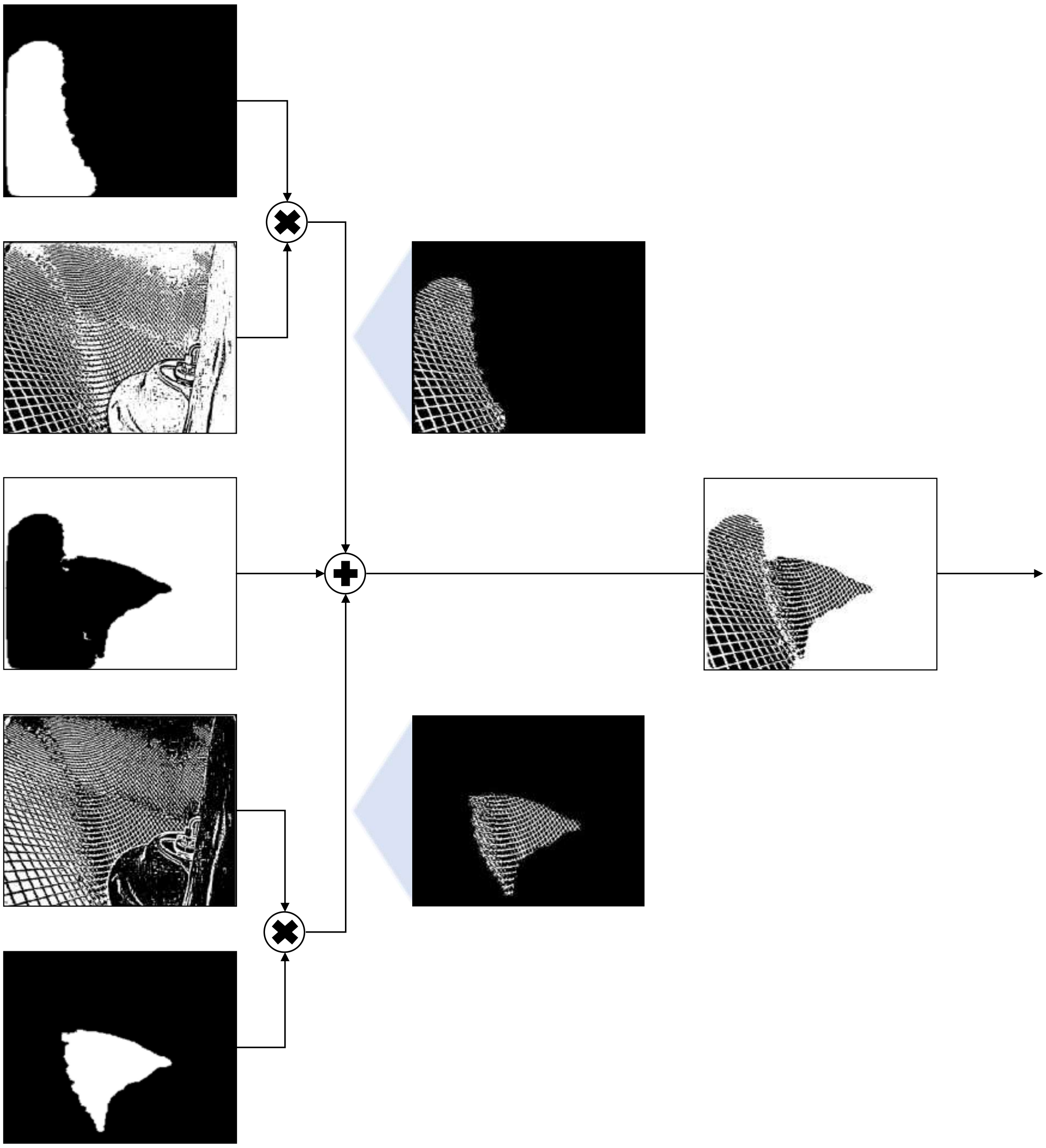


Figure 5.7: Five binary images combine to form a coherent binary image in which irregularities necessarily manifest themselves in large regions of black pixels. Starting from the top, in the leftmost column: White Net Mask, produced by the 3CAS U-Net. The original binary image, produced here by the adaptive thresholding algorithm. The Nonsense Mask, produced by the U-Net. The inverted binary image. The Dark Net Mask, produced by the U-Net.

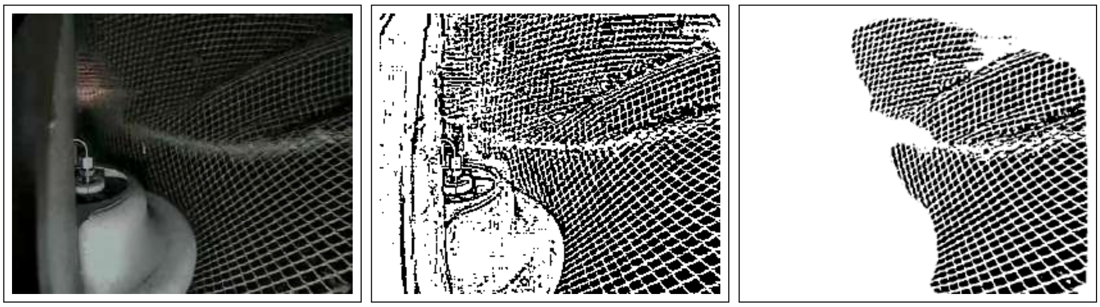


Figure 5.8: A scene and its binary equivalence, as suggested by a U-Net. The cleaner's impact on the net causes it to deform, no longer depicting even and straight lines easily analysable with Hough transforms.

Detecting Hough lines was favoured by both Betancourt [20] and, eventually, Paspalakis [21], but rejected by Jakobsen [19] and Olsen [58]. Given perfect circumstances, in which the net forms even and straight lines, this method is likely to succeed. However, most scenes during cleaning operations look like fig. 5.8, where the cleaning robot causes the net to deform. Such scenes are not easily handled by the Hough transform. The same goes for Jakobsen's line search algorithm, which innately assumes lines to be parallel.

5.3.1 Haugene's Damage Detection Algorithm

Tormod Haugene proposed morphological operations *as opposed to region growing* given its capability to recognise the intactness of Net Structure which is not necessarily entirely intact in the binary image. Typically, motion blur causes Net Structure to appear broken in occasional frames. If one were to apply region growing to each Background pixel, large regions would inevitably arise from such troublesome areas. This is well visualised in fig. 5.9. Even though the quality of the binary image was relatively good there, tiny, yet present, broken connections bridge several Background regions to form connex regions of relatively large size.

Haugene's approach consisted of first *closing* the binary image with a disk-shaped kernel of radius 14, then *opening* the closed image with the same kernel, to reconstruct the original shape of the damage. There seem to be at least one peculiarities with this reasoning. Firstly, the radii of the kernels were designed to *exclude most false positives and include most true positives* given a very small test video (19.4 seconds, displaying 11 different scenes in an artificial net set-up). This assumption is decent enough as proof-of-concept, but far from acceptable in a real-life application. For instance, fig. 5.10 illustrates clearly how false positives are detected due to the relatively large size of regular pieces of Background. Instead, an adaptive approach should calculate what size constitutes *regular* Background, and then search for pieces of Background sufficiently *larger* than that. Additionally, the closing and opening operations are themselves designed to retain the original size of the different regions. Applying opening to the closed image in order to *reconstruct the original shape of the damage* makes little sense.

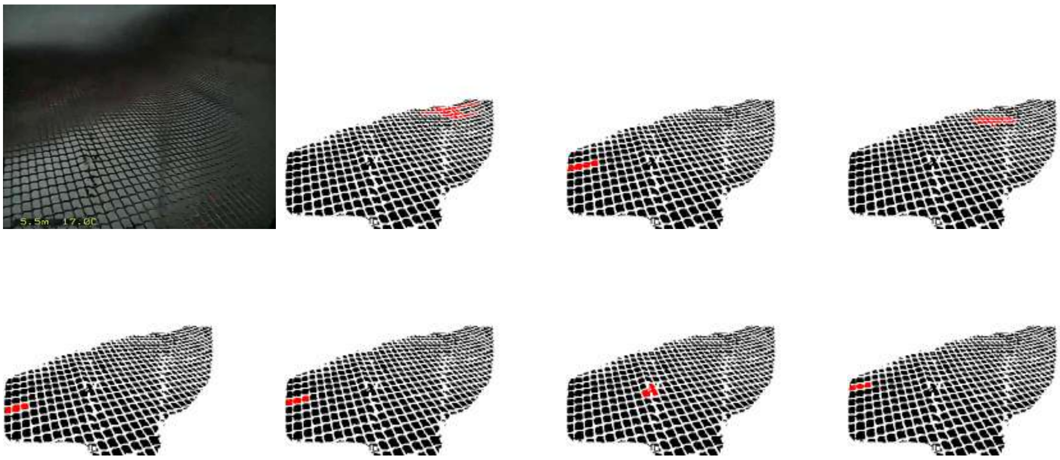


Figure 5.9: Naïve region growing approaches identify false holes where there are none due to apparent broken connections in the binary image. The real net tear was in this image recognised as the **sixth** largest region.

(a) Binary image. (b) Reported damage.

Figure 5.10: Haugene's proposed opening with a disk-shaped kernel of size 14 does not necessarily report true damage if the radii of the normal Background regions in the binary image exceed 14 pixels.

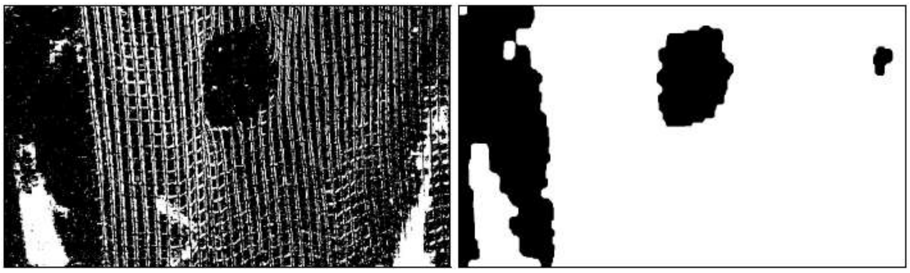


Figure 5.11: Haugene's video material depicted extremely severe damage, and his choice of structuring element shape was therefore not of critical importance.

5.3.2 Refining Haugene's Kernel Shape

Haugene proposed a disk-shaped kernel to identify damage. From one of his examples, fig. 5.11, it is evident that his scenarios needed no refinement of kernel shape in order to detect his damage. The damage he analysed was virtually fifty times the size of his ordinary pieces of Background. To identify more fine-grained damage, however, is suggested the cross-shaped kernel as showed in fig. 5.12.

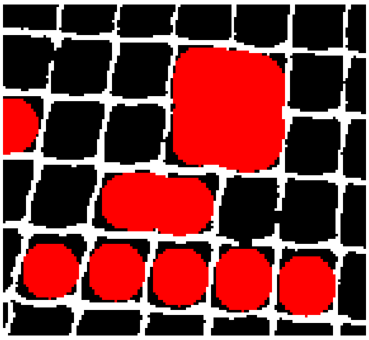
Whereas the square and circular kernels are very robust to false positive reports (they will only highlight areas which can be completely filled by the kernel), they tend to under-report actual damage where the shape of the damage is not square-like or circular. Of course, the damage is reported independently of shape when its size is of a different order of magnitude than the typical piece of Background, but when the damage is approximately the height, say, of an ordinary mesh, but rather broken in the horizontal direction, a circular kernel will simply not suffice.

One could argue that *flat* kernels such as those of fig. 5.13 should be utilised, but they will arguably over-report damage due to poor image quality. The cross-shaped kernel, on the other hand, requires some substance to the damage in both directions, whilst not the strict square- or circular shape as required by the other symmetric kernels.

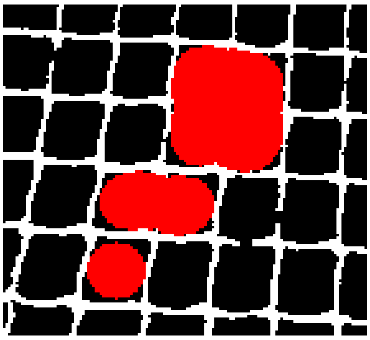
As per now, the cross-shaped kernels and the flat kernels will be tested and considered to evaluate whether or not they provide the required alertness to irregularities, and the desired robustness to false positive detection.

5.3.3 Refining Haugene's Kernel Size

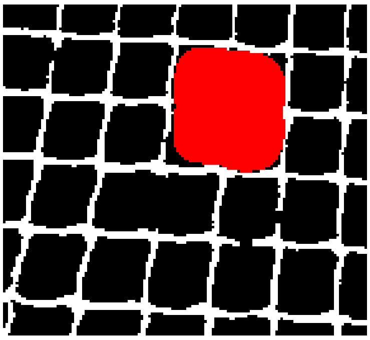
The results yielded by the, rather heuristically proposed, kernel size of 14 *obviously* depend on the characteristics of the scene. Holes will be found everywhere if the typical piece of Background is wider than 14 pixels. Instead, this work proposes an **adaptive elbow kernel size** (the Elbow), a measure of the typical Background size in the current frame.



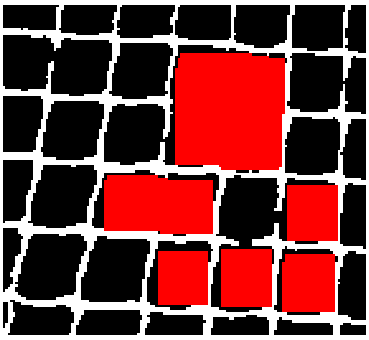
(a) 22 px disk.



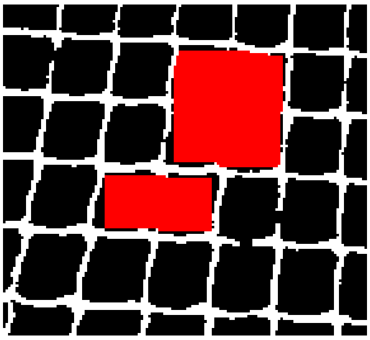
(b) 23 px disk.



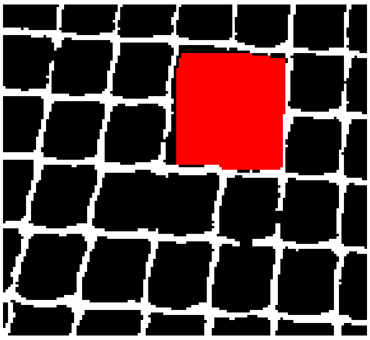
(c) 24 px disk.



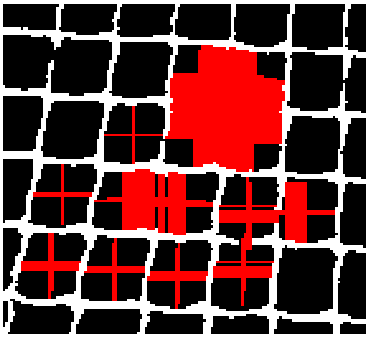
(d) 20 px square.



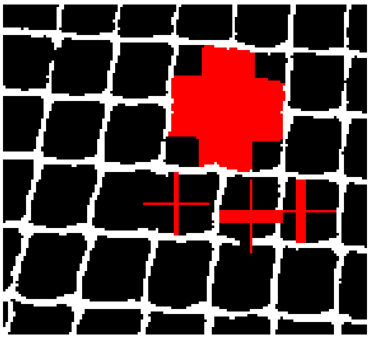
(e) 21 px square.



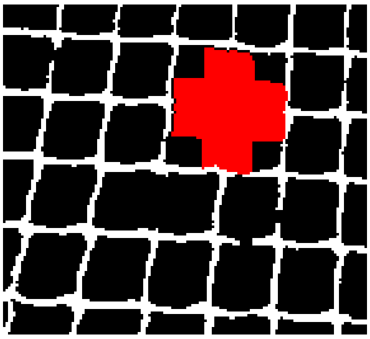
(f) 22 px square.



(g) 23 px cross.

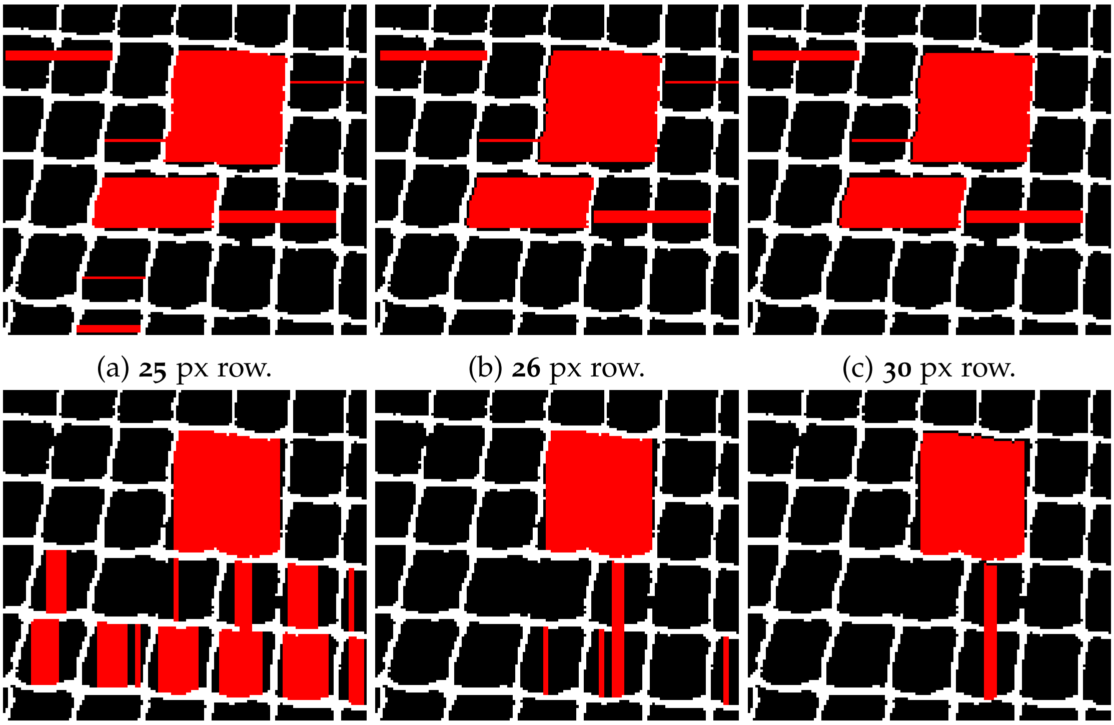


(h) 25 px cross.



(i) 26 px cross.

Figure 5.12: Different kernel shapes have their distinct advantages. The square kernel recognises well square-shape damage, but cannot fit within thin, broken, connections. If the binary image were flipped 45 degrees, it would also struggle to fit properly within the diamond-shaped Background pieces. The disk-shaped kernel is invariant to rotation, but cannot fit properly within broken connections either. The cross-shaped kernel is the only kernel to highlight the broken connections in addition to the larger damages, and it works just as well on diamond-shaped holes as square ones.



(a) **25** px row. (b) **26** px row. (c) **30** px row.

(d) **25** px col. (e) **27** px col. (f) **40** px col.

Figure 5.13: Toying with row-like and column-like kernels will allow the detection of all broken connections. However, these approaches will necessarily yield more false positives than the symmetric kernels since poor quality can give the impression of broken connections where there are none.

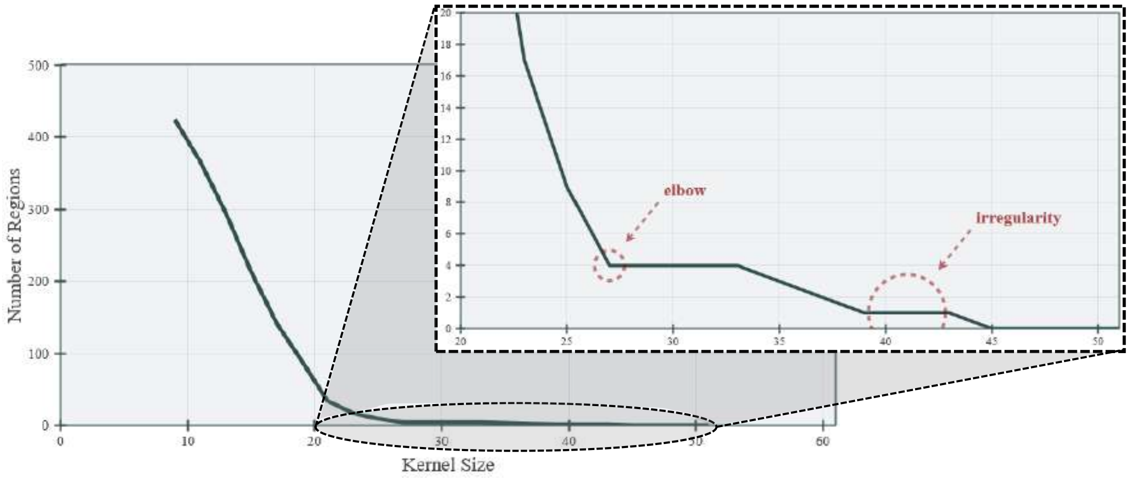


Figure 5.14: If counting persisting regions whilst closing an image with an incrementally increasing kernel size, irregularities can be identified at plateaus where only a few regions persist. Hypothetically, there exists some Elbow, a kernel size to which *most* regions disappear, and to which true irregularities are some X% larger.

The Elbow (see fig. 5.14) is thought to be a grounding to which *relatively large pieces of Background* can be compared. For instance, instead of recognising irregularities as pieces of Background *larger than 14 pixels*, they can be recognised as **X% larger than the Elbow**. This approach allows flexibility in terms of zoom, distance to net, and angle of view.

One constant, however, which needs be set, is related to the Elbow decision. The search for the Elbow should be as efficient, but yet as robust, as can be. A proposed method is to define it as the *largest size to which all but K pieces of Background disappear during closing*. In other words, it can be approximated by gradually closing the binary image with a given kernel size, then looking for at least K remaining Background regions in the resulting image. If K regions are found, the kernel size is incremented and a new closing operation is performed.

This method could seem tedious in a high-rate demand application. The Elbow could very well be 6 or 84, which means, intuitively, that dozens of closing operations and regions recognising algorithms need be run per frame. This is not true. Two very important shortcuts can be implemented, effectively reducing the time complexity of the local irregularity detector down to a couple of milliseconds, on an ordinary computer running on a single CPU. These are the improvements:

1. Start Elbow search at the previous frame's Elbow.
2. Do not (necessarily) count *all* regions in a closed image. Stop counting and increment kernel size once K regions have been counted.

where assumption (1) works since the spatial continuity is generally high during cleaning operations, and arguably more so during inspection videos, given the typically slower pace. This suspicion can be easily confirmed when tracking the historical Elbows of cleaning videos.

Choosing K

The ideal K should ideally not fluctuate too much – enabling very efficient detection of the Elbow frame-to-frame, assuming Elbow search starts at the previous frame's Elbow. Additionally, there should be a decisive difference between the Elbow and the size of irregularities – such that an intelligible X can be deduced which effectively separates wheat from chaff.

Two kernel shapes are subject to scrutiny, namely the cross-shaped and the flat kernels. Judging from fig. 5.15a and fig. 5.17a, based on cross-shaped kernels, it could look as though $K=\{3, 5, 10\}$ yield often approximately the same Elbow, especially so in fig. 5.17a. However, in both scenarios are the $K=3$ graphs more inclined to fluctuate with $K=1$.

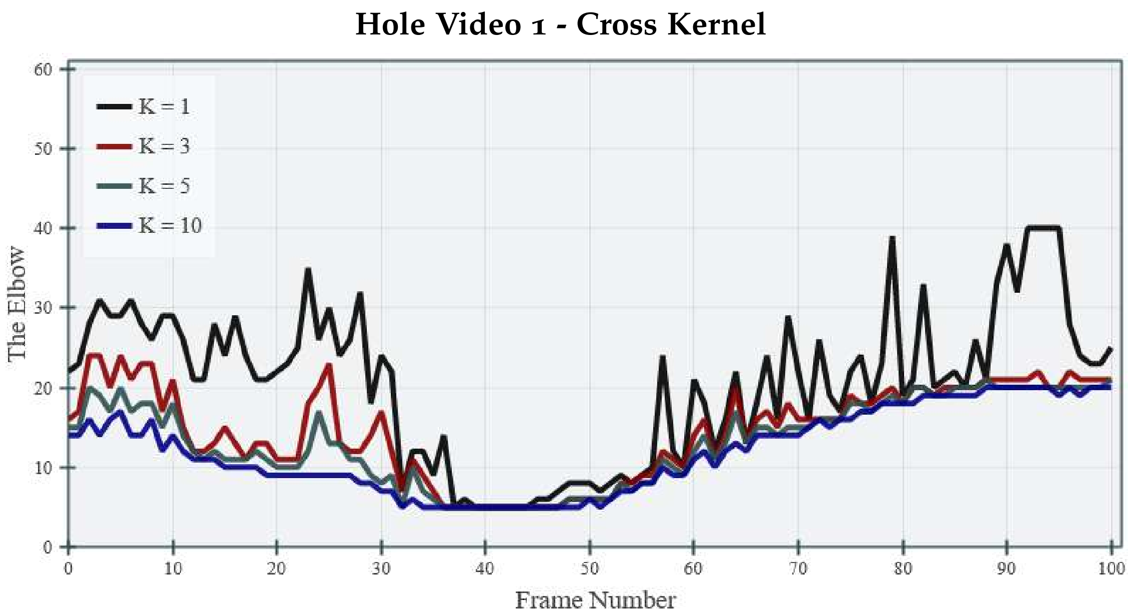
Intuitively, choosing $K=3$ allows one not to discover more than at most **two** irregularities, since the defining characteristic of an irregularity then is based on its size relative to the **third** largest piece of Background. Also, choosing $K=10$ increases time complexity two-fold compared to $K=5$, since ten regions need be counted for each iteration, as opposed to only five. Based on these observations, it seems that **$K=5$ is a sufficient middle ground** which allows for the detection of four irregularities within a frame, still providing a grounding that is almost as stable as the higher-complexity candidate $K=10$.

This behaviour seems to emerge also with the flat kernels, as observable in fig. 5.16a and fig. 5.18a, where $K=\{3, 4, 5\}$ typically stabilise at approximately the same value, especially when there is a clear hole present.

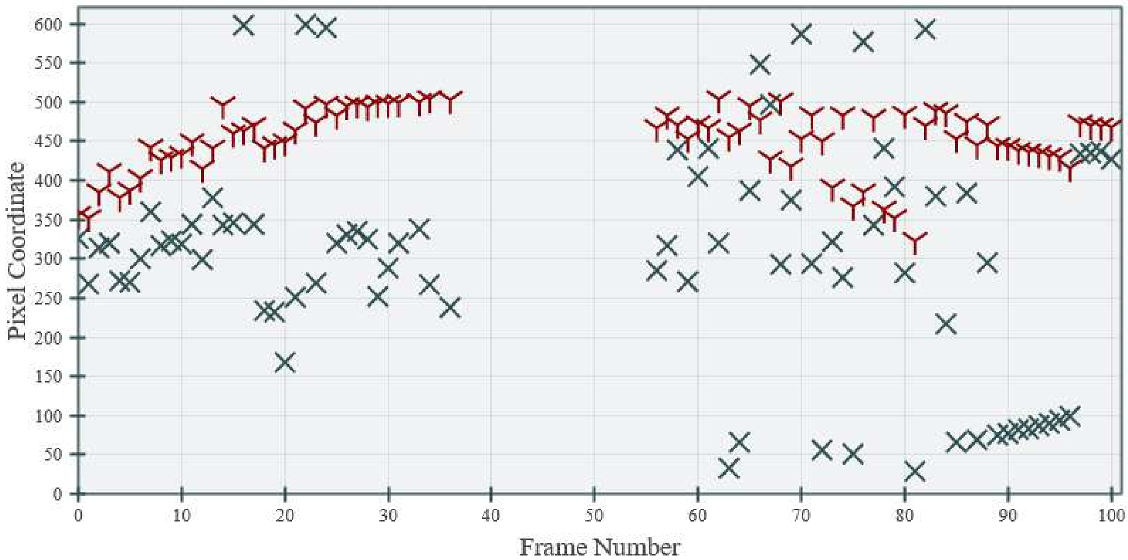
Choosing X

The optimal X could well be assumed to be a number that optimises the relationship between *false positives* and *false negatives*, where the latter arguably carries the heavier weight. Bear in mind, from the very start of this chapter, what the expectations of the local irregularity detector were: *[...] a lightweight module [that] analyse[s] each frame and mindlessly report[s] any deviations from the norm*.

Judging from fig. 5.17 especially, noting the hole that is doubtlessly present between frames 45 and 60 in (b), but not strikingly present as a large deviation from the Elbow in (a), one could very well argue that X should be set as low as 10 with a **cross-shaped kernel**, whereas other reports of holes in fig. 5.17a and fig. 5.15a could justify $X=50$. Considering **all the noisy reports** (especially in fig. 5.15), one

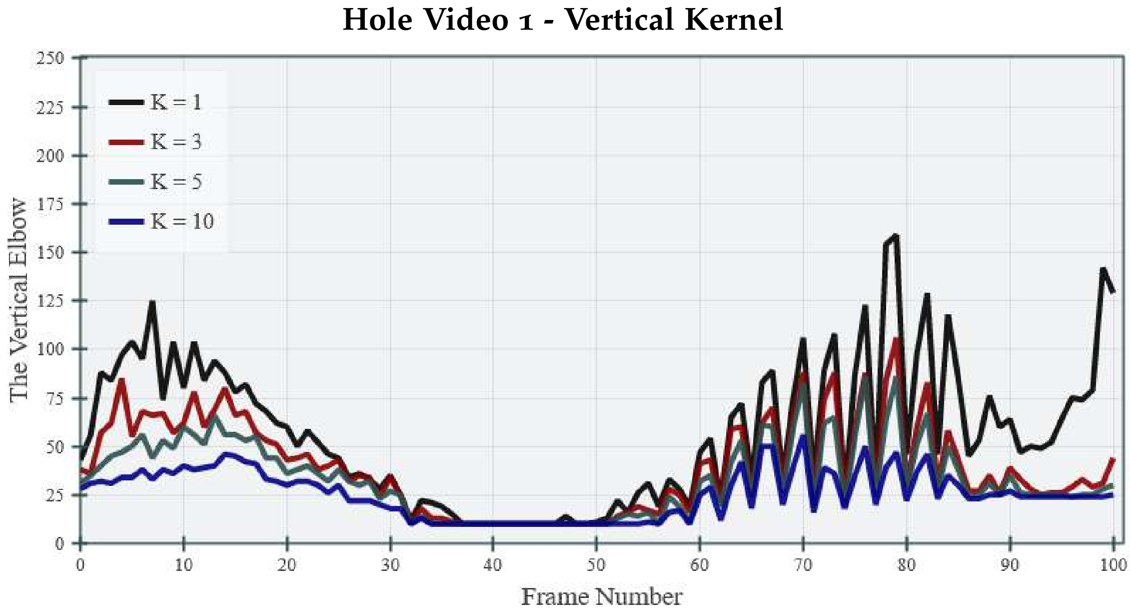


(a) Analysing the Elbow for 100 consecutive frames could reveal the typical – and the atypical – relationship between the tenth-, the fifth-, the third-, and the single largest piece of Background in each frame.

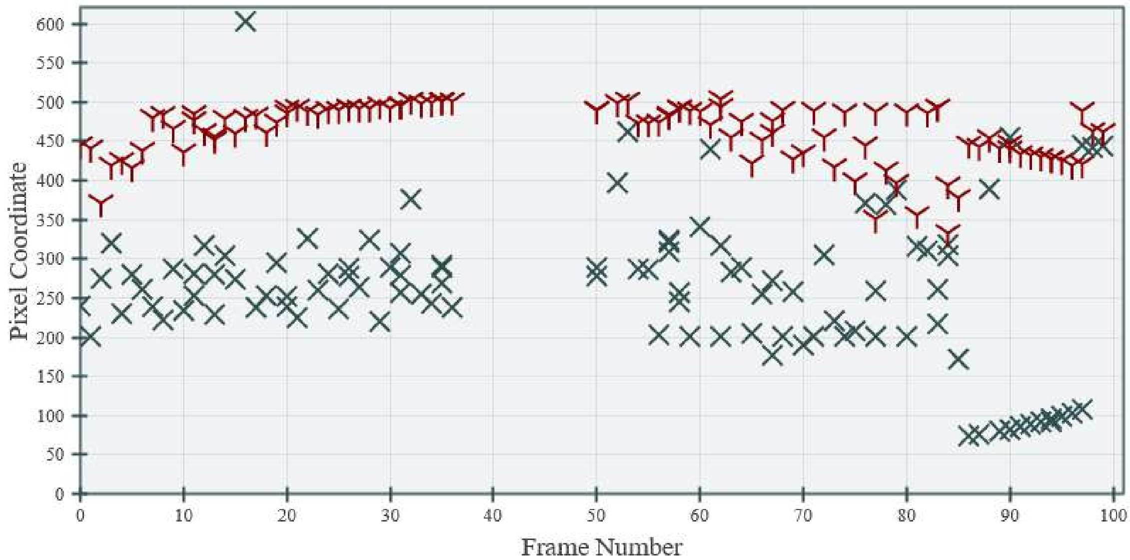


(b) Tracking the centre coordinate of the single largest piece of Background can help separating noise from real holes.

Figure 5.15: This specific video sequence reveals one single hole between frames **85 and 97**. Judging from (a), there is some fluctuation in K={3, 5, 10} in the former half of the time series, but relatively little in the latter, and especially little when the actual hole appears. However, K=1, reporting the single largest piece of Background, has an Elbow size typically 50% to 200% larger than the others. Even though K=1 yields responses in frames 0 through 30 which could be indicative of a hole, (b) shows no spatial continuity until frames 85 onward.

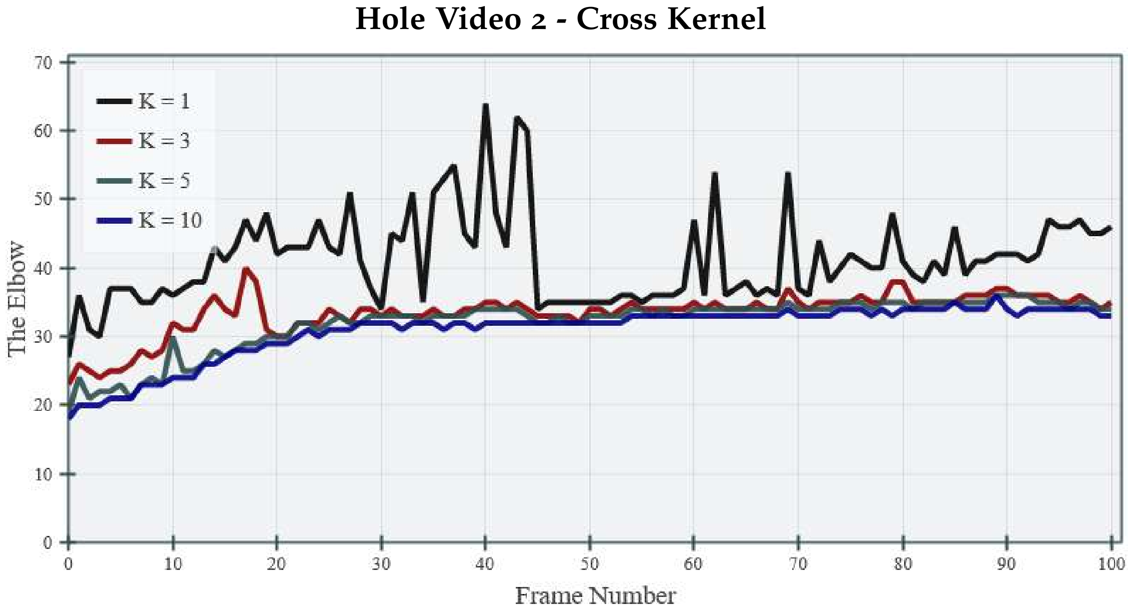


(a) The Elbow as deduced from flat vertical kernels is much larger than was the case with fig. 5.15a, but with a more pronounced divergence when the actual hole appears around frame 85.

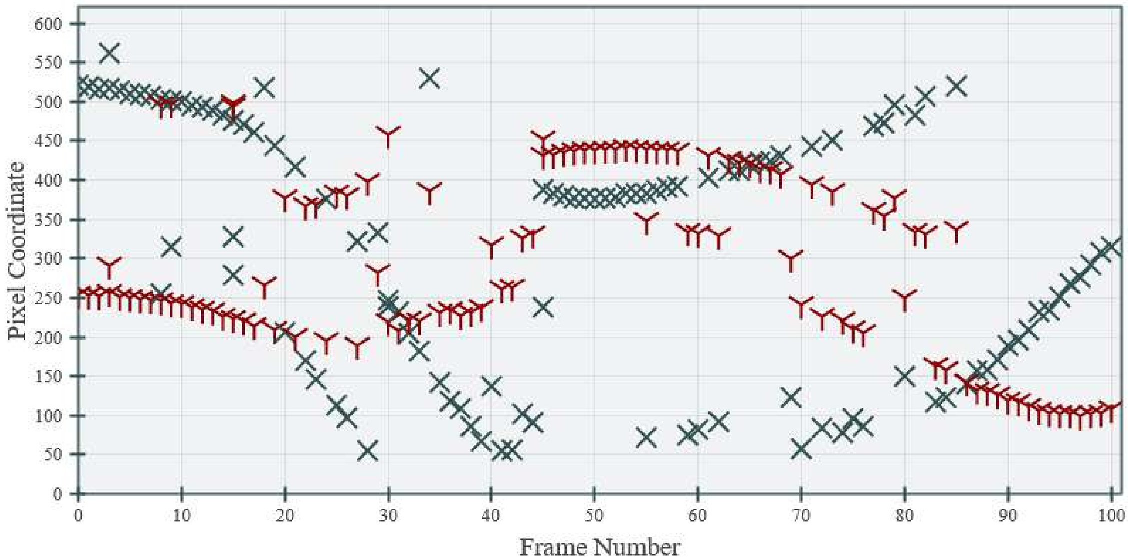


(b) The centre coordinate of the largest irregularity per frame stabilises distinctively around frame 85.

Figure 5.16: The single discoverable hole in Hole Video 1 is, judging from (b) as compared to fig. 5.15b, just as spatially confirmable whether the kernel is cross-shaped or flat. However, in order for a hole to be confirmable, it must first be discoverable, a trait enabled by the relationship between the Elbow and the hole. This relationship seems to manifest itself more clearly in (a) than in fig. 5.15a, suggesting the flat kernel is a better choice for irregularity detection in this situation.

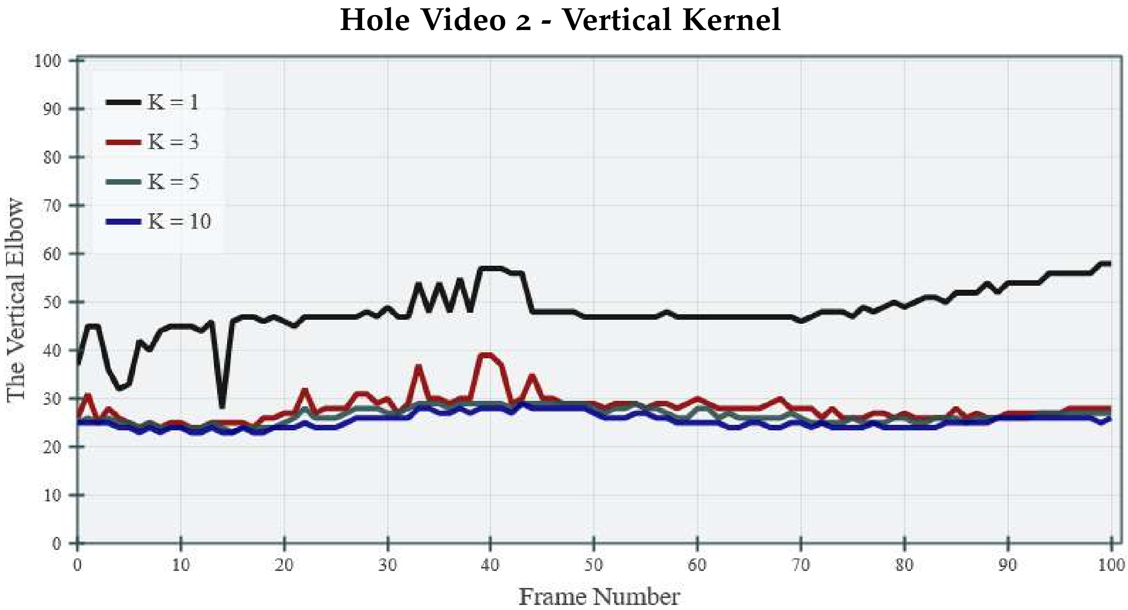


(a) Every instance of K=1 indicates, in this video, an actual hole. Interestingly, the size of the hole, relative to the typical piece of Background, fluctuates heavily throughout the first 100 frames of the video.

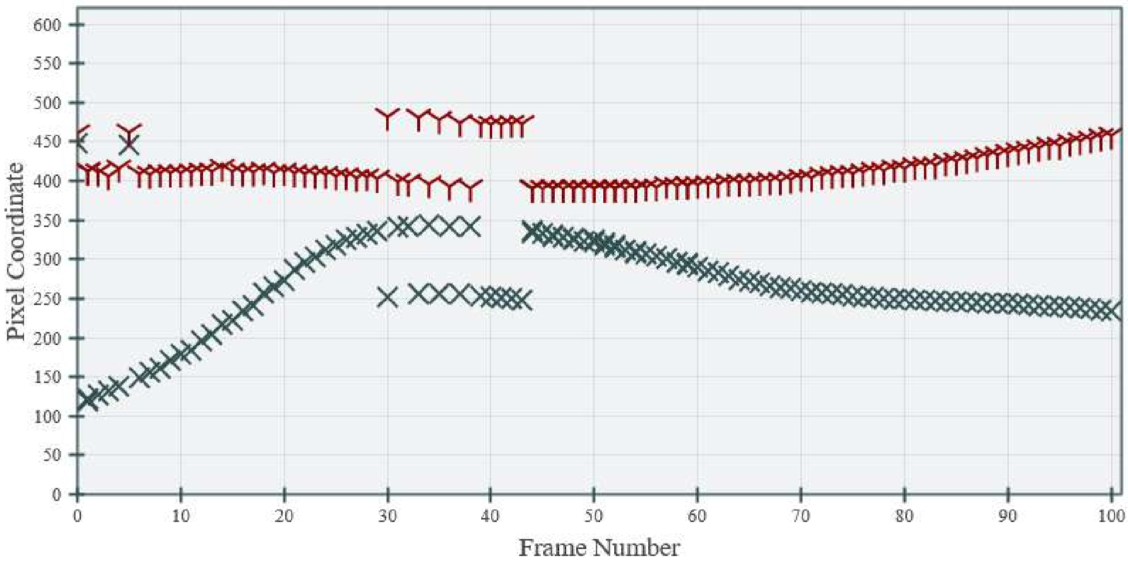


(b) Tracking the centre coordinate of the largest piece of Background reveals more temporal continuity than was the case with fig. 5.15b.

Figure 5.17: This video displays at most **three** holes simultaneously, which probably explains why K=5 and K=10 are extremely consistent whilst K=3 is relatively high in the first section of (a). Note that **every single K=1 is indicative of a true hole**. This is also quite apparent in (b), where there are more strong temporal patterns than was the case with fig. 5.15. The relationship between K=1 and K=5, for instance, in (a) shows that the distance is not always great between holes and regular pieces of Background. Between frames 45 and 60, the hole is less than 10% larger than the other pieces of Background, but (b) clearly shows a strong temporal continuity.



(a) The gap between the most pronounced piece of Background in the video compared to general pieces of Background is strikingly apparent when utilising a vertical kernel in Hole Video 2.



(b) The trajectory of the largest piece of Background, as identified by the vertical kernel, is impressively spatially consistent.

Figure 5.18: The vertical kernel yields an indisputable superior detection of the largest irregularity in Hole Video 2, as compared to fig. 5.17. In (a), the size of the single largest piece of Background is almost 100% larger than the third- and fifth largest piece of Background. More remarkably, perhaps, is the stability in K={3, 5, 10}, as opposed to the presumption that flat kernels would yield unstable responses.

could argue that **X should be set low and the temporal filter should be stricter**. The temporal continuity seems, at least from these observations, to be way more stable than the deviation in Elbow. Deviation in Elbow is nevertheless a necessary characteristic of true holes, but the size of the deviation can be assumed to be of less critical importance than the temporal filtering of local irregularity reports.

Interestingly, the **flat kernel** seems to offer an additional flexibility in terms of **choosing a larger X**. For instance, whereas the cross-shaped kernel showed sporadic tracking of holes in fig. 5.17 with as little as 10% difference in Elbow and the irregularity, the flat kernel offers under the same conditions impeccable tracking with a margin of at least 50%. Hence, the flat kernels will be favoured, with an X of 50, meaning irregularities will be signalised as areas that are 50% larger than the Elbow, allowing detection of smaller irregularities than were present in this case study, and avoiding the uncertainty of the 10% threshold offered by the cross-shaped kernel.

5.4 Tiles

A considerably large disadvantage to the Elbow approach is the assumption that a certain Elbow describes the typical Background size across the *global* frame. If the camera is not headed straight at the Net Structure, then pieces of Background closer to the camera will necessarily appear larger than pieces of Background farther away. One way to handle this problem, is to apply a *tiling* algorithm to the binary image, similarly to Paspalakis' [21] pixel-counting preparations.

fig. 5.19 shows how this tiling approach can detect irregularities in parts of the image that were previously excluded due to the relatively large pieces of Background closer to the camera. However, this approach could also trigger more false positives, especially so since Net Structure far away from the camera is typically represented in poorer resolution.

Arguably, one should make a set of overlapping tiles, but since the camera is in constant motion, it is assumed that irregularities will not reside too long on the unfortunate border between two tiles.

Additionally, this tiling approach is easily parallelised, since each tile can be assessed individually.

5.4.1 Building the Binary Image from Tiles

It was hinted at in section 5.1.4 that the adaptive thresholding algorithm could be assisted in choosing the optimal neighbourhood size based on a function of the local mesh size. A very good indicator of local mesh size is indeed the Elbow (with a cross-kernel) in a single tile. By performing an Elbow search within each tile, separately, one can obtain a precise description of the local neighbourhood.

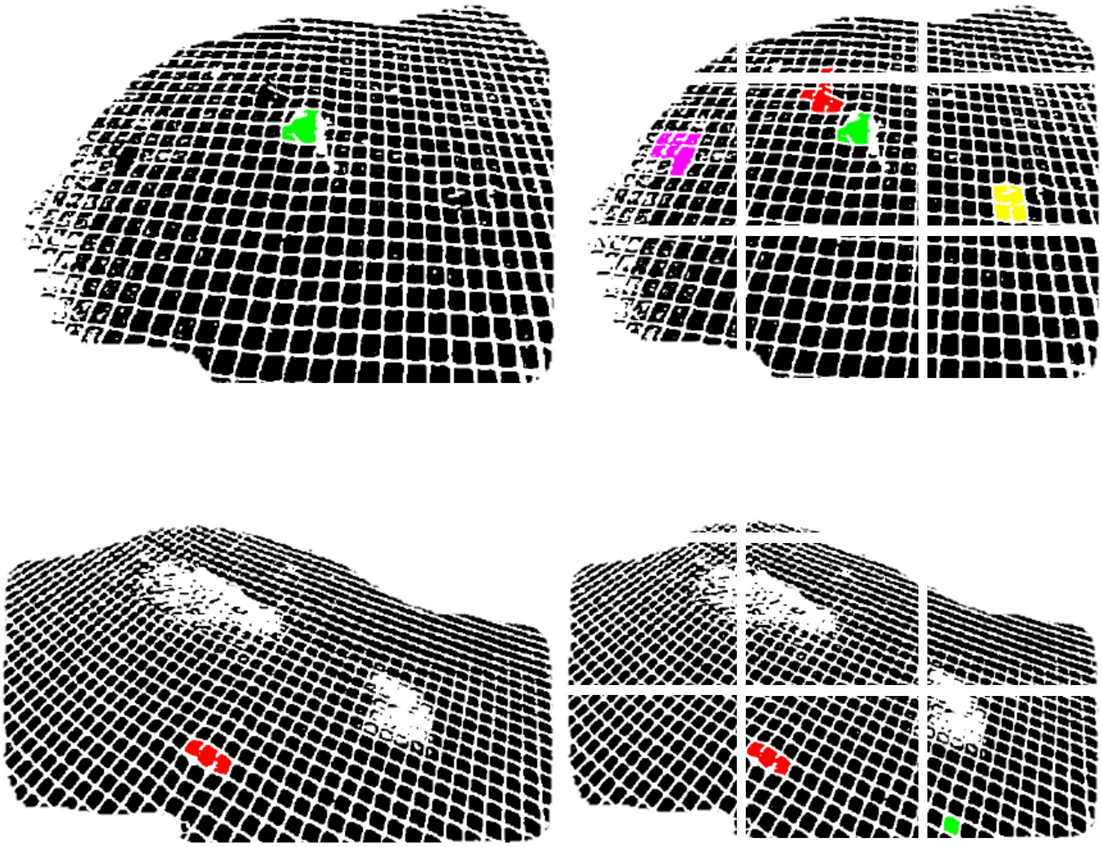


Figure 5.19: Dividing the binary image into tiles – tracking the Elbow and searching for irregularities within each tile –, individually, can increase recall if irregularities are found within neighbourhoods of small Background pieces relative to other parts of the image. Precision, however, will most likely also decrease as false positives tend to arise.

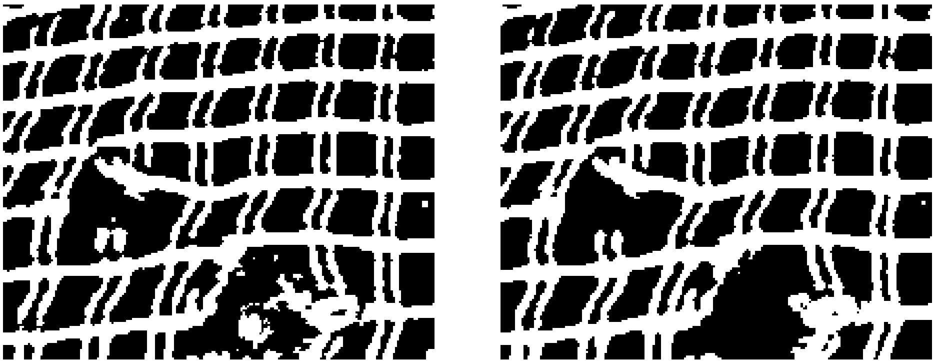


Figure 5.20: The leftmost image shows the result of an adaptive thresholding with a neighbourhood size twice the Elbow, and the rightmost six times the Elbow. Centre pixels in the hole cannot access Net Structure pixels to evaluate themselves against if the neighbourhood is too small, or the hole too big.

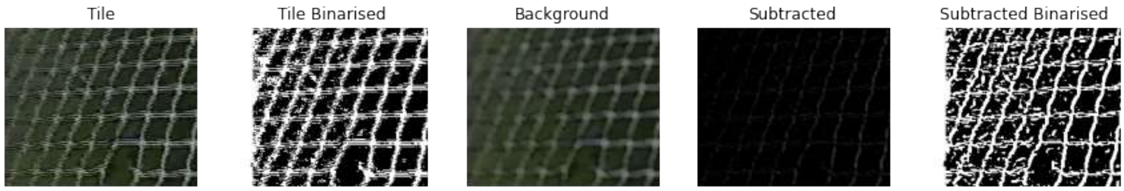
Keep in mind that the size of the neighbourhood evaluated by the adaptive thresholding algorithm needs be large enough such that pixels within holes can be correctly identified as Background. This implies that the pixel in the midst of a hole needs to be able to *see* actual Net Structure, and thus the size of the evaluated neighbourhood must be as large as the irregularities one want to be able to detect.

Each tile has in this work performed adaptive thresholding with a neighbouring size **six** times that of the Elbow in that tile, allowing (in principle) perfect binary representation of holes six times wider or longer that the typical mesh. It is worth mentioning, however, that holes larger than six times the typical mesh will be picked up by the irregularity detector, but the binary representation of that irregularity will have a white centre spot. This is illustrated in fig. 5.20, where the binary representation is imprecise when the neighbourhood is too small. Nevertheless, the area would most likely be recognised as an irregularity.

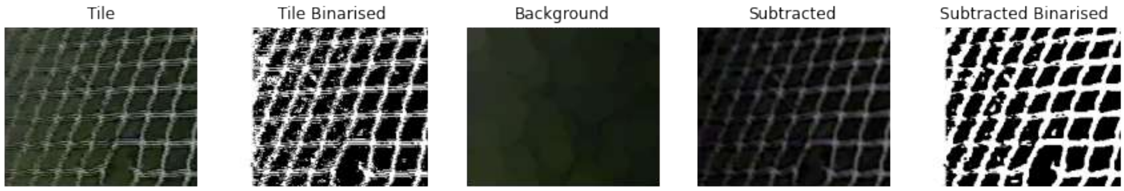
Keep in mind that these considerations are only applicable to the 3CAS paradigm, since NTS naturally builds the binary image without explicit information of tiling or the Elbow.

5.4.2 Denoising the Binary Image

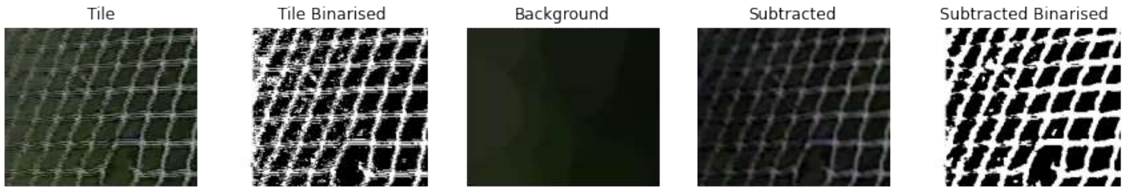
Although the adaptive thresholding algorithm effectively produces less noise the larger the neighbourhood, a few additional measures have been included to ensure optimal working conditions for the Local Irregularity Detector. These also take place on a tile-by-tile basis. Note that NTS segmentations have been left untouched – assuming that producing denoised binary images is a trait learnable by the U-Net if one **gives the U-Net ground truths free from noise even though the input image is noisy**.



(a) Opening with a kernel size 25% The Elbow includes the Net Structure in the Background, which can break true mesh connections in the binary image and cause false positive irregularity detection.



(b) Opening with a kernel size 150% The Elbow is a somewhat conservative approach which could eliminate some Background noise whilst preserving the integrity of the Net Structure representation.

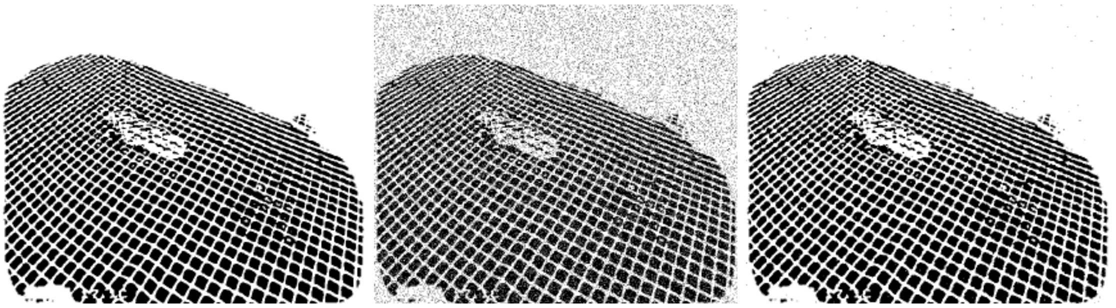


(c) Opening with a kernel size 500% is an even more conservative approach which yields similar results as 150%.

Figure 5.21: The Background subtraction approach effectively identifies high-level patterns in an RGB image. However, the level of pattern recognition must be carefully selected; if it is too small, then Net Structure will be identified as Background. By selecting a size that is significantly larger than The Elbow, it successfully denoises unclear edges and thus facilitates unambiguous binarisation.

The first measure is **Background subtraction** with morphological opening on the original RGB tile. This approach is described in fig. 5.21. Based on the identified Elbow, this algorithm can add an additional *edge* to the edge detection, by subtracting Background gradients from the original tile before edge detection takes place. The kernel of choice for Background removal is **disk-shaped**, with size 3x the Elbow, enabling a conservative filtering of images where only patterns larger than the meshes are removed.

The second measure is a simple **median filtering**, as proposed also by Haugene, on the binary image. Such filters are extremely efficient in removing salt-and-pepper noise, which can be detrimental to hole detection with morphological operations. Especially dense kernels, such as squares or disks (as utilised by Haugene), fail to *fit* in a hole if a single pixel within that is white. With flat- and cross-like kernels, one obtains greater robustness towards this. However, a simple median filter with a neighbourhood of **three** was utilised.



(a) Original binary image. (b) Salt-and-pepper noise. (c) Filtered noisy image.

Figure 5.22: The median filter with a neighbourhood of **three** is a very simple, yet effective, way of dealing with salt-and-pepper noise. In this case, it restores the original image more or less perfectly.

5.4.3 Building the Irregularity Space

Detecting irregularities within local areas of the frame could lead to double reports of irregularities localised on the border between two tiles. To finalise this section on tiles is introduced the concept of the irregularity space.

By creating a commonspace for irregularities where they are mapped according to their position in the original frame, one can seamlessly join irregularities found on the borders of the tiles. In fig. 5.23, tile 4 and 7 report the same irregularity, which overlap in the Irregularity Space.

Finally, the irregularity space of one frame can be stored efficiently as a set of (x,y)- coordinates describing the bounding boxes that surround each irregularity. This can be achieved by performing region growing on the irregularity space with each black pixel as a potential seed, and storing the coordinate extrema discovered for each region.

5.5 Introducing a Few Constants...

Even though this work set to eradicate as many heuristic constants as possible, a few have proven well worthy of implementation. The most crucial constants have been subject to investigation in this chapter, but some have not yet been explained.

For instance – it makes little sense to look for irregularities within a tile if the Elbow within that tile is as large, say, as the tile itself. A Maximum Elbow Size has therefore been implemented as 0.5x Tile Size. This is not to say that irregularities larger than half the tile cannot be found. This is to say that if the K-th largest piece of Background in that tile is as wide as half the tile, it is likely an indicator of very poor video quality, such as motion blur.

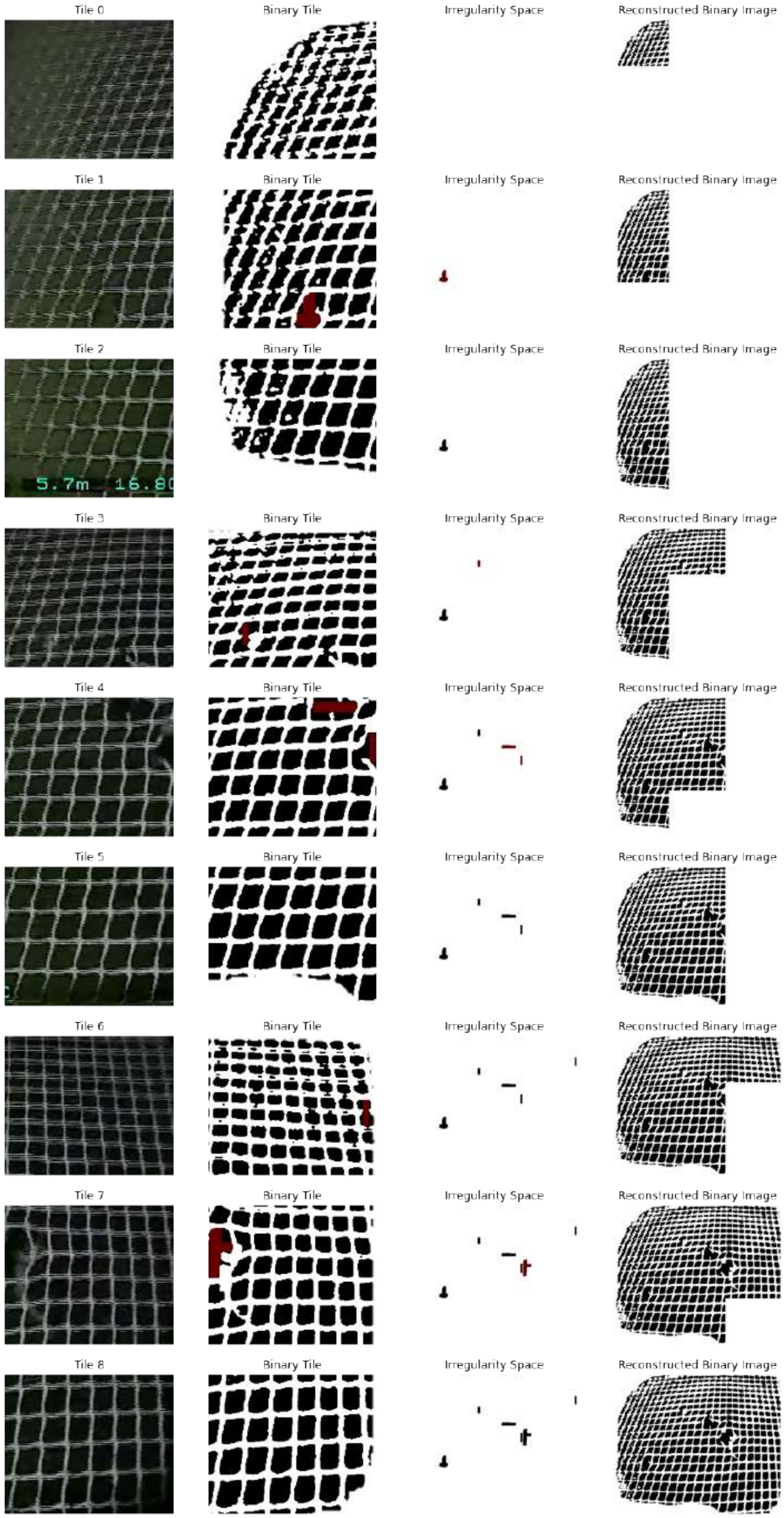


Figure 5.23: The irregularity space provides a commonspace for irregularities discovered within each tile.

Table 5.1: The local irregularity detector is constrained (or, **enabled**), by a small set of constants. Constants only applicable to the 3CAS system are written in **boldface**.

Name	Value
Local irregularity detector constants	
Elbow K	3
Elbow X	50
N Tiles	4x4
Min Elbow Size	11 px
Max Elbow Size	0.5x Tile Size
Min Kernel Width	8 px
Min Kernel Height	8 px
Median Filter Window	**3 px**
Background Removal Kernel Size	**3x Elbow**
Adaptive Threshold Neighbourhood	**6x Elbow**

Likewise, a Minimum Elbow Size has been implemented as 11 pixels. There is little evidence to suggest that 11 is a magical number, but it should be noted that, with X=50, an Elbow of 11 suggests irregularities must at least match a kernel of size 16. However, if the Elbow were 5, a 50% increase could imply 7 as an irregularity threshold, depending on rounding strategy. It seems fitting that some lower tolerance level should be set, and that there should be some gap between the Elbow and the irregularities.

A Minimum Kernel Width has been introduced alongside the Minimum Kernel Height to effectively eliminate the problem of false positives caused by single-row or single-column kernels working on binary images slightly misrepresenting reality. This threshold has been set to 8 pixels, meaning the smallest possible column kernel is in fact 8 columns wide, and 11 rows tall. This threshold has been set very low, consciously. Most false positives arise when there is a channel, an apparent broken connection, a single pixel or a few pixels wide. By requesting some small minimum breadth of such passage, one saves a tremendous amount of computational power in *not* investigating false positives which most likely are later filtered by the spatiotemporal filter (chapter 7).

All constants used in the Local Irregularity Detector are listed in table 5.1. Observant readers might notice that we have utilised a K of **three** instead of the previously considered **five**. This is because we have settled for a tiling approach where each frame is evaluated, locally, within the scope of 16 tiles. Since Elbow approximations at K=3 and K=5 were usually similar, the main drawback to choosing a lower K was previously the inability to discover more than two irregularities per frame. However, with a tiling approach, detecting more than two irregularities within a subspace one sixteenth of the image poses no longer a problem.

Chapter 6

The Scene Interpreter

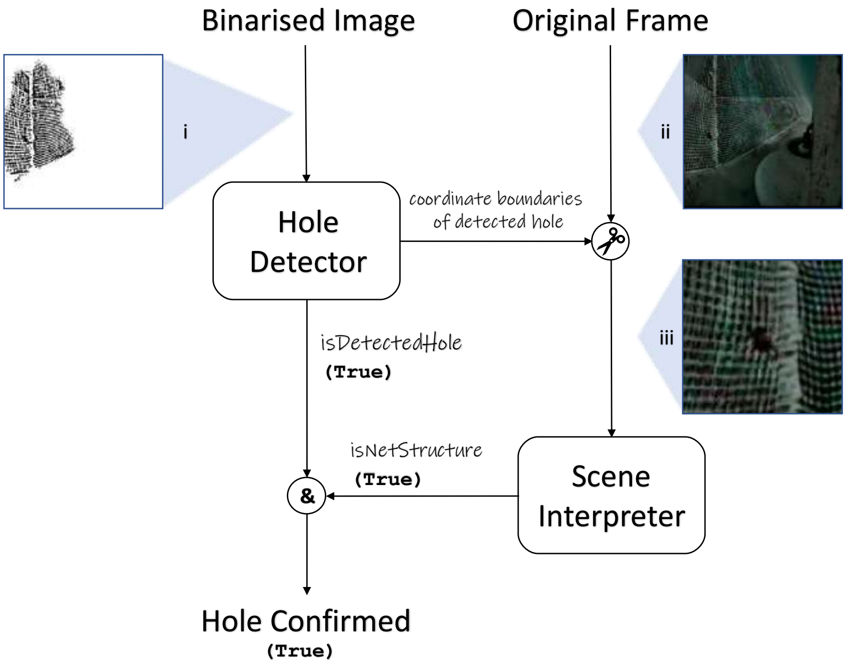
It could be argued that an efficient real-time-system should have some light-weight frame-to-frame analysis which is able to run in real-time (that is – significantly quicker than 30 frames per second if every frame is evaluated and the frame rate is 30) backed up by heavy-weight further analysis of *interesting* frames.

The very popular ORB-SLAM [59] solves the task of simultaneous localisation and mapping (SLAM) of an agent in an environment by having different *threads* solve different tasks – placing different time demands on each thread. One thread runs in real-time and extracts from each frame *ORB-descriptors* and calculating the agent's position in the local map. Building the map from the information in the video stream is a heavier task not suitable for the real-time thread. Instead – a second thread is triggered at a lower rate to optimise the local map if new territories have been detected, or if enough time has passed. Likewise – a third thread optimises the *global* map at an even lower rate.

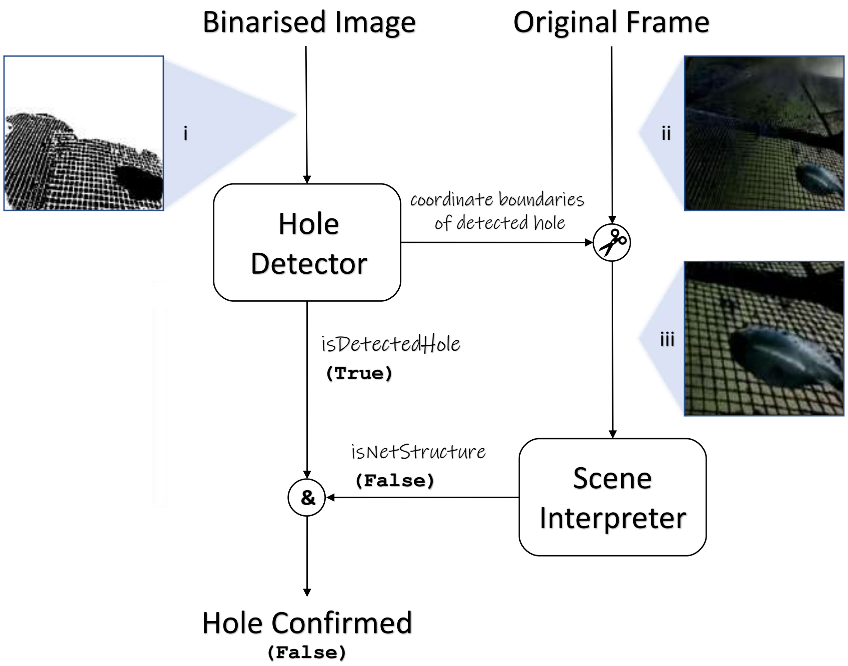
Building on the philosophy of ORB-SLAM, a scene interpreter module is suggested to reject false hole reports by the hole detection module. Assuming that holes rarely occur, it seems reasonable have hole reports pass a second test before they are confirmed. Due to the lack of sufficient images of holes – and the variety of their appearance – it is suggested to train a classifier to identify whether or not a scene depicts Fish, Nonsense, or Net, rather than have it explicitly tell if it depicts a hole. If an irregularity is reported, then the area of the frame containing this irregularity will be sent to the interpreter for assessment, and accepted as a hole only if the interpreter is confident that the image depicts Net. This idea is illustrated in fig. 6.1.

6.1 Constructing interpreter Data

Two videos from the training foundation were utilised to extract training and validation data for this task; MANTA2 and MANTA3. 300 images were constructed from parts of various frames from the videos, 150 images from each. One third of the images were of Fish, one third of Net, and one third of Nonsense, usually turbulent water columns or unintelligible Net. 20% of the data was used for



(a) If the local area of the discovered irregularity is classified as *Net*, the hole discovery prevails.



(b) If the irregularity discovery is caused by a fish falsely included in the binarised image, the scene interpreter will reject it.

Figure 6.1: If the irregularity detector finds a suspiciously large black area in the binarised image (i), then the area surrounding the hole is selected from the original frame (ii) and the resulting snip (iii) is sent to the scene interpreter. The interpreter classifies the image as either Net, Fish, or Nonsense, the former of which is the only interpretation that confirms the hole.

Figure 6.2: The data constructed for the interpreter consisted of 100 Fish images (first row), 100 Nonsense images (second row), and 100 Net images (last row). The images vary in size to increase prediction robustness. The idea is that valid holes can only be found within Net, and that Net obstructed by fish does not constitute a hole.

validation – and 80% for training. K-Fold Cross Validation was not utilised at this point, since an effective estimation of the performance of the idea was more interesting than a precise fine-tuning.

The sizes of these images originally varied significantly (196±67 x 218±80 pixels) but were resized to 128 x 128 pixels as part of preprocessing. The general idea is that a suspicious area will be cropped from the original frame with some padding, before resized and sent to the classifier. Having various sizes during training will add an additional aspect ratio robustness to the model and serve as a form of regularisation. Example images are shown in fig. 6.2.

All three RGB channels were utilised, and pixel intensity values were normalised (0.0 - 1.0 instead of 0 - 255). Each picture was accompanied with a label encoded as an array; [0, 0, 1] if the image depicted Nonsense, [0, 1, 0] for Net, and [1, 0, 0] for Fish. Hence, a model output of [0.2, 0.3, 0.5] means the model is 20% confident the image depicts fish, 30% confident it depicts Net, and 50% confident it depicts Nonsense. The label with the highest confidence is favoured as prediction.

Table 6.1: The simple model showed signs of learning during 600 epochs of training, but identified fewer than half the Fish images in the validation set. Circa three quarters of the Net images were correctly identified, and two thirds of the Nonsense images.

Class	Precision	Recall	F1-Score	Support
VALIDATION SCORES AFTER 600 EPOCHS				
Fish	0.50	0.42	0.46	19
Net	0.61	0.77	0.68	22
Nonsense	0.76	0.65	0.70	20

6.2 Model Architectures

A few different neural network architectures were investigated and compared during this research. All networks were heavily based on convolutional layers, but varied significantly in complexity. The art of constructing such nets is indeed part art, part craftsmanship, and designing them from scratch by oneself is not necessarily a worthwhile endeavour. The three architectures have therefore been borrowed from, or inspired by, the official documentation of Keras [60], Tensorflow's [61] high-level API.

6.2.1 Simple Model

The first model investigated was rather simple in it's nature – consisting of 3 convolutional layers each followed by MaxPooling layers – and a dropout layer for regularisation. The architecture is visualised in appendix B.1.

The simple model was trained for 600 epochs before training was terminated. The results provoked no resounding *hooray*; fig. 6.3 shows how loss decreased steadily but extremely slowly, and validation accuracy stabilised at approximately 65%, which could signalise that the model constantly misclassifies one of the three classes.

Other scores in table 6.1 reveal how the model struggles to identify Fish images in particular, but neither class is predicted with sufficient confidence. The low precision scores indicate that the model is rarely very confident in its predictions, but more so when it suggests an image to be Nonsense. The fact that Net has higher recall than precision means the model suggests Net to be true more often than is the case, such that 77% of the existing Net images have been recognised, whilst the model's prediction *this is Net* has only been true 61% of the time. This makes the hole rejection conservative to some degree, which is a favourable trait. One would rather have the model report one hole that is not than have it overlook an actual hole. In any case, these scores are too low to be trusted.

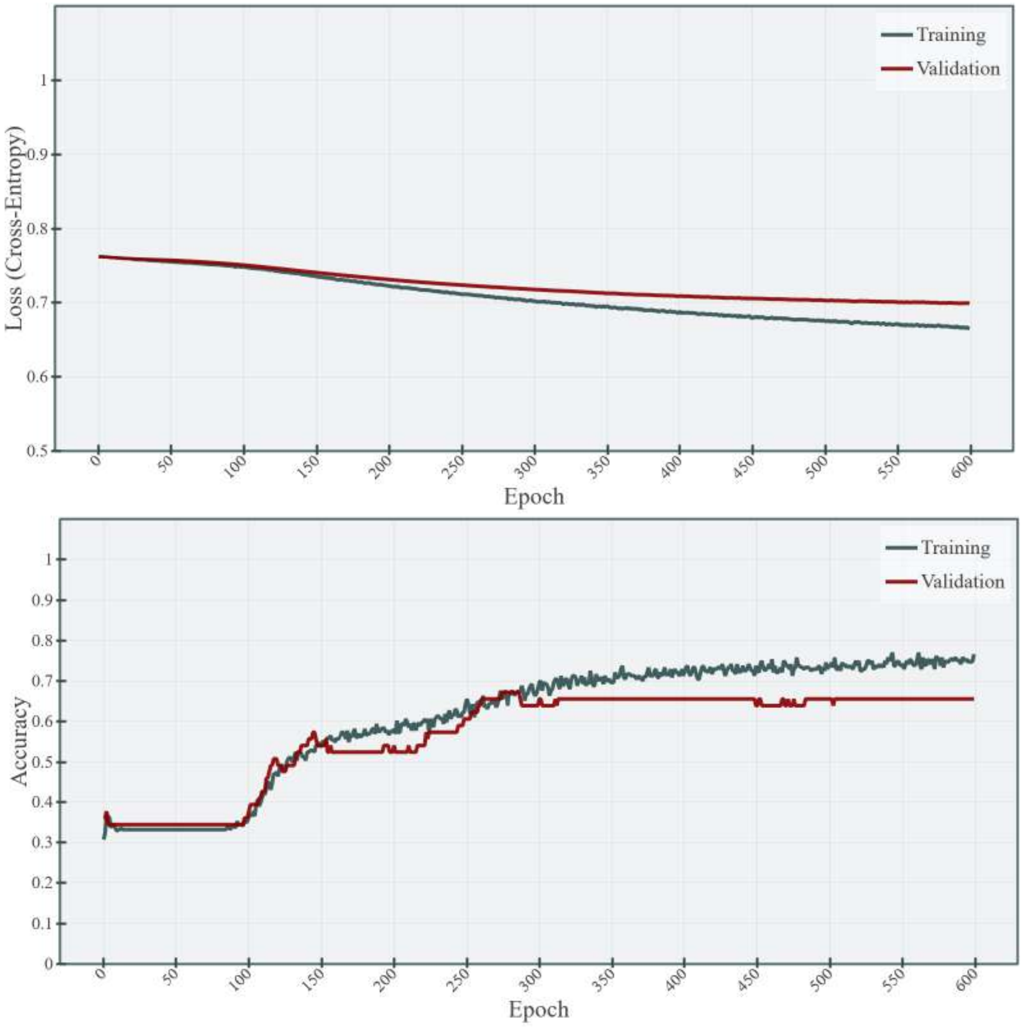


Figure 6.3: The loss of the simple model steadily decreased over 600 epochs, signalising learning, though training and validation curves diverged significantly at about 150 epochs. This could signalise overfitting. This is also reflected in the accuracy plot where the two curves follow one another up to about 150 epochs. After 300 epochs, the validation accuracy flattens at about 0.65 whilst the training accuracy climbs marginally over the next 300 epochs.

Table 6.2: The complex model showed no signs of learning during 100 epochs of training. At validation after the final epoch, it guessed Nonsense at every image, failing to ever step into the unknown and have a go at the other classes.

Class	Precision	Recall	F1-Score	Support
VALIDATION SCORES AFTER 100 EPOCHS				
Fish	0.00	0.00	0.00	20
Net	0.00	0.00	0.00	22
Nonsense	0.31	1.00	0.47	19

6.2.2 Complex Model

Sometimes low scores reflect the need for a more complex model. Especially so if the training scores remain low, typically signalising either lack of detectable patterns between the images and their labels – or a model that fails to detect such patterns. A second attempt was made with a Keras example model,[1] designed to classify cats from dogs, with significantly higher complexity: 15 convolutional layers, MaxPooling, skip-connections, and dropout for regularisation. The architecture is visualised in appendix B.2.

This model would have optimally been training for 600 epochs as well, but after 100 epochs of no progress but sky-rocketing validation loss, almost perfect training accuracy, and dismal validation accuracy, it was time to terminate the process. It is obvious from the scores in fig. 6.4 that the complexity of the model is way beyond the required for this task. At least considering the amount and diversity of available data. If the simple model is deemed unreliable, the complex model is indeed a complete disaster.

6.2.3 Something in-between: VGG16 and Transfer Learning

With an underperforming simple model, and a severely overfitting complex model, a natural next step would probably be to find something in-between. Simonyan and Zisserman's VGG16 [43] is one such model. Their CNN trained for weeks with four NVIDIA Titan Black GPUs to achieve state-of-the-art performance in the ImageNet 2014 Challenge.[2]

In short, their model consists of 13 convolutional layers (though much less intricately weaved together than were the layers in the complex model: compare appendix B.3 and appendix B.2), MaxPooling Layers, but no dropout. More importantly for this work; the authors noted that their pre-trained model worked as a brilliant generalised base model for different datasets.

[1] https://keras.io/examples/vision/image_classification_from_scratch/
[2] http://www.image-net.org/challenges/LSVRC/

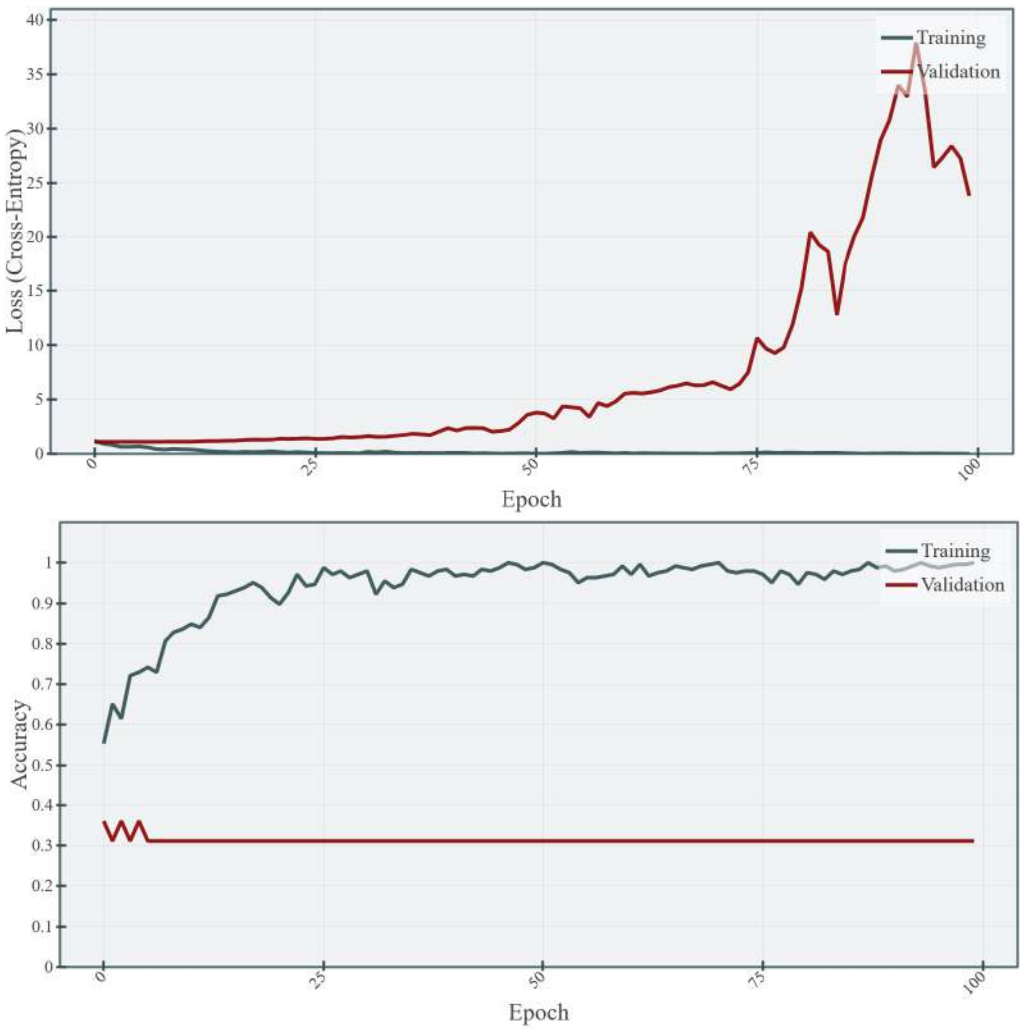


Figure 6.4: These score boards show quite brilliantly the potential downside to complex models. The model is complex enough as to capture the information in the training data almost perfectly, but manages not to generalise this knowledge to the validation data. As a result, the model has specialised in the training data, knowing exactly what features to extract to fit that, but these features are way too fine-grained to fit unseen data. When validation loss diverges from training loss it is usually time to put an end to training, and it is safe to say training could have been terminated well before 100 epochs.

Fine-Tuning the VGG16 Net

Having open access to Simonyan and Zisserman's model and its pre-trained weights, it is feasible to remove the top layer of their network (that which classified the 1000 ImageNet classes) and swap it for a new layer suited for three classes, namely Fish, Net, and Nonsense, and to train it for a few epochs on this data with very low learning rate. In addition to the new top layer, a dropout-layer was added for regularisation. The base model architecture and its extension to fit this new application is available in appendix B.3 and section B.3.1.

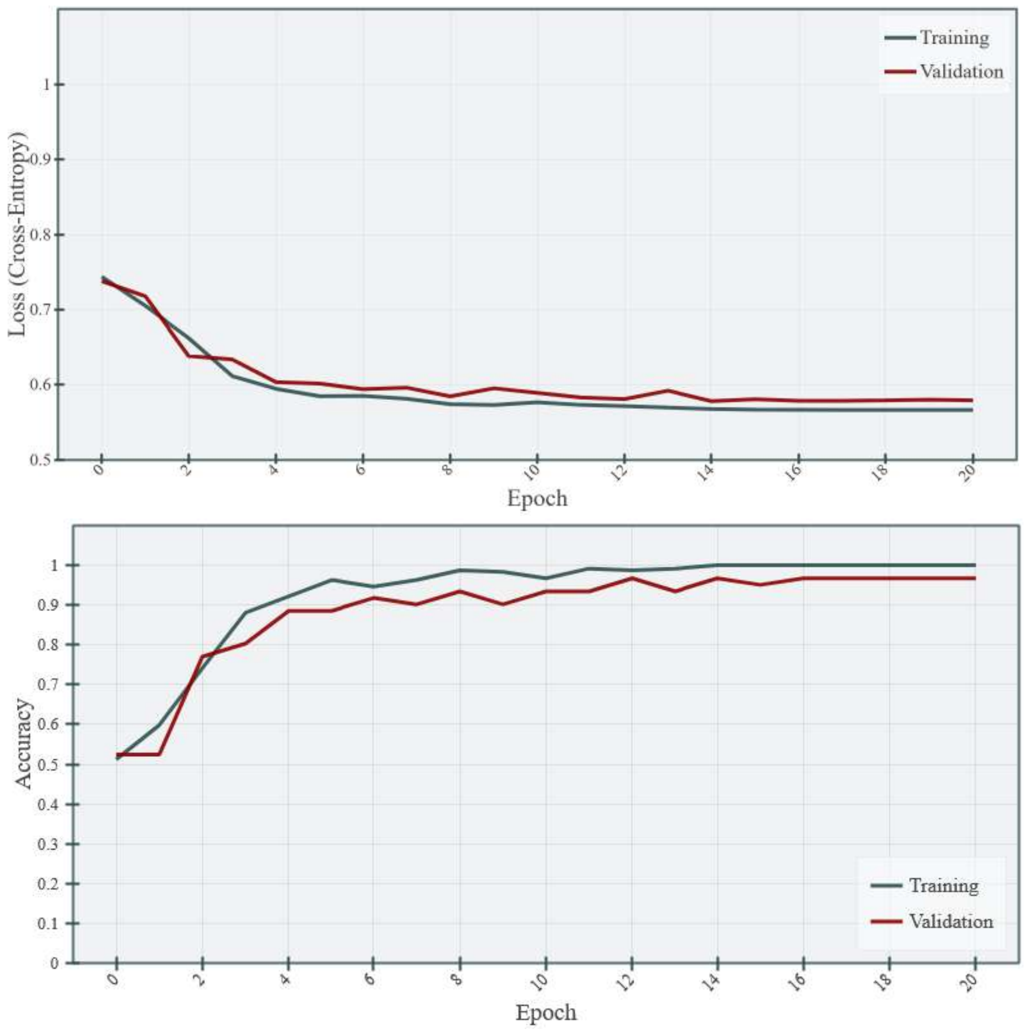


Figure 6.5: With transfer learning, both training and validation accuracy broke old records within two epochs of training. Validation loss decreased rather steadily at least for 6 epochs, but as training accuracy reached 100% no more information could be learned from the training data.

After a dozen of epochs of training with a learning rate of 0.00001, the accuracy of the validation data peaked at 98% (see fig. 6.5). The rapid convergence towards 100% training accuracy could indicate the model was too complex, at least given the modest size of the dataset. However, since validation accuracy reached uplifting heights, it could be an indication that the net generalises well to this specific task, but could benefit from more data. It was at this point decided to extend the dataset using *Data Augmentation*.

Data Augmentation

The problem of little training data – leading to the model grasping its entire complexity and specialising in the exact data which it is provided – generally leads to worse performance on validation data, or data which is otherwise slightly

Table 6.3: With transfer learning the validation scores reached very satisfactory levels after few epochs of training. The Net class has recall 100% which means no image of Net was mistaken for anything else but Net. This is very important in terms of not confusing images of holes for Fish or Nonsense. Additionally, the Fish precision of 100% indicates the model was correct every time it proposed Fish as the true class of an image. This could indicate the model has picked up indistinguishable Fish-features.

Class	Precision	Recall	F1-Score	Support
VALIDATION SCORES AFTER 20 EPOCHS				
Fish	1.00	0.95	0.97	19
Net	0.92	1.00	0.96	22
Nonsense	0.95	0.90	0.92	20

different from the training data. A strategy to counter this without the expensive labour of labelling thousands of new images is to generate duplicates of the already labelled data, albeit with small (or larger) perturbations.

Since the validation accuracy is already extremely high (98%), an important goal of the augmentation process is to delay the convergence of the training accuracy at 100% whilst upholding a solid validation score. Delaying this convergence indicates that the model has been exposed to *more* examples during training, and therefore be more robust. Whether or not validation accuracy reaches 100% is in a sense a secondary issue. Had the original score been lower, say, below 90%, its improvement would be a much more interesting watch.

With the help of a **random rotation generator** samples of training data were rotated in the interval (-180°, 180°) each epoch. This introduced randomness ensured the training data changed from epoch to epoch, a new-posed challenge for the neural net to overcome. Still, it managed to obtain steady 100% accuracy on the training data after approximately 35 epochs, which is circa 20 epochs later than it did with no rotation. The validation accuracy maxed, again, at 98%. Examples of such rotation operation is visualised in fig. 6.6.

Adding **translative perturbations** (see fig. 6.7) in addition to the aforementioned rotations had the model train for 100 epochs before cracking the mapping between training input and labels completely for the first time. However, it needed about 200 epochs before stably converging at 100% training accuracy, and validation accuracy hit 98%, again.

Having observed a clear pattern of delayed convergence, though no increasement (nor decreasement) in accuracy, it was decided to add several other perturbation generators: in addition to **rotational** and **translative** operations, random **zoom**, **horizontal** and **vertical flip**, and **shear transformations** were added. The specifics regarding ranges are given in table 6.4. With all augmentations, training

Figure 6.6: The training data was extended by applying random rotation perturbations. The leftmost image is the original image of a fish, followed by duplicates with rotation perturbations.

Figure 6.7: A second generator added translative perturbations to the training data. The leftmost image is the original image of a fish, followed by duplicates with horizontal and vertical translative perturbations.

score did not show signs of stabilising at 100% during the first 300 epochs. After 100 epochs, both training and validation accuracies surpassed 90%, and spent the next 200 epochs fluctuating between approximately 90% and 98%. Loss simultaneously stabilised for both at just below 0.6, which is about the same as did the loss without augmentation (fig. 6.5).

98% accuracy on 61 validation images means 59.78 images were correctly classified. This is perfectly feasible since the model reports its *confidence* in the different classes. If it suggests an image to be 90% Fish, and 10% Net, and the true class is Fish, it is only 90% correct. However, since the performance is *so* good on the validation data for either model (trained on augmented data or not) telling which one is the better is not trivial. To enable discrimination between the performances, a relatively large test dataset was created from a disjoint pool of video material. 300 images (as large as the training foundation) was extracted from different video material, some filmed by inspection robots and some by the Manta. The approach was the following:

1. Create large test dataset

2. Train model with augmented data, validate on test dataset

3. Train model without augmented data, validate on test dataset

4. Keep the best-performing model from the best-performing epoch

Table 6.4: Random perturbations were added to the training dataset during training to introduce the model to variations of the modest available dataset.

Transformation Type	Random Range
DATA AUGMENTATION GENERATORS	
Zoom	[50%, 110%]
Rotation	[-180°, 180°]
Vertical Flip	[True, False]
Horizontal Flip	[True, False]
Width Translation	[-33%, 33%]
Height Translation	[-33%, 33%]
Shear Transformation	[0%, 20%]

5. Compare the two best-performing models

An observative critic could notice that *we do validate on test data* and that *we do not use K-Fold cross-validation*. The reason for both is that we consider the validation data large enough such that it represents a variety of scenes and challenges.

K-Fold cross-validation is often utilised not to wound up in a situation where the validation dataset is unrepresentable of the general data in terms of complexity – either too easy (yielding unrealistically high validation scores) or too hard (yielding unrealistically low validation scores). This technique is also often used if one wants to get an idea of what number of epochs is optimal, such that one can train a model on the *entire* data foundation afterwards. Given this tactic's aim to create a test set at least as large as the training dataset, it would make sense to want to make use of this new data for training as well. After all, creating a large pool of training data and only using half to train the model seems, at first sight, like a waste. However, it is not necessarily obvious what constitutes the best tactic: the larger the validation dataset – the more confidence can be put in a *keep-best-performing-model* mindset. A set of model weights which perform well on a large dataset should, after all, be quite good on a large range of data. If one chooses to include *all* the data during training, and training in a K-fold manner, one can only approximate which number of epochs constitutes the optimal – this can never be verified unless one creates a *new* test dataset, in which case one finds oneself in a never-ending cycle.

Thus, creating an equally large test dataset, and using this for validation, was chosen to achieve trustworthy model weights.

Data Augmentation Evaluation

The model trained without data augmentation reached its highest validation score after 22 epochs. Bear in mind that this model required only 12 epochs to reach its high score when training on 240 images and validating on 60 (fig. 6.5).

Table 6.5: The models trained with and without data augmentation performed very similarly on the large test dataset. However, the augmentation model generally got the better of the other, marginally. If one assumes that every prediction was made with 100 % confidence, the augmented model retrieved three more images of Fish (89 vs 86), two more images of Net (91 vs 89) and one less image of Nonsense (86 vs 87).

Class	Precision	Recall	F1-Score	Support
AUGMENTATION SCORE | NO AUGMENTATION SCORE				
Fish	**0.84** | 0.83	**0.89** | 0.86	**0.86** | 0.84	100
Net	**0.91** | 0.89	**0.91** | 0.89	**0.91** | 0.89	100
Nonsense	0.91 | 0.91	0.86 | **0.87**	0.89 | 0.89	100

Including these 60 images in the training foundation obviously induced the need for more epochs. The augmented model, on the other hand, reached its high score after 219 epochs, almost ten times the number of epochs required by the non-augmented model.

These numbers reflect well the need for more patience, or, equivalently, more computing power, when increasing the number of training instances. However, table 6.5 reveals that patience (or computing power) pays off in results. The augmented model performs a bit better on two out of three classes, and matches the performance of the other on Nonsense class (if comparing F1-scores). These results might seem trivial. Since the (relatively) large test dataset is still *only* 300 images large, then two percentages different scores on a single class still only represents two misclassified images. This could be due to happenstance. But, arguably, judging from these results, data augmentation certainly does not seem to *degrade* the quality of the model. Conversely, it *seems* to give the model an extra edge which would definitely be cherished if present. The implementation of such augmentation is not expensive per se, so if patience or computational power can be afforded during training, it should indeed be included.

The test dataset, accompanied by the predictions of the augmented and non-augmented model, is listed in its entirety in appendix C.

Chapter 7

Irregularity Tracking

If an existing irregularity has been detected in one frame, chances are it will appear in several subsequent frames as well. Ideally – one single irregularity should only be reported once, and subsequent discoveries of that irregularity should be mapped to the already discovered and tagged instance. This motivates the investigation of irregularity tracking, a way of uniquely identifying the individual irregularities and assigning to them a unique ID. Another equally important motivation for this investigation, is to enable filtering of irregularities that appear momentarily in one frame only to disappear in the next. By assuming true irregularities to have spatiotemporal continuity, to be observable in several subsequent frames, one might effectively reduce the impact of bad frames or momentary bad segmentation.

7.1 A Spatiotemporal Irregularity Filter

One measure for continuity could be requiring *irregularities to be observable in approximately the same area* over subsequent frames, hence encapsulating the *spatial* component (a true irregularity is of approximately the same size at the same location as previous registrations) and the *temporal* (overlaps are calculated based on past discoveries of that irregularity). An implementation of this idea is to draw a bounding box around a reported irregularity, and to compare the overlap between that box and recently drawn boxes (see fig. 7.1). This method does not compare the content of these boxes, but assumes that the content is the same if they lie very close in time. An already utilised score to evaluate such overlap is the Jaccard Index, the *intersection over union*. By using this score, instead of a mere intersection score, one can punish reports not only for lack of overlap but also for size differences even if they overlap. This is desirable, since true irregularities do not change significantly in size from one frame to the next.

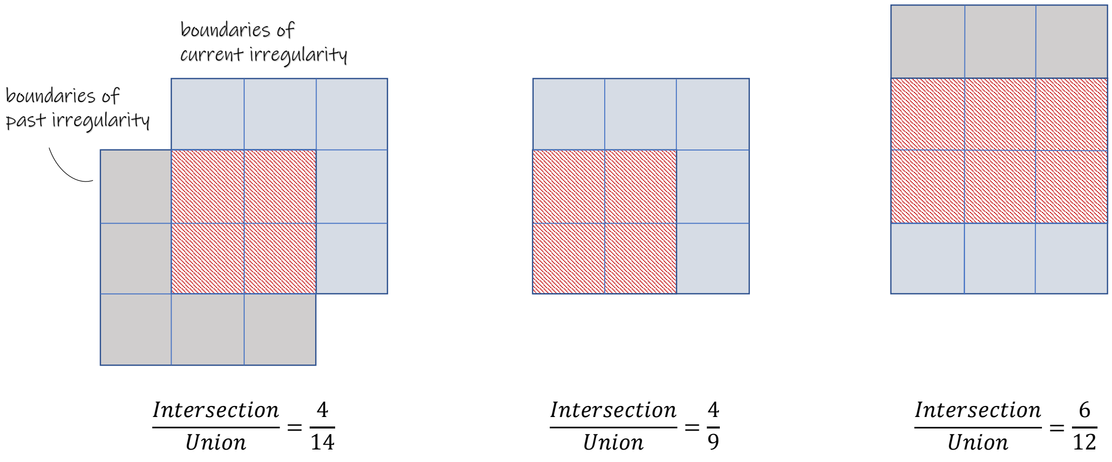


Figure 7.1: Requiring substantial intersection relative to union between coordinate boundaries of current and past irregularities makes an effective spatiotemporal irregularity filter. The score punishes both over- and undersegmentation, and hence disallows the matching of irregularities who overlap completely, but differ significantly in size.

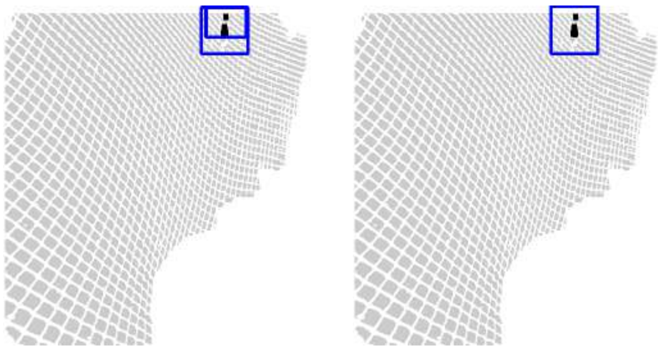
7.1.1 Padding, Conjoinment, and Thresholds

Bearing in mind that the quickest cleaning robots of the lot (the Manta and FNC) reach speeds of 1 m/s and capture video at 30 fps, the scene shifts roughly 3 cm per frame, depending of course on angle, and very small holes could therefore be missed if boxes are drawn too tightly.

It is proposed in this work to expand the bounding boxes some. By expanding the boundaries, equally, in each direction, one automatically achieves greater Jaccard indices (see fig. 7.3). This also means that nearby detected irregularities can be matched even though they in fact are separate. However, due to the mentioned problem of motion, implementing some padding proved to increase the tracking of true irregularities significantly. Ideally – few enough local irregularities should exist so that continuous, false, simultaneous, reports with close proximity is quite uncommon.

A few padding strategies have been developed. Practically speaking, one could either (i) *use little padding and require little overlap from frame to frame* or (ii) *use larger padding and require significant overlap from frame to frame*. This work utilised first the latter approach, but settled with the first, expanding the dimensions of the irregularities by 25 pixels in each direction and requiring a Jaccard index of 15% between past and current irregularities to validate a match. This is, in a sense, respecting the principle of Occam's razor[1]; the strategies might yield similar results, but (i) is simpler. One could also argue that larger padding yields larger boundaries, which in turn require more computational power to have their respective overlap calculated.

[1] https://en.wikipedia.org/wiki/Occam%27s_razor



(a) (b)

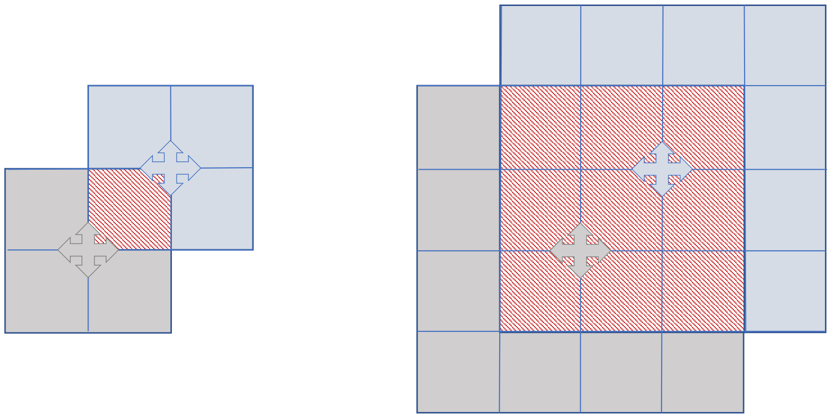
Figure 7.2: The irregularity space sometimes misrepresents a single irregularity as two or more disjoint irregularities. A conjoinment procedure is suggested to merge irregularities of close proximity.

Immediate spatial proximity between two local irregularities (in the very same frame) could be the result of sporadic awkward behaviour by the morphological operation, coarse resolution in the binary image, or a combination of the two. For example, in fig. 7.2a, a curious fish has been detected as two slightly disjoint irregularities. A proposal is to compare the *overlap* (not the Jaccard index) of irregularities in the irregularity space in a recursive manner; if two irregularities overlap at least 70% (meaning at least 70% of either irregularity overlaps with the other) the two form a new irregularity whose boundaries are the old boundaries' extrema. The vote count from the irregularity with the most votes propagates (votes will be introduced shortly). The recursive proposal means that the newly formed super-irregularity can absorb a *third* nearby irregularity, and so on.

7.1.2 Accumulating Votes

Requiring a single match with the previous frame filters those odd discoveries in single frames, but fails to identify irregularities that occur in every other frame (for instance because of varying segmentation quality) and for assurance one might want to have an irregularity appear more than twice before verifying it.

The problem of appearing and disappearing irregularities can be solved by implementing a sliding window – a short-time memory that contains irregularity reports within several previous frames. By enlarging this window one allows for longer intervals between the irregularities. This has certain disadvantages. For instance, matching two reports based on what part of the frame that irregularity has been detected makes little sense if the ROV has moved to a completely different scene. This matching strategy relies entirely on the assumed closeness in time between the past and current frame. In this project the size this window has been set to 3, giving the algorithm the opportunity to look at three past frames and their irregularity reports. Building on previous logic on ROV pace, the scene should maximally have changed by approximately 13 cm from frame three steps back.



$$\frac{Intersection}{Union} = \frac{1}{7} \qquad \frac{Intersection}{Union} = \frac{9}{23}$$

Figure 7.3: By padding the detected irregularity area, effectively expanding the boundary boxes, one achieves a greater score in terms of intersection over union. The padding therefore enables overlap detection even if irregularities have moved slightly.

Having a short-time memory of three could at first sight seem to imply a maximum match threshold of three – requiring our current irregularity to match with every report in the memory – and requiring the irregularity to actually appear in every frame. Another issue with this approach is that, as the ROV moves, the chances of overlapping with with that irregularity several frames back in time decreases. A proposal is rather to store each report within the memory with a *number of accumulated votes*. In this manner, our current irregularity can inherit the accumulated votes of a matching irregularity in the memory, and add one to the count. Thus – the size of the memory allows an irregularity not to appear in a couple of frames – and the accumulation and inheritance of votes allows an independently set vote threshold for verification. A downside to setting higher threshold means the hole will be verified at a later point in time if the algorithm is used in a real-time application. However – if implemented not in real-time, the verification tag can simply back-propagate to all previously matching frames once the threshold has been met and the verification tag assigned to the irregularity.

Again, there is a trade-off between proneness to false positive reports and recall. Requiring fewer votes will inevitably lead to more false alarms, whereas too high a threshold will fail to report holes which only appear for a short moment. The count should be able to filter the false local irregularity reports arising from corrupted image stream whilst moving at operating speed, and perhaps should one not demand of the system to identify very obscure holes appearing under such conditions. A threshold of 7 has been observed to disallow the verification of the typical corrupted reports whilst allowing the detection (hypothetically) if an irregularity in clear view for less than one-fourth of a second.

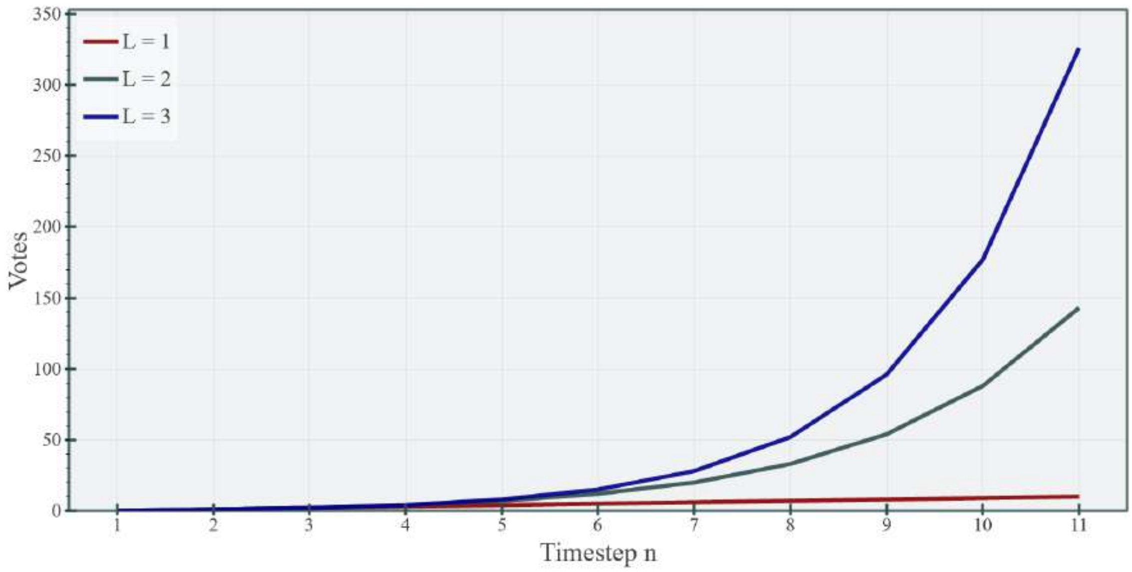


Figure 7.4: The accumulation of votes can accelerate in a Fibonacci-like manner as described in eq. (7.1) if previous instances of an irregularity are not cleared as they are re-discovered in new frames. Depending on buffer size, L, slow camera movements can cause a new discovery to overlap with several old reports of that discovery in the buffer, and effectively count the same votes more than once.

Beware of Accelerating Accumulation

It is, with this logic, important to clear any past version of an irregularity in the memory once its votes have been inherited in a new frame. If not – votes for irregularities detected every single frame will accumulate in a Fibonacci-like fashion, whose acceleration depends on short-time memory buffer size. This undesired effect is visualised in fig. 7.4 and described by eq. (7.1). One can observe from fig. 7.4 that the effect is not as large if the required threshold of votes ν to verify an irregularity is very low. However, there is clearly a troublesome difference between the exponential growth of any curve with an available memory buffer (here visualised with $L = 2$ and $L = 3$) and the linear progression of the curve with no memory ($L = 1$ depicts the desired vote accumulation – but gives not the desired benefit of not having to observe the irregularity in each and every frame). The proposed solution is therefore to look for matching irregularities starting at the most recent frame, and to only let the most recent match prevail. In this manner, old reports will eventually fade out and votes will only be counted once.

$$\nu[n] = \begin{cases} 0, & \text{for } n = 1, \\ 1 + \sum_{m=1}^{n-1} \nu[m], & \text{for } 1 < n \leqslant L, \\ 1 + \sum_{m=n-L}^{n-1} \nu[m], & \text{for } n > L \end{cases} \tag{7.1}$$

$$n, L \in \mathbb{N}$$

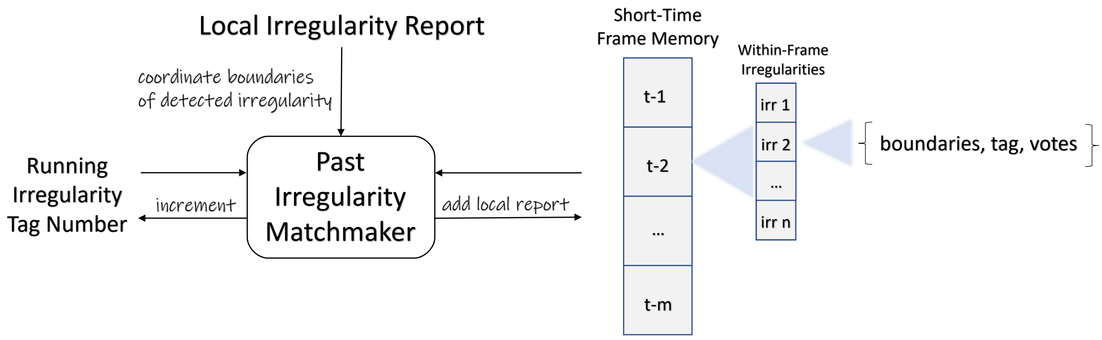


Figure 7.5: The past irregularity matchmaker compares the overlap between previous irregularities and the current. If a match is made, the current irregularity inherits the vote count and the tag number of the previous. If the tag is anything but zero, it means the previous irregularity is verified and thus the current can abort its vote accumulation mission. If no tag is inherited but a vote threshold is met, the irregularity receives a running irregularity tag number before it is stored in the short-time memory.

7.1.3 The Running Irregularity Tag Number

To enable grouping of matching irregularities, a running tag number is proposed to be assigned to irregularities that receive enough votes to breach the verification threshold. Once an irregularity has received such number, it can be stored with that tag in its irregularity report in the short-time memory. If later reports match with that irregularity, it acknowledges a match with a verified irregularity and inherits its tag without bothering with further vote accumulation. Unverified irregularities are simply stored with tag number 0.

In this manner, one can group matching irregularities once they have received verification, and since new matching reports are stored with the same tag but new boundaries, the location of the tagged irregularity can be tracked over subsequent frames. This complete module which interacts with the short-time memory and stores the running tag number has been named the *past irregularity matchmaker* and is visualised in fig. 7.5.

7.2 Integrating the Scene Interpreter

After receiving a verification tag it seems fitting to investigate the irregularity further. Perhaps is this where the scene interpreter (chapter 6) should come into play (after an irregularity is verified, not after its first appearance). Recall from that chapter how the task of the interpreter is to add a *smart* dimension to the hole verification – separating reports of holes from reports of fish – or simply reports of noisy water columns. Although the U-Net segmentation is supposed to ignore such, they are sometimes included in the segmentation. One reason is the spatiotemporal encouragement using lag masks. A fish that rapidly disturbs the centre of a segmentation mask might therefore be included rather that breaking up the mask.

From Irregularity to Hole

Whereas the matchmaker module simply requires irregularities to appear in a continuous manner, it does not necessarily know what that irregularity is. It is therefore consciously referred to as an *irregularity*, but now is the time to talk about *classified irregularities*, namely Holes, Fish, or Nonsense. These classes are best handled by their own module called the irregularity librarian. The librarian keeps track of a register which maps tags to classes, and sends unclassified tags to the scene interpreter for classification.

Although fig. 7.6 shows the tag: class register as a simple key -> class look-up, it was successfully extended to achieve greater robustness. One could imagine that the scene interpreter misclassified the first picture of an irregularity, and it would be a pity if this classification were to sustain itself. The tag: class register is therefore suggested to contain *class votes* such that the irregularity be classified multiple frames in a row before its nature is verified. This work suggests a threshold of three, meaning an irregularity is classified every frame it is observed until one class has three more votes than the second most popular class. In this manner we do at least three independent classifications of each irregularity before confirming its class.

However, it is suggested that the irregularity is highlighted from the moment it is verified, although awaiting class confirmation. In this work the irregularity is presented as the current *leading* vote at all times. For instance – if an irregularity is first recognised as Fish, it should be visualised as Fish in the video in the current frame. If the irregularity is observed in the next frame and identified as Hole, it should change visualisation to Hole. If the next frame thereafter identifies it as Fish, then Fish leads the vote count and the irregularity is visualised accordingly. Still, the irregularity should be classified for every subsequent frame until one class leads by three votes.

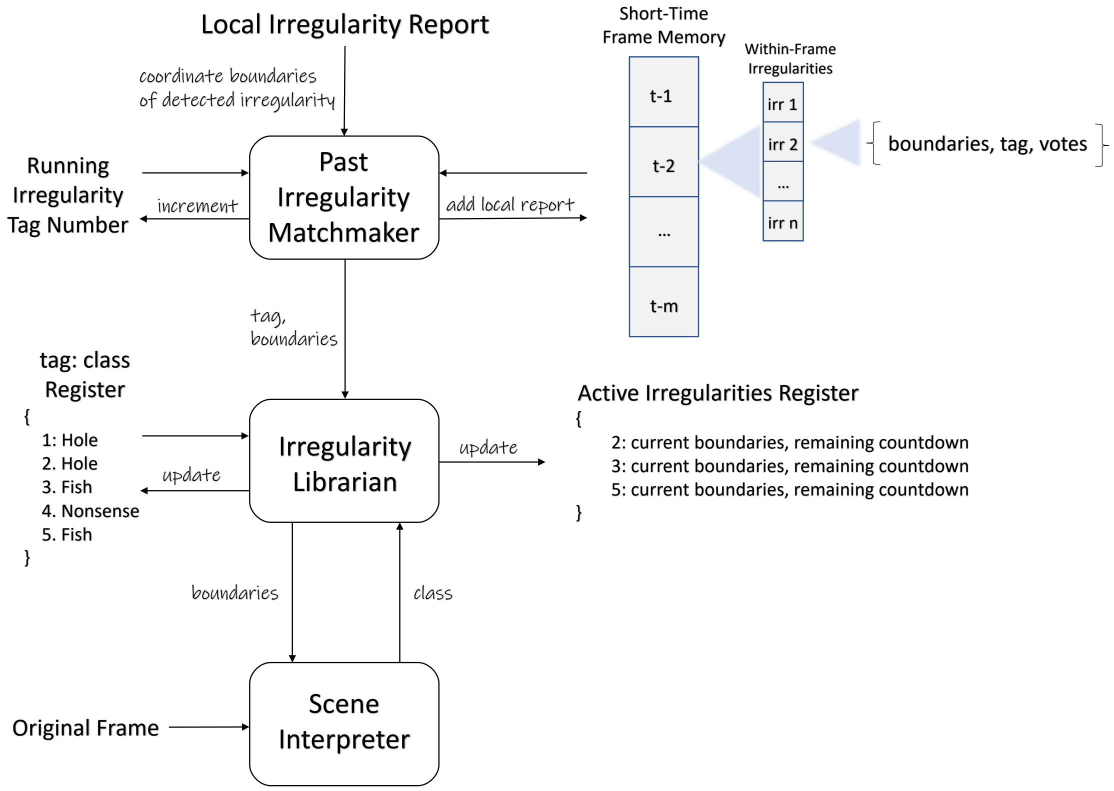


Figure 7.6: The matchmaker (described in fig. 7.5) coupled with the irregularity librarian makes sure a verified irregularity is linked with a class. The scene interpreter (see chapter 6) classifies new irregularities as either Fish, Net, or Nonsense, and assuming holes may only occur within Net, this is the only class which signifies trouble. Detected irregularities which require highlighting are stored in the active irregularities register, preferably Holes (but Fish could also be highlighted). A countdown accompanies the irregularities such that the highlighting can last a fraction longer than the irregularity is visible.

Active Irregularities

A second register handled by the librarian contains irregularities visible in the current frame. This register is called the *active irregularities register*. If an already visible irregularity has been discovered in the current frame, its boundaries are updated such that the new location of the irregularity is saved. In addition, a countdown timer is proposed to accompany its report in this register. This to give the irregularity the chance to be highlighted for a couple of frames after its disappearance, and give the observer the chance to catch a glimpse of the irregularity even though it only appeared in the fraction of a second. The complete spatiotemporal irregularity filter – with its classification extension, can be seen in fig. 7.6.

7.3 Guided Tracking

The naïve system for irregularity verification requires indeed little advanced technology given a very important precondition; the irregularity needs be discoverable by several frames, independently. This precondition seems to be favourable when one wishes to establish the presence of a true irregularity. However, it could be argued that, once an irregularity has been verified, that it should be actively tracked. If – for instance – poor segmentation leads to an irregularity no longer being picked up by the local irregularity detector. This happened in fig. 7.7, where the assumption *an irregularity is X% larger than the k-th largest piece of background in the segmentation* did not stay true due to shadows included in the segmentation. When the irregularity fell out of short-time memory, it had to be re-verified, and thus, receive a new irregularity tag when re-appearing several frames later. Such cases motivate the implementation of a guided tracking algorithm to actively track the trajectory of verified irregularities.

7.3.1 Projected Movement

One technique might be an indirect tracking of the irregularity, namely tracking the motion of its bounding box and project that onto subsequent frames in which the irregularity is not detected, but hypothetically still exists. In this manner one could continuously move the highlighting rectangle in the same direction the irregularity has been moving either until the irregularity, hypothetically, leaves the scene, or until some countdown expires. This could be interpreted as an extension of the already implemented countdown module which highlights the area in which an irregularity was latest seen for some time. In addition to highlighting that area for some time, one could also change the position of the highlighted area. In addition – by updating the hypothetical position of an irregularity in the current frame – one can increase the chances of achieving sufficient overlap when the irregularity re-appears – ensuring the irregularity receives one and only one irregularity tag.

Centre Tracking

Calculating the movement of the rectangle is done by calculating the relative distance, δ, between the centre of the irregularity, (x^c, y^c), from one frame to the next:

$$\begin{aligned} \delta_t^{x^c} &= x_t^c - x_{t-1}^c \\ \delta_t^{y^c} &= y_t^c - y_{t-1}^c \end{aligned} \tag{7.2}$$

If one wishes to project the irregularity onto a frame in which it is not directly detected, one *could* simply utilise *current movement*, the relative distance between the previous instance of that irregularity and the one before that. However, this strategy tends to destabilise if the camera movement jerks or in any other way behaves nonlinearly. Instead, it is proposed to accumulate every detected

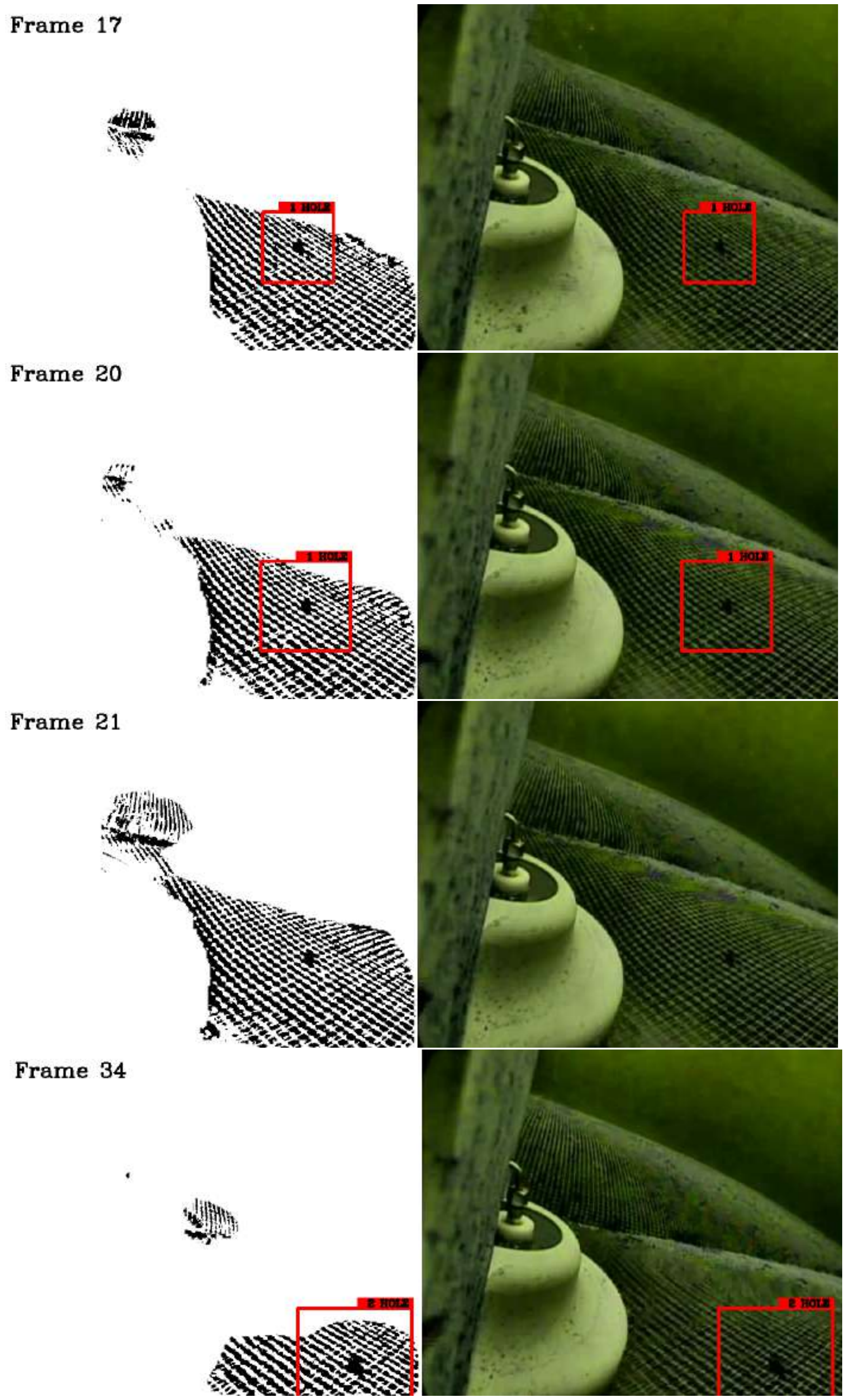


Figure 7.7: The probable hole with irregularity tag 1 is tracked from the 17^{th} until the 20^{th} frame, before segmentation causes it not to be identified in frame 21. When the irregularity is yet again verified – in frame 34 – it has received a new tag.

movement of a single irregularity, and to calculate the projected movement at t in time $\hat{\delta}_t$ as the median value of some number of previously detected movements for that irregularity. If t^* is the last K ts in which it was detected then this can be written as

$$\begin{aligned} \hat{\delta}_t^{x^c} &= median(\delta_{t^*}^{x^c}) \\ \hat{\delta}_t^{y^c} &= median(\delta_{t^*}^{y^c}) \end{aligned} \tag{7.3}$$

By utilising the median value of previous movement as projected movement, one can counter outliers which may or may not be occur from one frame to another based on segmentation quality. An implementation of the median movement scheme in the case study of fig. 7.7 is shown in fig. 7.8b and fig. 7.9. Notice in fig. 7.8 the difference between the instability of the tracking when the movement hypothesis is based on current movement (fig. 7.8a) rather than the median approach (fig. 7.8b).

Detected boundaries of an irregularity's bounding box are effectively stored as two coordinates, $(x^{min}, y^{max}), (x^{max}, y^{min})$, and the projected boundaries $(\hat{x}, \hat{y})$ of an irregularity not discovered in this frame, but hypothetically present, can thus be deduced by adding the projected movement delta to the previous boundaries:

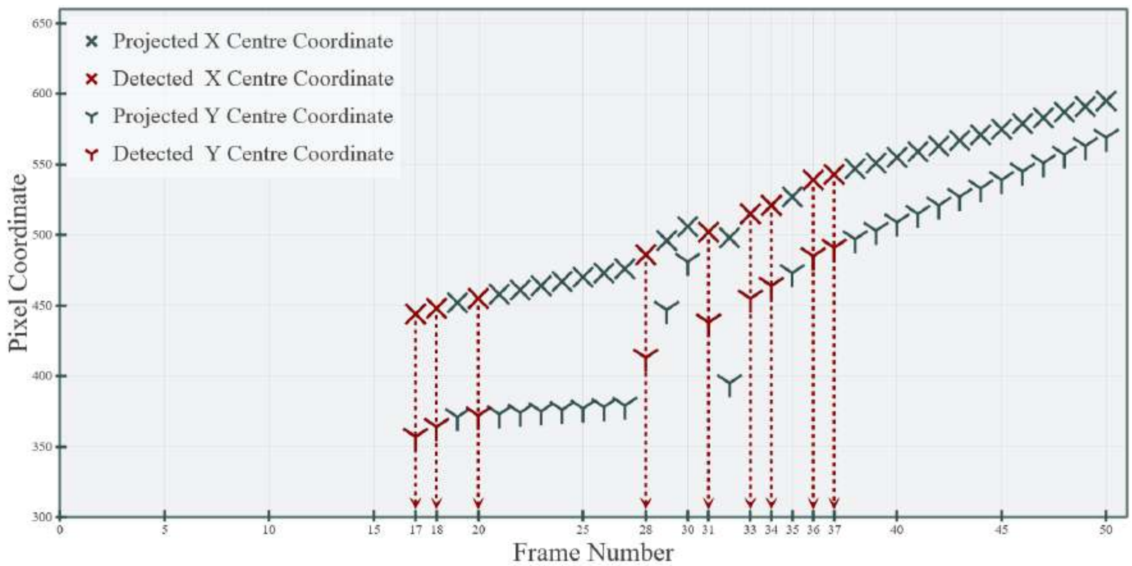
$$\begin{aligned} \hat{x}_t^{min} &= x_{t-1}^{min} + \hat{\delta}_t^{x^c} \\ \hat{x}_t^{max} &= x_{t-1}^{max} + \hat{\delta}_t^{x^c} \\ \hat{y}_t^{min} &= y_{t-1}^{min} + \hat{\delta}_t^{y^c} \\ \hat{y}_t^{min} &= y_{t-1}^{min} + \hat{\delta}_t^{y^c} \end{aligned} \tag{7.4}$$

Notice that the previous boundaries need not be directly detected in the previous frame, in which case x_{t-1}^{min} equals $\hat{x}_{t-1}^{min}$ et cetera. Nevertheless the median projection $\hat{\delta}$ is *exclusively* drawn from detected movements, and not from the accumulation of projected movements.

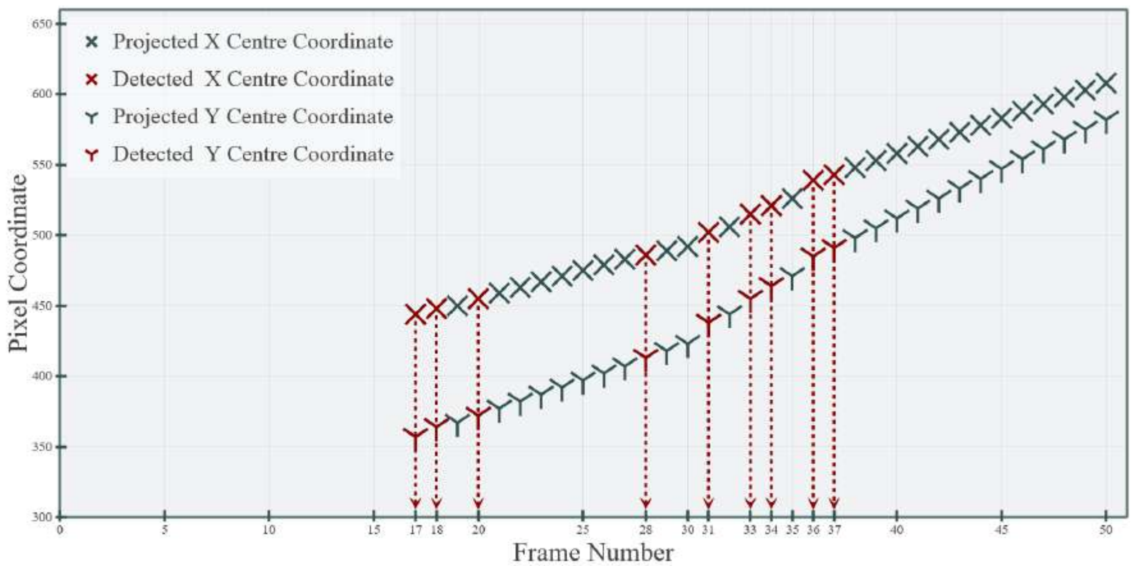
These lines of reasoning produced a handsome tracking of the case present in fig. 7.7, and fig. 7.9 shows how the algorithm now follows the irregularity from entrance to extrance. Even better – since the boundaries are continuously updated in the active irregularities register, it keeps its irregularity tag 1 instead of receiving a new one when re-verified in frame 34. The propagation of the projected boundaries in the short-time memory has in addition enabled rediscovering the irregularity in frames 28, 31, and 33 in addition to frame 34, since the irregularity still exists in memory it needs not accumulate votes yet another time to be picked up. The registration of discovered irregularities and the projection of irregularities not discovered, but hypothetically present, is listed in algorithm 1.

```
for verified irregularity in frame do
    get previous boundaries of irregularity
    classify irregularity if class is unclear
    calculate current movement
    store movement in movement register
    update detected boundaries in active irregularities register
    refresh countdown in active irregularities register
end
store detected irregularities in short-time memory
for active irregularity not found in frame do
    get previous boundaries of irregularity
    calculate projected movement
    calculate projected boundaries
    update projected boundaries in active irregularities register
    decrement countdown in active irregularities register
end
store projected irregularities in short-time memory
```

Algorithm 1: Once an irregularity has accumulated enough votes to be verified, extra measures are made to make sure the irregularity is highlighted in every frame in which it hypothetically exists. If the irregularity is detected directly in a frame, its movement from the previous frame is stored in the movement register, and it is added to the active irregularities register with its boundaries and a refreshed countdown. If irregularities exist in this register without being directly observed in a frame, then previous observations of that irregularity and its movement are used to calculate the hypothetical location of that irregularity in the current frame. In this case, the countdown timer is decremented such that the irregularity eventually dies if not re-detected. Projected and detected boundaries are always stored in the short-time memory to increase chances of re-detecting that irregularity in subsequent frames.



(a) Detected and projected centre coordinates with movement hypothesis based on relative movement of boundaries from previous til current frame.



(b) Detected and projected centre coordinates with movement hypothesis based on median of all previous detected movements.

Figure 7.8: A movement hypothesis based on the relative movement between current and previous frame such as (a) suffers from occasions where the observation does not smoothly approach the camera. Acknowledging that the movement could be jerky or elsewise nonlinear, a median hypothesis such as (b) smoothens the approximation of the actual coordinates to overlap extremely well with the observations when they re-occur.

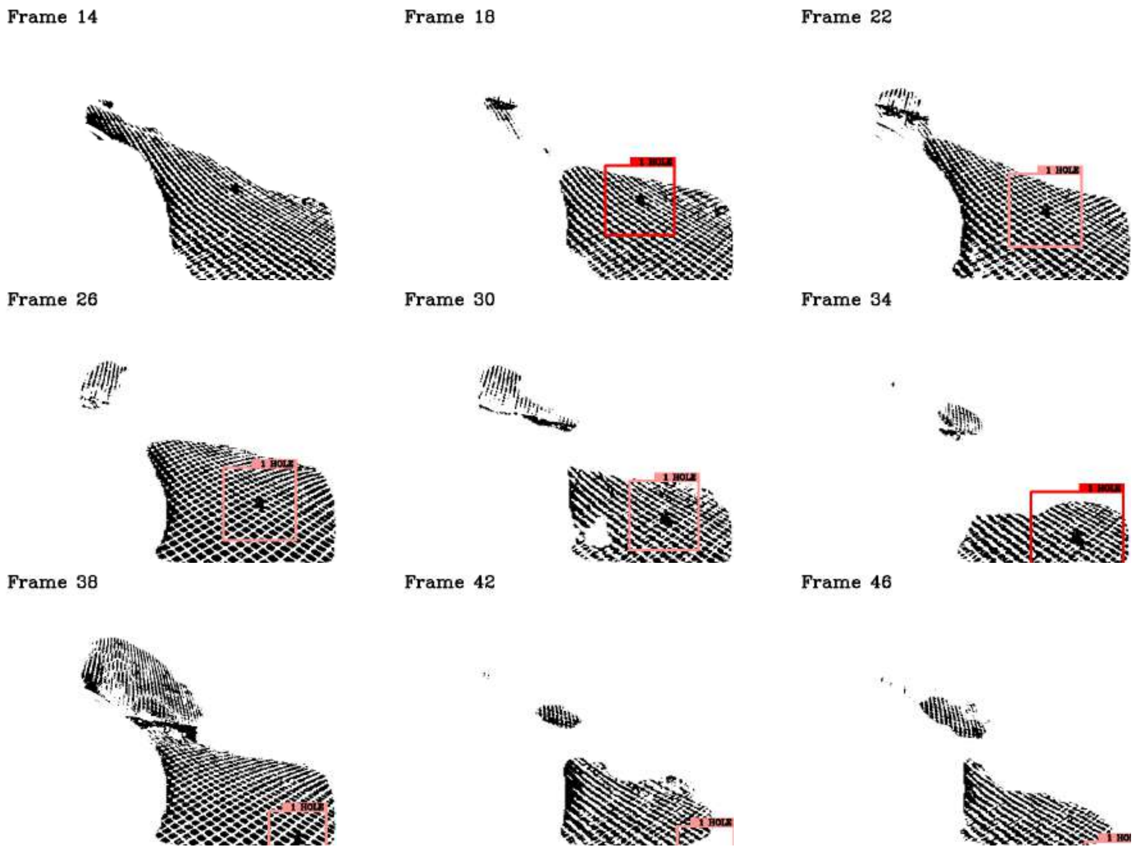


Figure 7.9: The probable hole with irregularity tag 1 is sporadically observed between frames 17 and 37 (see fig. 7.8). In-between these discoveries, the registered motion of the irregularity's centre allows it to be projected onto the scene (projections are visualised in pink colour). Updating the projected boundaries of the irregularity in the short-time memory allows it to be refreshed in later frames with the same irregularity tag, if the projected boundaries overlap with the newly detected boundaries.

Chapter 8

Results

Four main modules have now been established – all delicately intertwined to achieve robust hole detection in realistic environments. The deep learning based net thread segmentation module (NTS) from chapter 4 is at the heart of the program – interpreting every video frame as a binary scene where black areas represent the Background in-between net threads. The local irregularity detector in chapter 5 analyses every such binary image and scans them for atypical pieces of Background – recognised by a morphological scheme utilising an adaptive variable called *The Elbow*, a measure of the typical Background in a local neighbourhood. The spatiotemporal filter deduced in chapter 7 tracks local irregularity reports and verifies only those irregularities that sustain themselves in both space and time. Once an irregularity has been verified, the scene interpreter from chapter 6 is brought to play with a final trial – a deep convolutional network trained on millions of images and then specialised on hole recognition – deciding whether or not the verified irregularity is indeed a Hole, a Fish, or, simply, Nonsense. Then, eventually, a hole can be reported, highlighted, tracked, and archived.

8.1 Scores

To investigate the effectiveness of this scheme was developed 10 ten-second test videos drawn from the MANTA1 and MANTA4 videos. Four videos displayed actual holes (No. 1, 2, 5, and, 9), a couple contained swimming fish (No. 8 and 9), and five videos (No. 3, 4, 6, 7, and 10) were seemingly hole- and fish-free. Properties of especial concern during testing were the following:

1. How well-produced is the binary image? Is irrelevance correctly ignored, and is the net structure fairly and coherently represented?

2. How robust is the local irregularity detector? The number of detected irregularities (yet to be filtered by the spatiotemporal filter) affects runtime, so it should be kept as low as can be without missing holes.

3. Are all true holes reported? TP (true positives) denotes the number of correctly reported holes. FP (false positives) indicates the number

of false alarms raised – hole reports where there are none. Lastly, FN (false negatives) – the number of holes present but not reported. TN (true negatives) is not a useful metric in this application since the number of *not present holes not reported on* is, practically, infinite.

4. Are fish correctly neglected? Ideally, fish are overlooked already by the segmentation module, but if they are not – does the scene interpreter manage to separate them from real holes? TP counts the number of fish correctly classified by the scene interpreter, and FP the number of irregularities falsely classified by Fish as the scene interpreter. FN is considered not useful since the program's main concern is with hole detection and not fish detection. Neither is TN considered – yet again because the number of *not* present fish *not* detected on is infinite.
5. Is Nonsense properly classified? As with Fish, we will only care for TP and FP to measure Nonsense detection. The main concern is, yet again, not to detect every instance of Nonsense, but rather to achieve precision when any irregularity is classified as it.
6. How effective is the execution of the analysis? This work is at proof-of-concept stages indeed, but are some analyses significantly halted?

Additionally, all test videos were run a second time with salt-and-pepper noise corrupting 2% of the pixels of each frame. These tests were executed to reveal potential weaknesses to noise existed, and whether or not such could be ameliorated.

table 8.1 summarises the findings of these tests, and a brief discussion of each video follows. Each discussion is illustrated with a screenshot from the operation, the components of which are described in fig. 8.1.

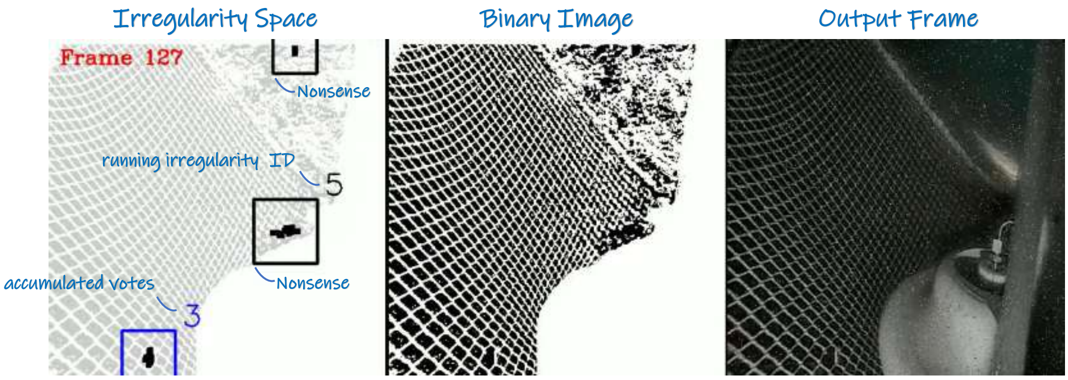


Figure 8.1: A scene from the noisy version of test video 1. The leftmost image shows the irregularity space, where local irregularities appear as **blue** squares with a vote count in an upper corner. Irregularities reaching a vote count of 7 are evaluated by the scene interpreter, and the square changes colour to **black** (indicative of Nonsense), **green** (for Fish), or **red** (for Holes). The vote count is replaced by the running irregularity number, starting at 1 and incrementing. The middle image shows the binary representation produced by the NTS module (often poor in noisy test runs), and the rightmost image, the original frame with verified Holes and Fish highlighted.

Table 8.1: Noisy tests are tagged with N-suffixes. QR codes lead to videos (or visit appendix A). Views: S(tarboard), P(ort), F(ore). NIRR is the number of local irregularities. Subsequent short-hands are True Positives (-TP), False Positives (-FP), and False Negatives (-FN), for Holes (H-), Fish (F-), and Nonsense (N-). SP30F is mean execution time [s] per 30 frames.

No.	QR	View	Segmentation	NIRR	HTP	HFP	HFN	FTP	FFP	NTP	NFP	SP30F
1		S	Mostly Excellent	152	1	0	0	0	0	0	0	2.38
1N			Mostly Poor	659	0	0	1	0	1	7	0	2.87
2		P	Mostly Excellent	169	1	1	1	0	0	0	0	2.47
2N			Decent	1181	1	0	1	0	0	10	0	11.63
3		P	Excellent	109	0	0	0	0	1	0	0	2.43
3N			Mostly Good	523	0	0	0	0	0	2	0	2.87
4		F	Very Good	584	0	0	0	0	0	1	0	3.06
4N			Poor	2154	0	0	0	0	0	18	0	20.39
5		S	Very Good	148	2	0	0	0	0	1	0	2.20
5N			Poor	1409	0	1	2	0	0	18	0	6.99
6		F	Good	74	0	1	0	0	0	0	0	2.18
6N			Poor	881	0	0	0	0	0	6	0	4.00
7		F	Decent	1769	0	0	0	0	0	19	0	22.53
7N			Poor	2416	0	0	0	0	0	25	0	21.60
8		F	Excellent	126	0	0	0	1	0	0	0	2.21
8N			Poor	2764	0	0	0	0	0	32	0	17.43
9		S	Excellent	291	1	0	1	2	0	1	0	2.44
9N			Mostly Good	834	0	0	2	1	0	6	0	3.88
10		F	Varying	993	0	0	0	0	0	13	0	3.19
10N			Poor	2691	0	0	0	0	0	26	0	45.67

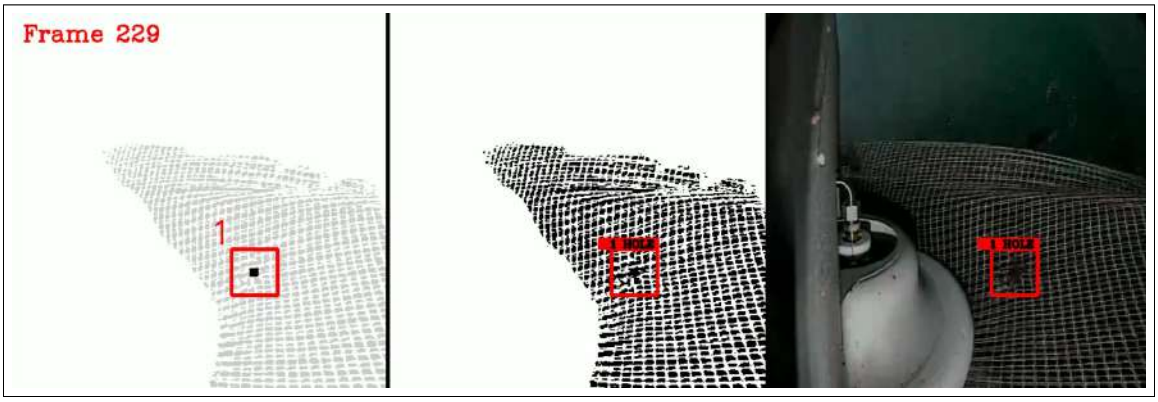


Figure 8.2: The scheme finds the potential hole in test video 1, even though speed is high and net structure is deformed.

8.2 Test Video 1: A Quick Glimpse

The first test video is fetched from the Manta's starboard view, often not prioritised by ROV operators during a cleaning operation. The net is clean and, mostly, excellently segmented by the NTS module. However, it is rather impacted by the water jets, spatially deforming in ways which could pose trouble to hole detectors. One hole is detected (see fig. 4.12), but it is partially *filled* by the segmentation module shortly after its detection – making it a short-lived discovery.

The local irregularity detector detects 152 irregularities in this video, half an irregularity per frame, a workload comfortably handled by the program. No other irregularities are verified in this video.

With additive noise – the segmentation module *severely* oversegments the video, including not only net structure in its segmentation, but frankly, the entire frame. This causes 659 local irregularities to arise – most of which discovered in the water column above the net, but neither verified as Hole. This proves to show that a dysfunctional segmentation module can be rescued by the scene interpreter – classifying 7 verified irregularities as Nonsense and one (falsely) as Fish. The actual Hole was reported in 3 frames, not reaching the 7 frames-threshold set for verification.

8.3 Test Video 2: Two Holes

The second test video (fig. 8.3) is of approximately the same nature as test video 1, showing the port view, and clean, slightly deformed net structure with two visible holes. One is misrepresented, mostly, by the segmentation module, appearing intact more often than not. The other hole is reported and neatly tracked for circa two seconds. Nearing the end of the video, a whirl of turbid waters obscure the net, causing large irregularities to (falsely) appear in the binary image. This lasts for long enough for it to be reported as a Hole.

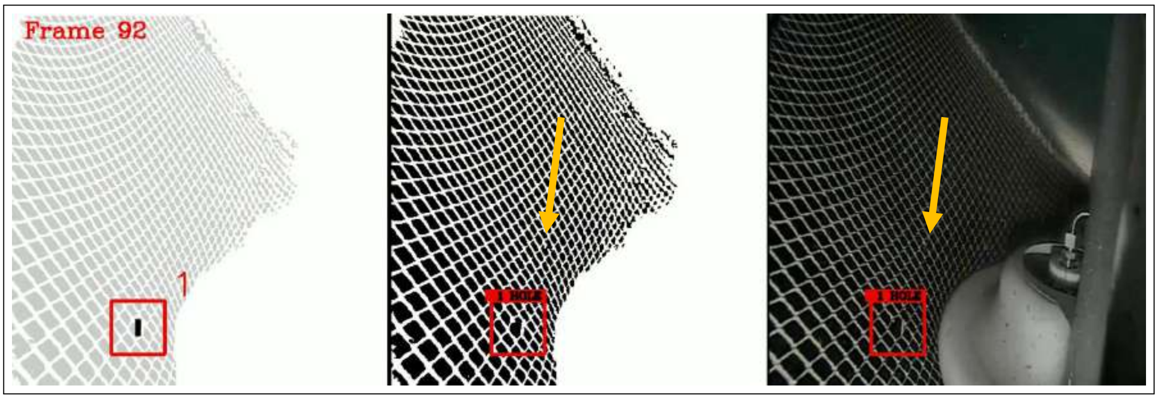


Figure 8.3: One of two holes in test video 2 is recognised regardless of the additive noise. However, one hole is missed in both cases (yellow arrow), most definitely because it is often segmented as intact. The unreported hole is more prominent during earlier stages of the video.

With additive noise, interestingly, one hole is still recognised, albeit with some delay as compared to the noise-free test. The second hole is still not recognised. As with test video 1, the noise confuses the segmentation module, but this has apparently less severe impact since more of the screen is covered by net structure in the video. Nevertheless, 1181 local irregularities arise (compared to the noise-free's 169) which causes significantly computational strain on other modules. On a positive note, no false positive hole detection takes place, because the scene interpreter correctly identifies all other verified irregularities as Nonsense.

8.4 Test Video 3: A Curious Fish

Test video 3 (fig. 8.4) shows a stationary scene filmed from the port view where a fish enters the scene at about 3 seconds. The fish spends the rest of the screen time of 7 seconds swimming towards the ROV before backing up. The fish is sometimes recognised as Foreground – white – sometimes as Background – black – in the binary image. This problem was also encountered and discussed by Haugene [17]. However, the scene interpreter confidently identifies the potential Hole as nothing but Fish, and the fish is henceforth neatly tracked.

With additive noise, oversegmentation causes 523 local irregularities to arise (as compared to 109 with no noise), of which two were verified and tracked, correctly, as Nonsense. The fish was not identified in the noisy test video. All things considered, the noise did not manage to cause any significant trouble but increasing runtime due to more hefty involvement of the scene interpreter and spatiotemporal filtering.

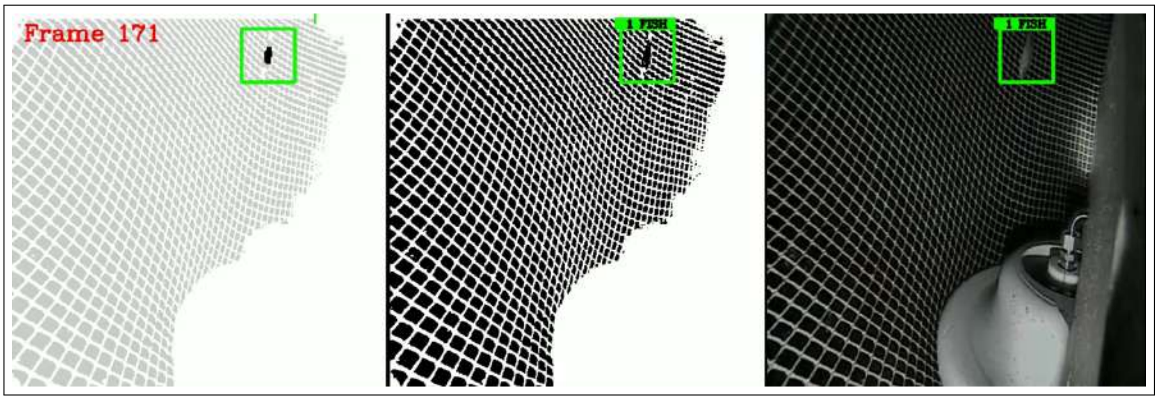


Figure 8.4: The third test video shows a quiet scene and a curious fish.

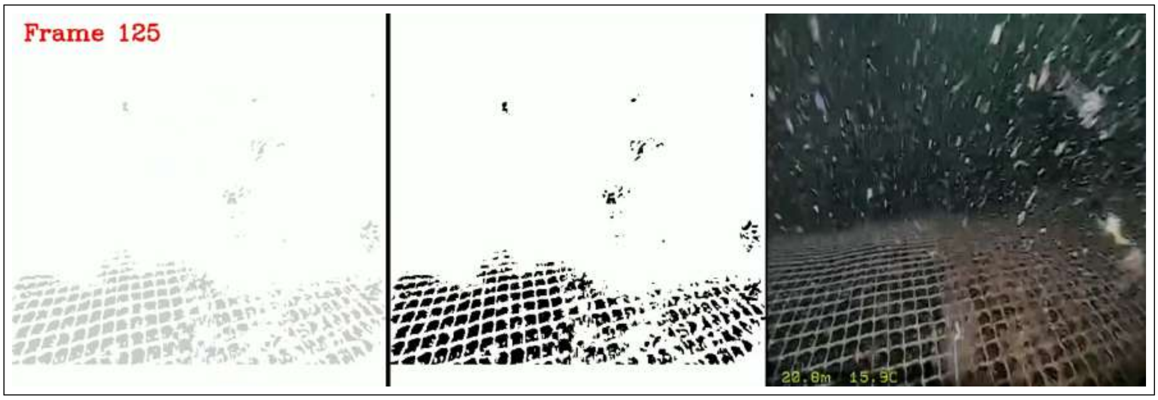


Figure 8.5: The fourth video depicts a challenging scene during a net cleaning operation. No false positive hole reports are generated.

8.5 Test Video 4: A Messy Clean-Up

The fourth test video (fig. 8.5) shows a cleaning operation, from the fore view, at high speed, with large chunks of algae flying in the face of the camera. The segmentation manages to represent the net structure fairly well, but also includes a lot of floating algae, causing 584 local irregularities to appear during noise-free testing. However, only one of these is verified and correctly classified as Nonsense.

With salt-and-pepper noise, the segmentation is pathologically overeager, including yet again too much irrelevance in its product, but still representing net structure fairly well. 2154 local irregularity reports arose, of which 18 were verified and classified as Nonsense. Implementational details can likely be improved significantly (perhaps by migrating from Python, and parallelising effectively) but an increased number of local irregularities is probably why the test executes at 20 x real-time, instead of the typical 2.2 - 2.5 x real-time achieved in noise-free videos.

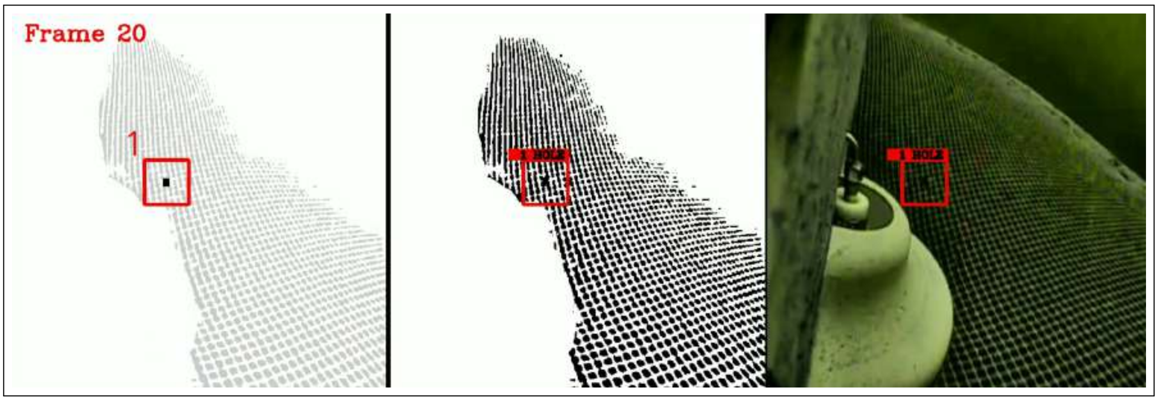


Figure 8.6: A tiny hole seems to appear in the early stages of test video 5. A few seconds later appears a more prominent hole.

8.6 Test Video 5: Two (?) Holes

The hole detection scheme discovered the intended, probable, hole near the end of the video sequence, but also a tiny yet credible contendent appearing early on in the video (see fig. 8.6). It is surely hard to tell whether or not these two discoveries constitute real tangible holes, but they seem to be very interesting discoveries given the non-spectacular video quality. The non-noisy test is therefore granted two true positive hole discoveries, and no erroneous reports.

The noise causes, again, the segmentation module to malfunction, triggering one false hole report and neglecting the two holes actually present. Again, the spatiotemporal filter and the scene interpreter manage to prevent 1409 local irregularities from wreaking havoc, limiting the impact of those to 18 verified instances of Nonsense and the aforementioned false positive hole report.

8.7 Test Video 6: School of Fish and Motion Blur

The sixth test video is one where the camera *jumps*, as it were, filming a school of fish and then some net structure. The net is clean, but the video is clearly impacted by motion blur, causing the net sporadically to appear as a set of parallel straight lines (see fig. 8.7). This causes a false positive hole to be verified in the very last frame of the video.

One fair criticism could be that the spatiotemporal filter does not have an impression of the overall scene movement. In other words, if a hole is consistently reported in the very same section of a frame, if the scene moves significantly, then a *true* irregularity would move consistently with the scene and not be repeatedly reported at the same location.

The noisy test executes like previous noisy tests, with the segmentation crumbling and secondary modules coming to the rescue. No false hole report is

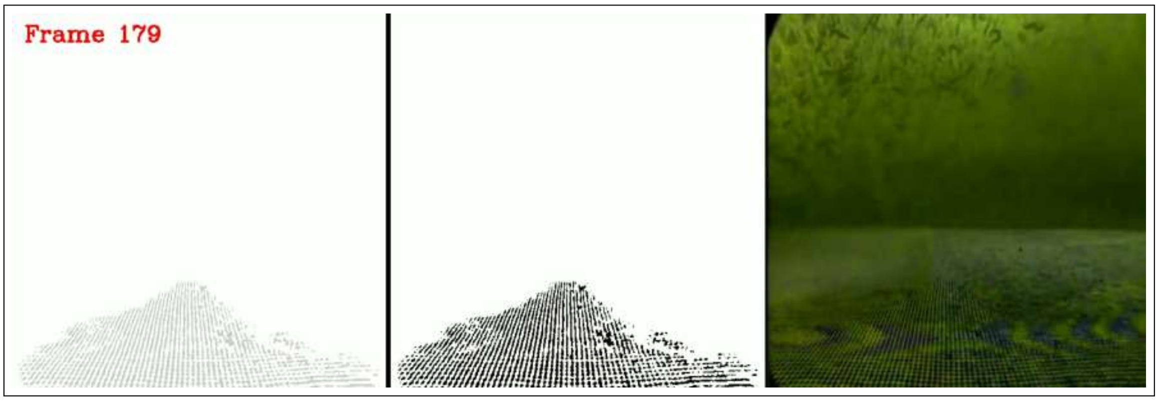


Figure 8.7: The video quality poses problems for the hole detector in test video 6, eventually triggering a false hole report.

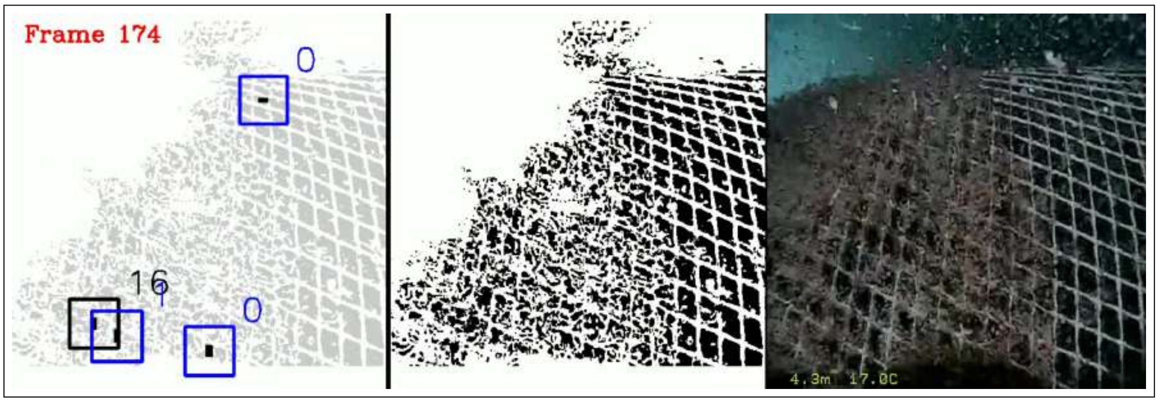


Figure 8.8: Test video 7 shows a lot of heavily grown net structure, most of which included in the segmentation. The scene interpreter, however, correctly identified verified irregularities as Nonsense.

generated in this case, but the program slows down due to the massive number of local irregularities demanding attention.

8.8 Test Video 7: Close-Up Heavy Growth

The seventh video is challenging, showing first net structure extremely close-up and completely covered by algae (see fig. 8.8). Ideally, the first seconds of the video should therefore not be considered by the segmentation module, since the scene is hardly interpretable. After a few seconds, clean net structure enters the scene. This net is perfectly represented in the binary image.

The inclusion of heavily grown net structure triggers 1769 local irregularities in the **noise-free** test, and 2416 during testing **with** noise. These numbers are reasons for concern indeed, but despite this no false hole reports are generated.

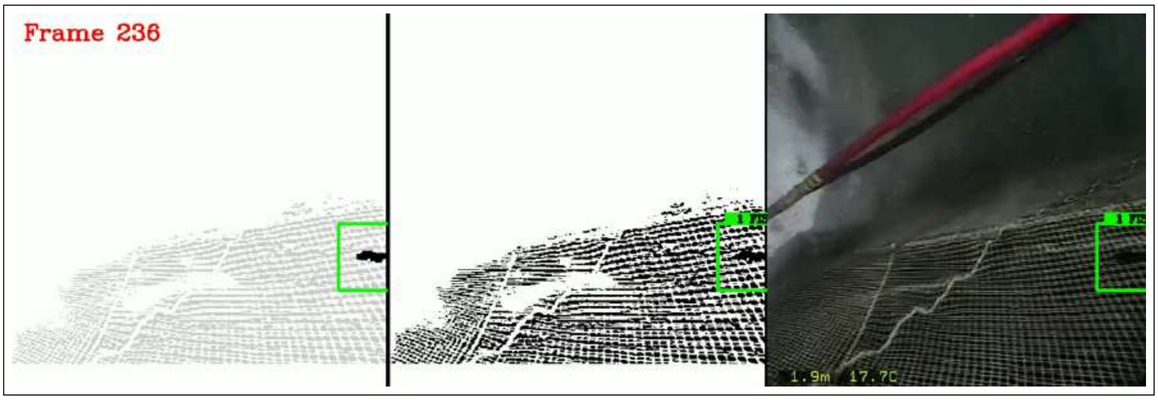


Figure 8.9: The scene interpreter claims to have detected a fish in test video 8. Several fish in the starboard view (see test video 9) substantiate that claim.

8.9 Test Video 8: Surface, Cables, and... a Fish?

Test video number 8 contains a myriad of foreign objects and deformed net structure. The segmentation is nothing but exemplary, providing subsequent modules with easily interpretable binary representations. Nearing the end of the video, one dark spot is reported as Fish (see fig. 8.9). This is most likely a correct observation given the shape of the shadow. Test video 9 is the starboard view at the same time interval, and there are indeed several fish present, substantiating the claim that the scene interpreter makes a perfectly valid judgement (even superhuman).

With additive noise, all alien objects are segmented, raising 2764 local irregularities of which 32 are verified as Nonsense. No false positive hole reports are generated.

8.10 Test Video 9: Test Video 8 – But Starboard

Test video 9 contains what seems to be a tiny hole that is tracked for only three frames (through frame 69, see fig. 8.10), and therefore not attaining verification. One hole report is generated, though it is not perfectly whether or not the detected hole is in fact a hole or an old, repaired, hole. Furthermore, two fish are correctly identified.

With additive noise, the suspected unreported hole is discovered in only two separate frames, and neither the second hole was not consistently enough noticed to be verified. The fish, however, was correctly classified once, and no false positive hole reports were generated. The segmentations, however, were in a sense better than for other videos, but mostly because net structure covered more or less the entire frame.

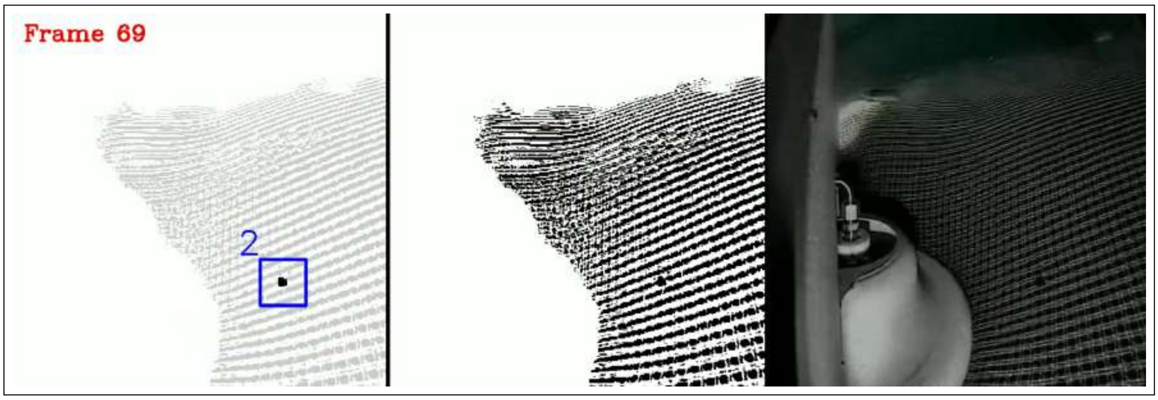


Figure 8.10: There seems to be a hole in video 9 that received too few votes to be verified.

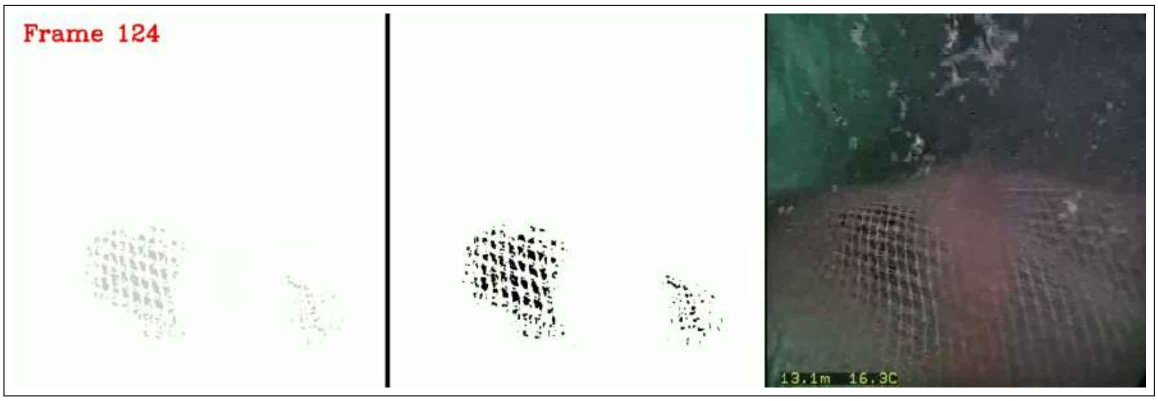


Figure 8.11: This particular distance to the net dominates the NTS training dataset, hypothetically answering the question why it struggles so hard to segment net structure when it is very close-up. The segmentation of test video 10 is good at this distance to the net, but horrific in later parts of the video.

8.11 Test Video 10: More Flying Algae

The tenth video resembles what we witnessed in test video 4; rapid movements during cleaning of significantly algae-covered net. The segmentation is fairly good, and early stages of the video displays both dark and bright net structure, simultaneously (see for instance around frame 11), which is impressively well segmented. However, it struggles with very close-up net, hypothetically due to lack of exposure to such during training.

No video triggers more local irregularity reports than the noisy version of test video 10: 2691 culminating in 26 verified instances of Nonsense and astonishingly high runtime. Still no false positive hole reports arise, but the computational effort is beyond significant.

Chapter 9

Discussion

> *"Almost all ideas are wrong. It doesn't matter if they're your ideas or someone else's ideas. They're probably wrong – and even if they strike you with the force of brilliance – your job is to assume, first of all, that they're probably wrong, and to assault them with everything you have in your arsenal and see if they can survive."* [1]

The writing of this thesis has been an adventurous journey. We have ventured through traditional computer vision techniques and novel neural network architectures and training strategies; visiting ORB-matching, transfer learning, edge detection, spatiotemporal U-Net segmentation, scene classification, data augmentation, region growing, and mathematical morphology. All woven together in a new and, hypothetically, interesting way to attempt solving the problem of *robust fish cage hole detection in challenging environments*.

The results from the previous chapter, chapter 8, bring to the table convincing evidence that the proposed framework is indeed capable of detecting holes in fish cages in challenging environments, and, furthermore, showing robustness in the sense that fish and foreign objects are discriminated from holes. However, there are talking points which deserve scrutiny. This chapter will discuss the achieved results, and how properties of the proposed framework have contributed to strengthening, or weakening, the results.

9.1 Segmentation

First and foremost, the MultiRes U-Net proved to be an extraordinary segmentation agent, and the temporal continuity encouragement through lag masks evidently stabilised its product significantly. The introduction of lag masks initially lead to deadlock situations when naïvely applied to real video footage, but the proposed blurring and regularisation scheme during training yielded unprecedented results.

[1] Psychologist Jordan Peterson in debate with philosopher Slavoj Zizek in Toronto, 19 April 2019 [62].

Quite a few words have been written on a strategy that did not make it to the final stage – 3CAS coupled with adaptive binarisation. 3CAS was a natural *next step* from the initial NeNoS module, but it lacked, in a sense, ambition. Although 3CAS prediction could be executed at high speeds, when coupled with the binarisation scheme (and refinement in terms of denoising and background removal) the advantage to NTS was lost. It was simply not expected that NTS would provide such stable results, seamlessly combining segmentation, binarisation, and denoising in a single operation.

It was frankly surprising to notice how sensitive the NTS algorithm was to salt-and-pepper noise during each and every test video in the previous chapter. Unexpected – since the testing videos were quite different from the training material (NTS trained on MANTA2 and tested on MANTA1 and MANTA4), but they were nevertheless well-segmented – even when "naturally corrupted by noise", i.e. by floating algae, poor video quality, or elsewise troublesome scenes.

NTS struggled not to segment the net structure per se, but it severely over-segmented when exposed to noise, including the entire frame in its segmentation. This was awkwardly interesting. Awkward because it showed a significant weakness in the proposed segmentation module. Interesting because *extremely few false positive hole reports arose*. It was a brilliant stress-test for the scene interpreter, who called off every false alarm but **one**, in the noisy video 5. However, the badness of the segmentation lead to significant increase in runtime due to the extreme amount of arising local irregularities (median runtime *per second of frames* was 9.31 s for noisy videos and 2.44 s for the original videos). Exactly *where* this bottleneck happened is not well-mapped. Suspicions are that the recursive deduplication of nearby irregularities in the irregularity space was not well-designed for large numbers of irregularities. Or it could simply be the case that matching dozens of new irregularities per frame with an equally large number of old irregularities in short-time memory was not carried out in a manner suitable for such numbers.

9.1.1 Proposed Actions

The input to the NTS module is a 512 x 512 x 4 image, where the first three components of each pixel are its RGB intensity values and the latter, the binary lag mask pixel value. We experimented successfully with corrupting the lag masks during training and, hence, making the module less reliant on the information of the past. A suggestion, to make the module robust also to corruption in the input image, is to apply regularisation to the first three components of its pixels. If the networks were trained, also, with a significant number of its inputs affected by salt-and-pepper noise, then, intuitively, they should be better equipped to handle such events in real-life. Naturally, the training data foundation should be significantly extended. By exposing the NN to more examples of what a satisfactory segmentation looks like in varying environments, it is better prepared to handle new situations properly on its own.

An additional note: if parts of the frame contain static ROV parts, one may *help* the algorithm by applying to each proposed segmentation a static mask that excludes these parts. If an observant reader wonders why over-eager segmentations in noisy videos perfectly ignored ROV parts in port and starboard views – here is the reason.

9.2 The Local Irregularity Detector

Choosing *flat kernels* over *square* or *cross* kernels was a risk taken well aware of the potential rising number of false positives triggered. The kernel shape over-reports especially in scenes with severe motion blur. However, it is the only kernel capable of recognising tear in single threads, creating rectangular holes not broader than the typical mesh. Requiring some minimum kernel breadth- or -height was absolutely vital; without such demand, single-column irregularities appear very frequently doe to the imperfect nature of real-life video.

Tiling solved, somewhat, the problem that *relatively large holes in one part of the image are not necessarily large compared to intact net "holes" near the camera*. Tile analysis is intuitively parallelisable, and a moving camera will not let any hole dwell on the unfortunate border between tiles for long. However, there is something unsatisfactory with the approach due to its crude and somewhat trivial drawing of borders. It seems that a more *continuous* idea would be more ideal – more like what humans do. Exactly how one can, in a computationally efficient manner, compare each piece of Background to its immediate neighbourhood is, however, not clear as of yet.

9.2.1 Proposed Actions

All things considered, the flat kernels with tiling managed to identify most present holes in the test material, in both near and far proximity to the camera. However, the number of local irregularities arising especially when the segmentation is poor is worrying. Perhaps is this a problem best solved by increasing the reliability of the segmentation. Alternatively, one could choose a cross-shaped kernel in the local irregularity detector – promoting aversion to false-positive detection, but, equivalently, increasing blindness to small holes. In any case, it seems not fitting to recommend this program for commercial use until the stability of the local irregularity detector is improved, and perhaps it needs complete rethink. In that case – there are plenty of ideas left in the framework which will, unbothered, facilitate another hole detector.

9.3 The Scene Interpreter

The scene interpreter came into play more often than it should during testing – due to faulty segmentations in the noisy videos. And luckily so, proving to be incredibly reliable in its predictions. Utilising the VGG16 [43] pre-trained model as a base model for this module turned out to be an ingenious move, ramping up results from useless to formidable. Hardly any mistakes were made by the module during testing, and it even outperformed its maker in test video 8 where it correctly identified a moving shadow as Fish.

9.3.1 Proposed Actions

The training foundation for the module should be increased. Even though performance is stellar – it is absolutely vital that the module recognises a hole when it sees one. Ideally, one should fetch as many examples of holes as possible and include those in the *Net Structure* data foundation. If not, one can wound up in a situation where it finds the appearance of holes to be more similar to that of Fish than to Net Structure.

Additionally, one can adjust the training examples such that the scene interpreter learns to classify images with severe motion blur as Nonsense. This could work as an additional barricade to prevent motion blur from triggering hole reports.

9.4 Spatiotemporal Filtering

The spatiotemporal filter cannot itself identify irregularities in a scene. It is entirely reliant on the local irregularity detector doing its job properly and reporting the presence of irregularities, repeatedly. However, in order to enable repeated reports on *actual* irregularities, we may have to make the local irregularity sensitive to the point that it engages in a certain degree of over-reporting. The spatiotemporal filter is designed to solve this very conundrum, by evaluating local reports in the light of past reports – searching for spatial and temporal continuity.

The tuning of this filter inevitably affects its performance, and there are certain parameters that should be considered; first and foremost, irregularities of spatial proximity are being merged through a recursive algorithm. This to lower the number of irregularities per frame, especially considering the fact that nearby irregularities will most likely match with similar future irregularities anyways. Irregularities with a Jaccard index surpassing 75% were merged to a single instance, and the new bounding coordinates were simply inherited from the extrema of the two conjoined instances. This threshold was visually satisfactory, but it could be the case that some other threshold is better. However, lowering the threshold could lead to the conjoinment of irregularity reports that truly do not belong together, and, possibly, if the coordinate boundaries fluctuate too much on a frame-to-frame basis, rendering frame-to-frame matching very difficult. Increasing the threshold, on the other hand, can put computational strain on the matching process, since there are *more* matches to make. Additionally, true irregularities can sometimes manifest themselves in two slightly disjoint reports in the irregularity space (due to properties of the binary image and the morphological operations) in which case a conjoinment of the two is very much preferable, contrary to treating them as two different entities.

The required Jaccard index to verify a match should be tuned alongside padding strategies and the required number of matches to verify an irregularity. Interestingly, when working with Jaccard index, even what seems like a significant overlap can yield a pretty low score. The working thresholds have been tuned throughout this project, and we utilised the thresholds of 15% Jaccard index and 7 votes for the two. Even though slow-paced video can yield Jaccard indices of up to 90% on a frame-to-frame basis, the overlap is significantly smaller in a high-speed sequence where, for instance, we have to utilise the buffer because an irregularity is missed in one or two frames. By requiring little overlap, but several votes, we allow high speed and sporadic blindness, but only if the temporal consistency is high. Of course – these numbers can be tuned differently. For instance, efficiency will increase if the required Jaccard score in increased, because fewer irregularities will match, and, hence, fewer will sustain themselves to accumulate votes. With a significantly large vote threshold, it is extremely hard for false positives to make it through to verification, but chances are that real holes will be missed as well.

The short-time memory length was set to **three**. It is surely not recommended to increase this length, but perhaps, to reduce it to **two**. If the length were to be reduced to **one**, there is no longer room for skipping a frame (which is undesirable), but with larger buffer it is no longer certain that what you observed in some area a long time ago is what you currently observe in that area. If the local hole detector is good (in that it consistently reports true holes) then there is perhaps no need for a large buffer. Shortening the buffer will surely increase efficiency, making fewer past reports available for matching with the current ones.

9.4.1 Proposed Actions

Investigations could be carried out to, in a scientific manner, deduce optimal parameters for this module; Jaccard indices for irregularity conjoinment, Jaccard indices for matchmaking, padding schemes, required number of votes to verify irregularities, and short-time memory length. How these test should be carried out has not been considered in this work, but a set of parameters on which all testing has been executed is suggested.

There has been some annoyance with the spatiotemporal filter's willingness to match apparent stationary irregularities in a moving scene. This tendency triggered false positive hole reports in test video 6 and 2, and is surely one of the reasons why the framework is not yet fit for commercial use. A suggestion is to deduce information about optical flow in the scene and to project past irregularities accordingly. Hence, only matching current irregularities with past ones if they occur where we *expect* past reports to be at in the current scene.

9.5 Tracking

The tracking of verified irregularities using a median centre movement hypothesis allowed *projection* of these irregularities onto frames in which they were not identified. This projection facilitated the *rediscovery* of these irregularities in future frames if irregularities in that frame were to match with the projection. This hypothesis was successfully implemented, making holes rediscoverable (after momentary loss of track) in videos 2 and 5. The projection of a recent irregularity also makes short-lived discoveries such as that in test video 1 detectable by humans, by providing a highlight that lasts for longer than the discovery.

9.5.1 Proposed Actions

More sophisticated movement hypotheses could be deduced to achieve better tracking under circumstances when the movement is not constant. The optical flow-proposal to improve the spatiotemporal filter could also be a viable option. After all – projecting past *unverified* irregularities is completely analogous to projecting past *verified* irregularities.

9.6 Conclusion

A complete framework has been developed, capable of reliable hole detection in a set of ten challenging 10-second test videos. The framework is of modular nature, and distinguishes successfully between net tear and alien obstruction such as by-passing fish.

Two modules were particularly effective; the deep learning approaches to net thread segmentation and scene interpretation. The local irregularity detector combined with the spatiotemporal filter operates as intended, but concerns have been raised with the irregularity detector's tendency to over-report, a property that puts considerable computational strain on the spatiotemporal filter. This is an issue that must be addressed before real-time usage can be achieved, a goal that requires a doubling of the suggested framework's (in its current Python environment) execution speed.

9.7 Future Work

The deep learning approach to net structure segmentation, specifically, the MultiRes U-Net with access to lag masks, produced outstanding binary representations of the test videos. However, more effort should be put into the creation of large and diverse datasets to further improve the models' robustness to noise.

The proposed local irregularity detector recognised most present holes in the test videos. Its main drawback is a certain degree of over-reporting in noisy environments, which in turn affects runtime. Specific parameters of the detector can be tuned to dampen this tendency, but potentially at the cost of missing true holes. A less rigid alternative to the tiling approach would be appreciated.

The spatiotemporal filter performed to a satisfactory degree during testing, but it should be further improved by including a scene motion hypothesis. Furthermore – alternatives to brute-force matching of past and current irregularities should be investigated. Perhaps can R-trees [63] serve as efficient data structures for past irregularities' boundaries.

Some effort was carried out to make use of *scene similarity* measures to try and match new discoveries with old ones. This could be relevant if the ROV operator drives past a hole and re-visits it at some later stage. As currently implemented, the framework will register the hole with two different tags. Limited success was made with Siamese networks [64] to compare the similarity of two scenes (see appendix D). Scene similarity was also attempted with ORB-matching. Underwater feature matching (specifically on homogeneous net structure) is notoriously difficult (read, for instance, [65]) but the coupling of this hole detection framework with a SLAM framework could render scene similarity research futile, since discovered hole can then be tagged with a global coordinate.

Bibliography

[1] A. Madshaven. *Pre-project: Fish Cage Net Analysis Exploting Computer Vision Techniques*. Trondheim, 2020.

[2] T. J. Pitcher and M. E. Lam. “Fish commoditization and the historical origins of catching fish for profit”. In: *Maritime Studies* 14.1 (Feb. 2015). ISSN: 2212-9790. URL: https://doi.org/10.1186/s40152-014-0014-5.

[3] J. Gustavsson et al. “Global Food Losses and Food Waste- Extent, Causes and Prevention”. In: *International Congress SAVE FOOD! at Interpack* (Jan. 2011). URL: http://www.fao.org/3/a-i2697e.pdf.

[4] FAO. *The State of World Fisheries and Aquaculture 2020. Sustainability in action.* 2020. URL: https://doi.org/10.4060/ca9229en.

[5] J. M. Erlandson and T. C. Rick. “Archaeology Meets Marine Ecology: The Antiquity of Maritime Cultures and Human Impacts on Marine Fisheries and Ecosystems”. In: *Annual Review of Marine Science* 2.1 (2010). PMID: 21141664, pp. 231–251. URL: https://doi.org/10.1146/annurev.marine.010908.163749.

[6] VKM. *Benefit-risk assessment of fish and fish products in the Norwegian diet – an update. Scientific Opinion of the Scientific Steering Committee.* 2014. URL: www.vkm.no.

[7] *Skattlegging av havbruksvirksomhet : utredning fra utvalg oppnevnt ved kongelig resolusjon 7. september 2018 : avgitt til Finansdepartementet 4. november 2019.* Oslo: Norge Havbruksskatteutvalget, 2019.

[8] *Status of wild Atlantic salmon in Norway 2020.* Accessed: 01.11.2020. Norwegian Scientific Advisory Committee for Atlantic Salmon, 2020. URL: https://www.vitenskapsradet.no/Portals/vitenskapsradet/Pdf/Status%5C%20of%5C%20wild%5C%20Atlantic%5C%20salmon%5C%20in%5C%20Norway%5C%202020T.pdf?ver=5edE21dH2DFOaPNai04Uig%5C%3d%5C%3d.

[9] O. Torrissen et al. “Salmon lice - impact on wild salmonids and salmon aquaculture”. In: *Journal of fish diseases* 36.3 (2013), pp. 171–194. ISSN: 0140-7775.

[10] *Om vern av villaksen og ferdigstilling av nasjonale laksevassdrag og laksefjorder.* Accessed: 01.11.2020. Norwegian Ministry of Climate and Environment, 2006. URL: https://www.regjeringen.no/no/dokumenter/stprp-nr-32-2006-2007-/id442061/?ch=9.

[11] *NS 9415:2003 (NS 9415:2009) Marine fish farms - Requirements for design, dimensioning, production, installation and operation*. Standard Norge, 2009.

[12] *Forskrift om krav til teknisk standard for flytende akvakulturanlegg (NYTEK-forskriften)*. nob. Accessed: 01.11.2020. Norwegian Ministry of Fisheries, 2012. URL: https://lovdata.no/dokument/LTI/forskrift/2011-08-16-849.

[13] *NS9416:2013 Landbased aquaculture farms for fish - Requirements for risk analyses, design, execution, operation, user handbook and product data sheet*. Standard Norge, 2013.

[14] *Losses in the production 1999-2019*. nob. Accessed: 25.05.2021. Norwegian Directorate of Fisheries, 2020. URL: https://www.fiskeridir.no/English/Aquaculture/Statistics/Atlantic-salmon-and-rainbow-trout.

[15] *Rømmingsstatistikk - antall og art*. Accessed: 25.05.2021. Norwegian Directorate of Fisheries, 2020. URL: https://www.fiskeridir.no/Akvakultur/Tall-og-analyse/Roemmingsstatistikk.

[16] H. M. Føre et al. *Tekniske årsaker til rømming av oppdrettslaks og regnbueørret*. Report no. 2019:00668. Trondheim: SINTEF Ocean AS, 2019.

[17] T. Haugene. "Evaluation of Methods for Robust, Automatic Detection of Net Tear with Remotely Operated Vehicle and Remote Sensing". MA thesis. Trondheim: Norwegian University of Science and Technology, 2014.

[18] W. Haugerud. *Kan ROV'er erstatte dykkere i havbruksnæringen?* nob. Accessed: 01.12.2020. Nov. 2019. URL: https://blog.akvagroup.com/no/kan-rover-erstatte-dykkere-i-havbruksnaringen.

[19] R. A. H. Jakobsen. "Automatic Inspection of Cage Integrity with Underwater Vehicle". MA thesis. Trondheim: Norwegian University of Science and Technology, 2011.

[20] J. Betancourt, W. Coral, and J. Colorado. "An integrated ROV solution for underwater net-cage inspection in fish farms using computer vision". In: *SN Applied Sciences* 2.12 (2020), pp. 1–15.

[21] S. Paspalakis et al. "Automated fish cage net inspection using image processing techniques". eng. In: *IET image processing* 14.10 (2020), pp. 2028–2034. ISSN: 1751-9659.

[22] Y.-P. Zhao et al. "An adaptive method of damage detection for fishing nets based on image processing technology". In: *Aquacultural Engineering* 90 (2020), p. 102071. ISSN: 0144-8609. DOI: https://doi.org/10.1016/j.aquaeng.2020.102071. URL: https://www.sciencedirect.com/science/article/pii/S0144860919301876.

[23] J. Y. Lettvin et al. "What the Frog's Eye Tells the Frog's Brain". In: *Proceedings of the IRE* 47.11 (1959), pp. 1940–1951. ISSN: 0096-8390.

[24] T. S. Huang. "Computer Vision: Evolution and Promise". In: *High technology: Imaging science and technology, International conference; 5th, High technology: Imaging science and technology*. 1996, pp. 13–20. URL: https://www.tib.eu/de/suchen/id/BLCP%5C%3ACN021986683.

[25] S. Gunasekaran. “Computer vision technology for food quality assurance”. In: *Trends in Food Science & Technology* 7.8 (1996), pp. 245–256. ISSN: 0924-2244. DOI: https://doi.org/10.1016/0924-2244(96)10028-5.

[26] H. Freeman. “On the Encoding of Arbitrary Geometric Configurations”. In: *IRE Transactions on Electronic Computers* EC-10.2 (1961), pp. 260–268. DOI: 10.1109/TEC.1961.5219197.

[27] M. E. W. Putra and I. Supriana. “Structural offline handwriting character recognition using levenshtein distance”. In: *2015 International Conference on Electrical Engineering and Informatics (ICEEI)*. 2015, pp. 31–36. DOI: 10.1109/ICEEI.2015.7352465.

[28] M. Akmal Butt and P. Maragos. “Optimum design of chamfer distance transforms”. In: *IEEE Transactions on Image Processing* 7.10 (1998), pp. 1477–1484. DOI: 10.1109/83.718487.

[29] C. T. Zahn and R. Z. Roskies. “Fourier Descriptors for Plane Closed Curves”. In: *IEEE Transactions on Computers* C-21.3 (1972), pp. 269–281. DOI: 10.1109/TC.1972.5008949.

[30] J. Schmidhuber. “Deep learning in neural networks: An overview”. eng. In: 61 (2015), pp. 85–117. ISSN: 0893-6080.

[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Commun. ACM* 60.6 (May 2017), pp. 84–90. ISSN: 0001-0782. DOI: 10.1145/3065386.

[32] S. Ji et al. “3D Convolutional Neural Networks for Human Action Recognition”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.1 (2013), pp. 221–231. ISSN: 0162-8828.

[33] R. Kohavi et al. “A study of cross-validation and bootstrap for accuracy estimation and model selection”. In: *Ijcai*. Vol. 14. 2. Montreal, Canada. 1995, pp. 1137–1145.

[34] Y. LeCun, C. Cortes, and C. J. C. Burges. *THE MNIST DATABASE*. Accessed: 01.12.2020. URL: http://yann.lecun.com/exdb/mnist/.

[35] L. Fei-Fei et al. *ImageNet*. Accessed: 01.12.2020. URL: http://image-net.org.

[36] J. MacQueen. “Some methods for classification and analysis of multivariate observations”. In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*. University of California Press, 1967, pp. 281–297. URL: https://projecteuclid.org/euclid.bsmsp/1200512992.

[37] M. Ester et al. “A density-based algorithm for discovering clusters in large spatial databases with noise”. In: AAAI Press, 1996, pp. 226–231.

[38] N. Otsu. “A Threshold Selection Method from Gray-Level Histograms”. In: *IEEE Transactions on Systems, Man, and Cybernetics* 9.1 (1979), pp. 62–66. DOI: 10.1109/TSMC.1979.4310076.

[39] J. Canny. “A Computational Approach to Edge Detection”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8.6 (1986), pp. 679–698. DOI: 10.1109/TPAMI.1986.4767851.

[40] N. Ibtehaz and M. S. Rahman. "MultiResUNet : Rethinking the U-Net architecture for multimodal biomedical image segmentation". eng. In: *Neural networks* 121 (2020), pp. 74–87. ISSN: 0893-6080.

[41] "Local Adaptive Thresholding". In: *Encyclopedia of Biometrics*. Ed. by S. Z. Li and A. Jain. Boston, MA: Springer US, 2009, pp. 939–939. ISBN: 978-0-387-73003-5. DOI: 10.1007/978-0-387-73003-5_506.

[42] R. M. Haralick, S. R. Sternberg, and X. Zhuang. "Image Analysis Using Mathematical Morphology". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-9.4 (1987), pp. 532–550. DOI: 10.1109/TPAMI.1987.4767941.

[43] K. Simonyan and A. Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556.

[44] M. Cristani et al. "Background Subtraction for Automated Multisensor Surveillance: A Comprehensive Review". In: *EURASIP J. Adv. Signal Process* 2010 (Feb. 2010). ISSN: 1110-8657. DOI: 10.1155/2010/343057. URL: https://doi.org/10.1155/2010/343057.

[45] B. K. Horn and B. G. Schunck. "Determining optical flow". In: *Artificial intelligence* 17.1-3 (1981), pp. 185–203. ISSN: 0004-3702.

[46] D. Marr, E. Hildreth, and S. Brenner. "Theory of edge detection". In: *Proceedings of the Royal Society of London. Series B. Biological Sciences* 207.1167 (1980), pp. 187–217. DOI: 10.1098/rspb.1980.0020.

[47] I. Sobel. "An Isotropic 3x3 Image Gradient Operator". In: *Presentation at Stanford A.I. Project 1968* (Feb. 2014).

[48] J. Illingworth and J. Kittler. "A survey of the Hough transform". In: *Computer vision, graphics, and image processing* 44.1 (1988), pp. 87–116.

[49] C. Harris and M. Stephens. "A combined corner and edge detector". In: *In Proc. of Fourth Alvey Vision Conference*. 1988, pp. 147–151.

[50] *Flying Net Cleaner*. Accessed: 23.02.2021. URL: https://sperre-as.com/portfolio/flying-net-cleaner-8/.

[51] *Manta Net Cleaner*. Accessed: 04.12.2020. URL: https://www.stranda.net/en/products/manta-net-cleaner.

[52] D. J. Simons and C. F. Chabris. "Gorillas in our midst: sustained inattentional blindness for dynamic events". In: *Perception* 28.9 (1999), pp. 1059–1074. DOI: 10.1068/p2952.

[53] The GIMP Development Team. *GIMP*. Version 2.10.12. June 12, 2019. URL: https://www.gimp.org.

[54] P. Bressan et al. *Semantic Segmentation with Labeling Uncertainty and Class Imbalance*. Feb. 2021. DOI: 10.13140/RG.2.2.21970.27846.

[55] K. Xu et al. "Spatiotemporal CNN for Video Object Segmentation". In: *CoRR* abs/1904.02363 (2019). arXiv: 1904.02363. URL: http://arxiv.org/abs/1904.02363.

[56] J. Cheng et al. *SegFlow: Joint Learning for Video Object Segmentation and Optical Flow*. 2017. arXiv: 1709.06750 [cs.CV].

[57] J. J. Mark. *Siddhartha Gautama*. May 2021. URL: https://www.worldhistory.org/Siddhartha_Gautama/.

[58] M. E. Olsen. "Camera-assisted ROV Navigation in Sea Cages". MA thesis. Trondheim: Norwegian University of Science and Technology, 2013.

[59] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. "ORB-SLAM: A Versatile and Accurate Monocular SLAM System". In: *IEEE Transactions on Robotics* 31.5 (2015), pp. 1147–1163. DOI: 10.1109/TRO.2015.2463671.

[60] F. Chollet et al. *Keras*. 2015. URL: https://github.com/fchollet/keras.

[61] Martin Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. URL: https://www.tensorflow.org/.

[62] J. B. Peterson and S. Zizek. *Marxism: Zizek/Peterson: Official Video*. Accessed: 23.05.2021. Youtube. Apr. 2019. URL: https://youtu.be/lsWndfzuOc4?t=1080.

[63] A. Guttman. "R Trees: A Dynamic Index Structure for Spatial Searching". In: vol. 14. Jan. 1984, pp. 47–57. DOI: 10.1145/971697.602266.

[64] S. Appalaraju and V. Chaoji. *Image similarity using Deep CNN and Curriculum Learning*. 2018. arXiv: 1709.08761 [cs.CV].

[65] L. O. Libjå. *Pre-project: Localization of defects in aquaculture fish cages*. Trondheim, 2020.

Appendices

Appendix A

QR Codes Hyperlinks

A.1 3CAS Segmentation Scores

fig. 4.12
https://youtu.be/y9_u2UUTT64
https://youtu.be/I1kC9Ie-Yms
https://youtu.be/7YEk_GcF9Lk
https://youtu.be/lpjBi8C9PbA
fig. 4.13
https://youtu.be/d5spFNWn328
https://youtu.be/i9IA7WuHIZU
https://youtu.be/fZt2euM0pMw
https://youtu.be/1Bgf7_MMfCo
fig. 4.14
https://youtu.be/Yop6l1qZQHI
https://youtu.be/CYNEuy6HK3A
https://youtu.be/XDASdk4eJSE
https://youtu.be/MAEcAXWugog
fig. 4.15
https://youtu.be/nFWTaJuWipl
https://youtu.be/2VEn9W3c3Lc
https://youtu.be/ETgN6pKLNPk
https://youtu.be/Ak67DKiQkXs

A.2 NTS Segmentation Scores

fig. 4.23
https://youtu.be/K_KzPQ2Pq50
https://youtu.be/a0YZytivHS8
fig. 4.25
https://youtu.be/JwUKgXxPCNs
https://youtu.be/ENc_Wcnk980
fig. 4.27
https://youtu.be/j5N9vPJni8M
https://youtu.be/6wVVJ-xnTz0
fig. 4.28
https://youtu.be/a0EwlkIdLaY
https://youtu.be/E-fqZnVfaCo

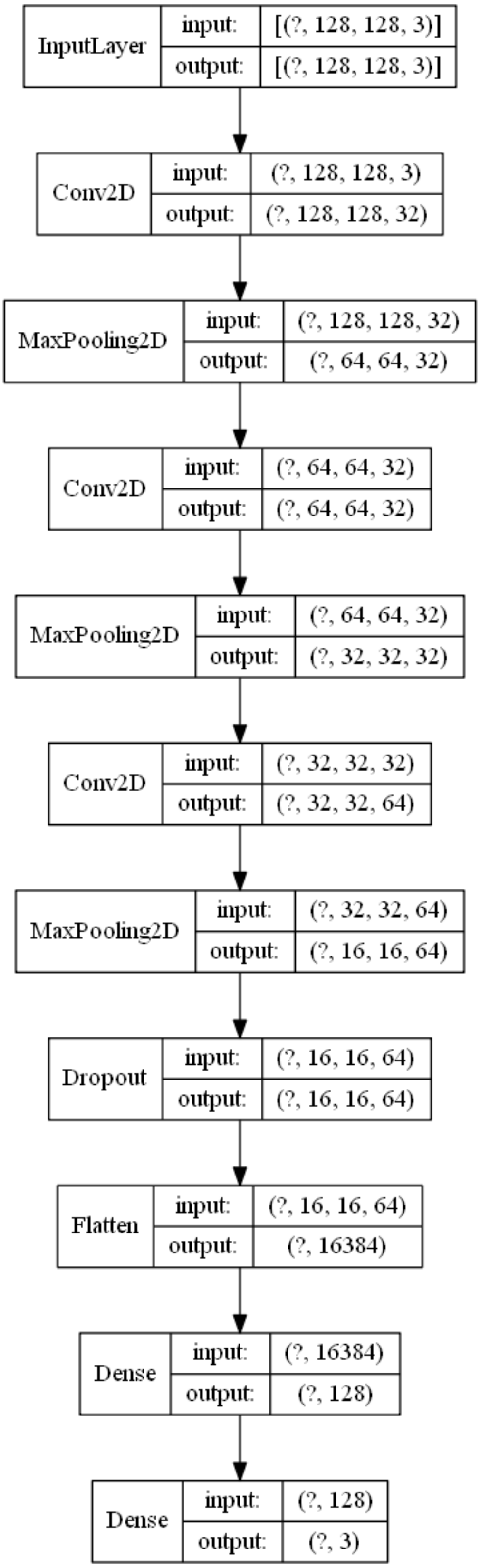
A.3 Test Video Results

table 8.1:
Test Video 1: https://youtu.be/4gsmAk-wgQI
Test Video 2: https://youtu.be/jikbSHNxkqA
Test Video 3: https://youtu.be/5vONsXaDQiU
Test Video 4: https://youtu.be/c-1wR3QSd7Y
Test Video 5: https://youtu.be/RGvdgyfO0Sg
Test Video 6: https://youtu.be/eCNNJ74jQjs
Test Video 7: https://youtu.be/1pe6VPuDMdk
Test Video 8: https://youtu.be/zA0dY0hS-hk
Test Video 9: https://youtu.be/4U0wmLrSzm8
Test Video 10: https://youtu.be/TOSdqTp4pBE

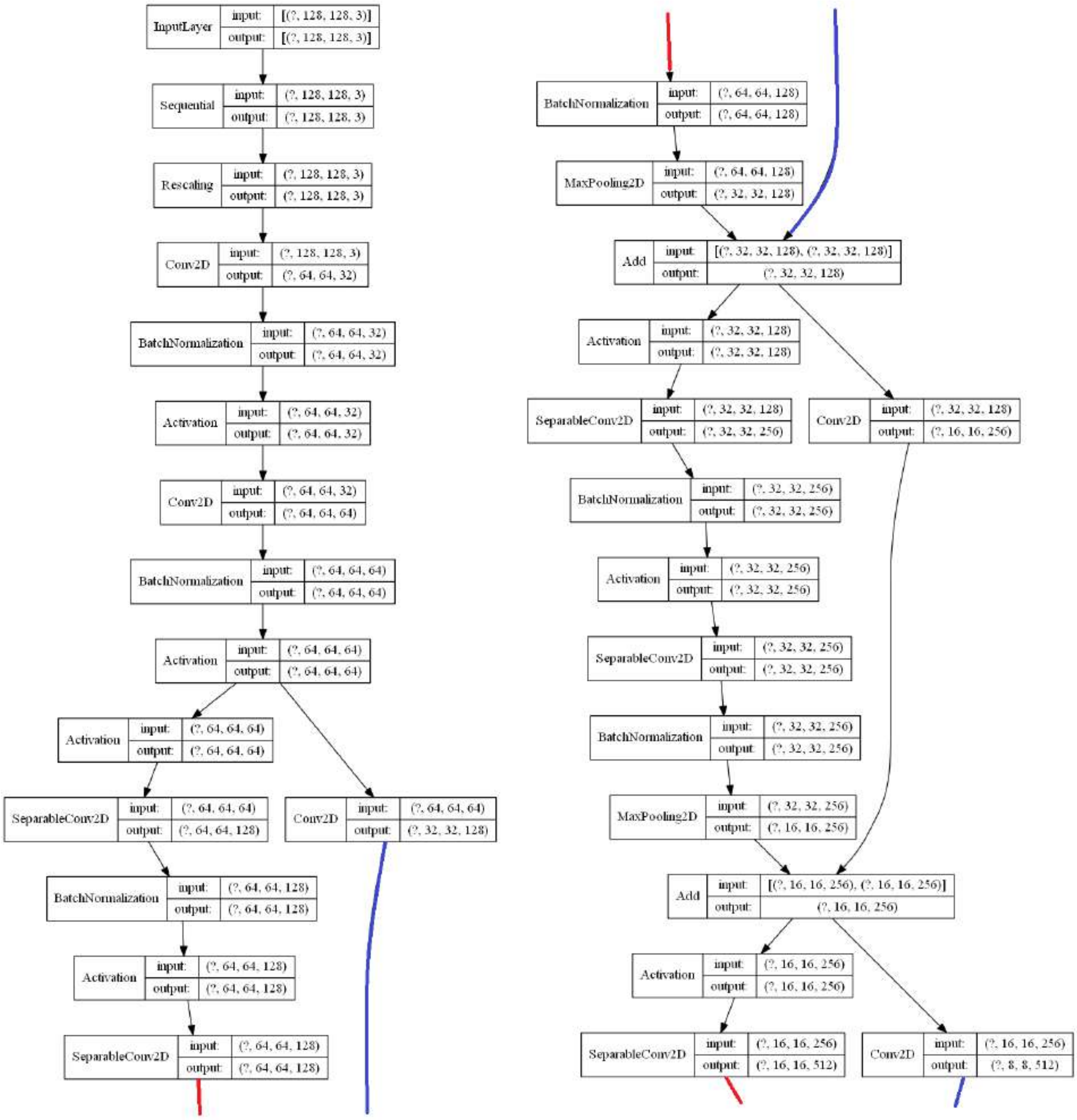
Appendix B

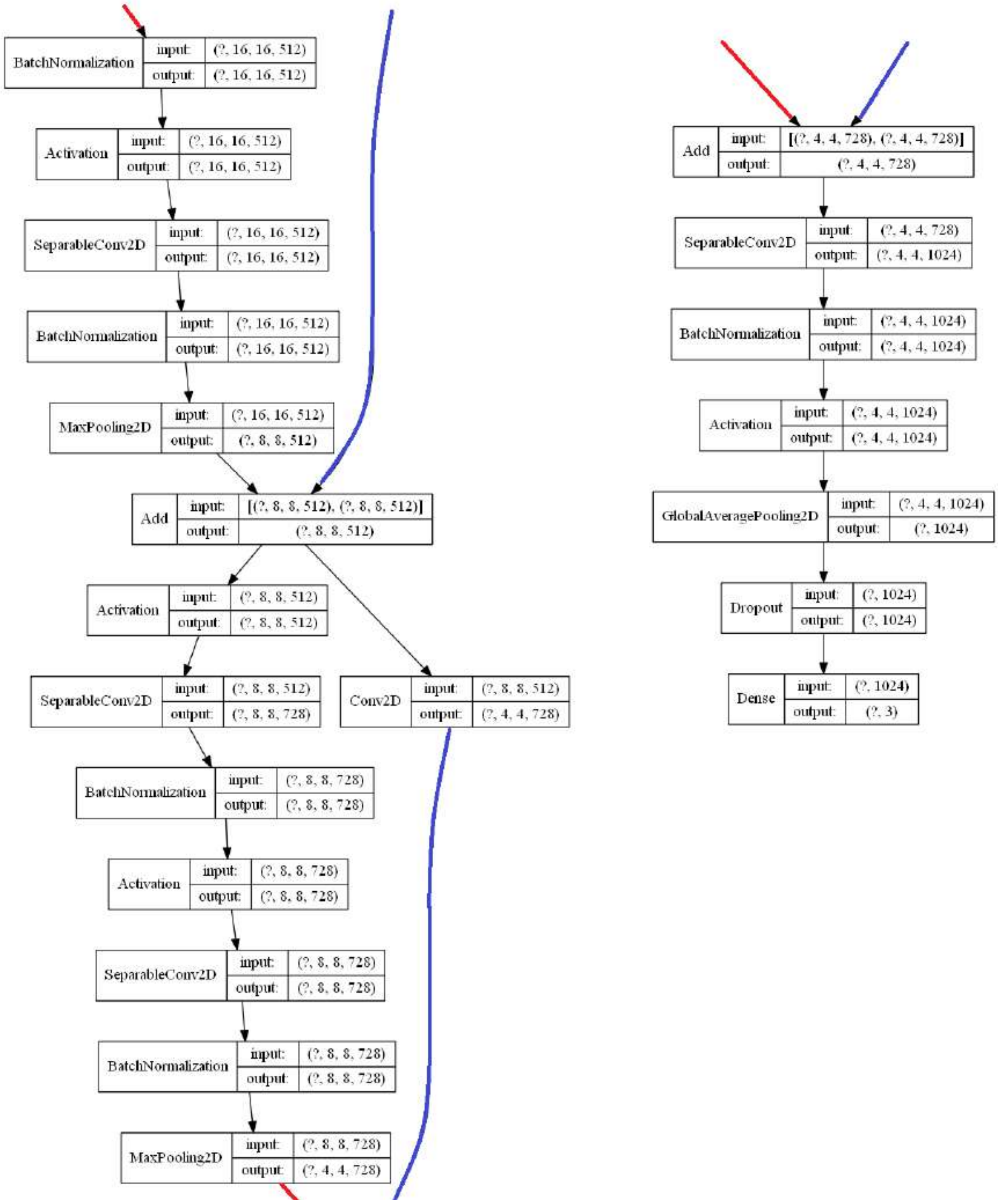
Scene Interpreter Architectures

B.1 Simple Model

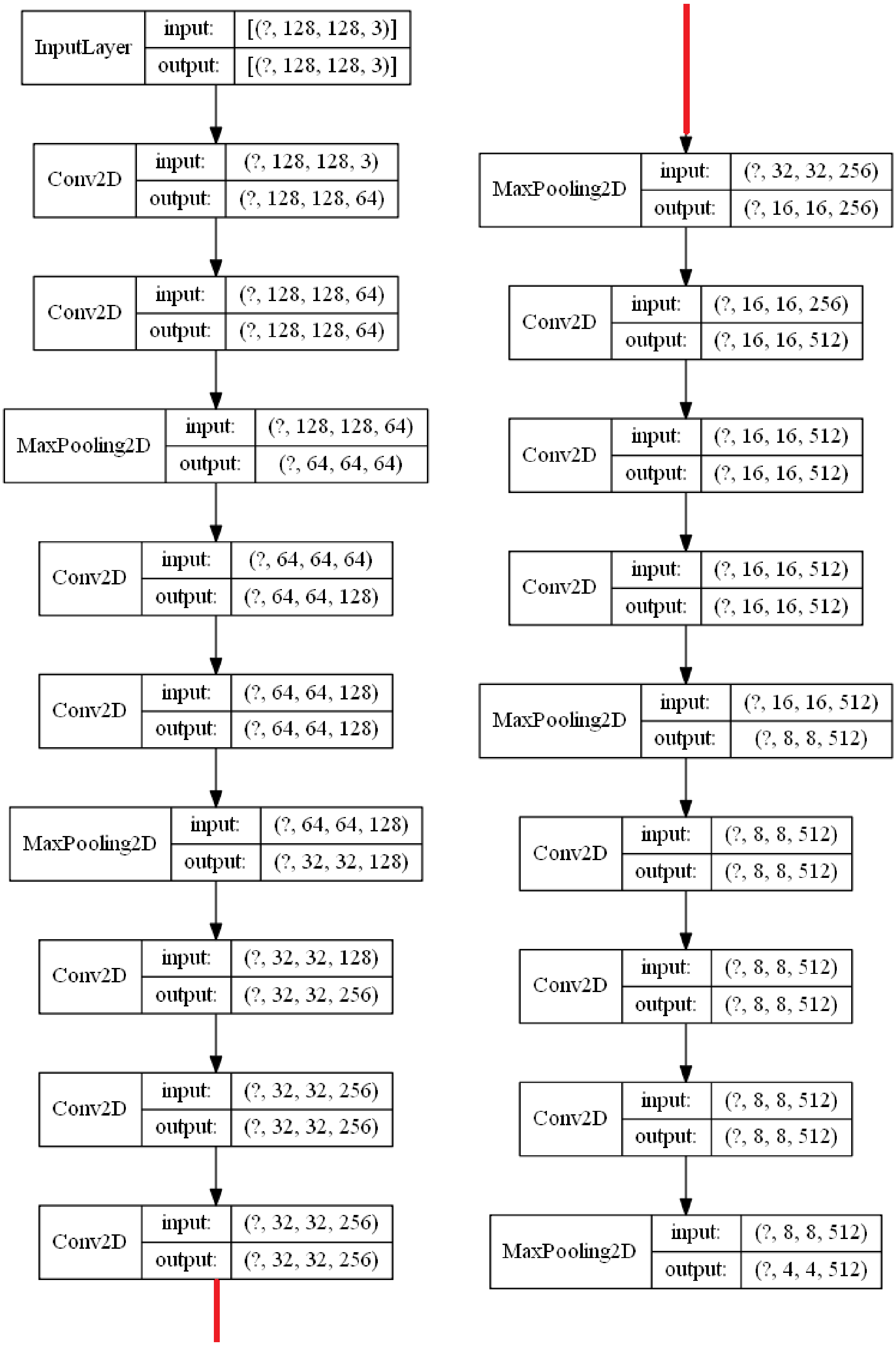


B.2 Complex Model

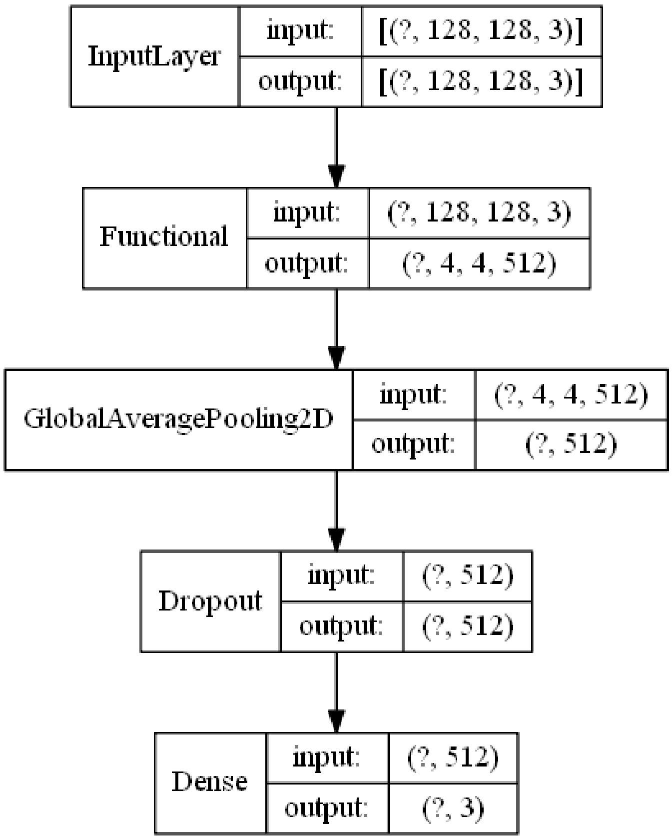



BatchNormalization
input: (?, 16, 16, 512)
output: (?, 16, 16, 512)
Activation
input: (?, 16, 16, 512)
output: (?, 16, 16, 512)
SeparableConv2D
input: (?, 16, 16, 512)
output: (?, 16, 16, 512)
BatchNormalization
input: (?, 16, 16, 512)
output: (?, 16, 16, 512)
MaxPooling2D
input: (?, 16, 16, 512)
output: (?, 8, 8, 512)
Add
input: [(?, 8, 8, 512), (?, 8, 8, 512)]
output: (?, 8, 8, 512)
Activation
input: (?, 8, 8, 512)
output: (?, 8, 8, 512)
SeparableConv2D
input: (?, 8, 8, 512)
output: (?, 8, 8, 728)
Conv2D
input: (?, 8, 8, 512)
output: (?, 4, 4, 728)
BatchNormalization
input: (?, 8, 8, 728)
output: (?, 8, 8, 728)
Activation
input: (?, 8, 8, 728)
output: (?, 8, 8, 728)
SeparableConv2D
input: (?, 8, 8, 728)
output: (?, 8, 8, 728)
BatchNormalization
input: (?, 8, 8, 728)
output: (?, 8, 8, 728)
MaxPooling2D
input: (?, 8, 8, 728)
output: (?, 4, 4, 728)
Add
input: [(?, 4, 4, 728), (?, 4, 4, 728)]
output: (?, 4, 4, 728)
SeparableConv2D
input: (?, 4, 4, 728)
output: (?, 4, 4, 1024)
BatchNormalization
input: (?, 4, 4, 1024)
output: (?, 4, 4, 1024)
Activation
input: (?, 4, 4, 1024)
output: (?, 4, 4, 1024)
GlobalAveragePooling2D
input: (?, 4, 4, 1024)
output: (?, 1024)
Dropout
input: (?, 1024)
output: (?, 1024)
Dense
input: (?, 1024)
output: (?, 3)

B.3 VGG16 Base Model



B.3.1 VGG16 Extended Model



Appendix C

Scene Interpreter Test Dataset

fish

Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

fish

Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

fish

Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

fish

Class	Aug.	No Aug.
Fish	1.00	0.99
Net	0.00	0.01
Non-sense	0.00	0.00

fish

Class	Aug.	No Aug.
Fish	1.00	0.88
Net	0.00	0.12
Non-sense	0.00	0.00

fish

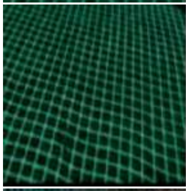
Class	Aug.	No Aug.
Fish	1.00	0.95
Net	0.00	0.00
Non-sense	0.00	0.05

fish

Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

net

Class	Aug.	No Aug.
Fish	1.00	0.01
Net	0.00	0.99
Non-sense	0.00	0.00



net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

net

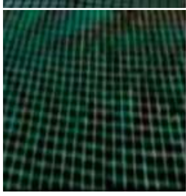
Class	Aug.	No Aug.
Fish	0.00	0.40
Net	1.00	0.60
Non-sense	0.00	0.00

net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

net

Class	Aug.	No Aug.
Fish	0.00	0.04
Net	1.00	0.96
Non-sense	0.00	0.00



net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

nonsense

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00

nonsense

Class	Aug.	No Aug.
Fish	0.00	0.98
Net	0.00	0.00
Non-sense	1.00	0.02

nonsense

Class	Aug.	No Aug.
Fish	0.00	0.42
Net	0.00	0.00
Non-sense	1.00	0.58

nonsense

Class	Aug.	No Aug.
Fish	0.03	0.11
Net	0.00	0.30
Non-sense	0.97	0.58

nonsense

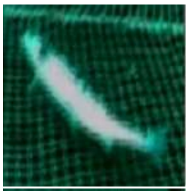
Class	Aug.	No Aug.
Fish	0.01	0.00
Net	0.00	0.00
Non-sense	0.99	1.00

nonsense

Class	Aug.	No Aug.
Fish	0.29	0.05
Net	0.05	0.00
Non-sense	0.66	0.95

nonsense

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00



fish

Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

fish

Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

fish

Class	Aug.	No Aug.
Fish	1.00	0.98
Net	0.00	0.00
Non-sense	0.00	0.02

fish

Class	Aug.	No Aug.
Fish	1.00	0.90
Net	0.00	0.00
Non-sense	0.00	0.10

fish

Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

fish

Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

fish

Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

fish

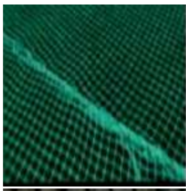
Class	Aug.	No Aug.
Fish	0.95	0.61
Net	0.00	0.00
Non-sense	0.05	0.39

fish

Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

fish

Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

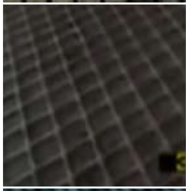


net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

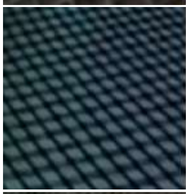
net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00



net

Class	Aug.	No Aug.
Fish	0.10	0.00
Net	0.90	1.00
Non-sense	0.00	0.00

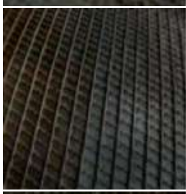


net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00



net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

net

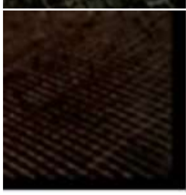
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

net

Class	Aug.	No Aug.
Fish	0.00	0.50
Net	1.00	0.50
Non-sense	0.00	0.00

net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00



net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

nonsense

Class	Aug.	No Aug.
Fish	1.00	0.33
Net	0.00	0.00
Non-sense	0.00	0.67

nonsense

Class	Aug.	No Aug.
Fish	1.00	0.03
Net	0.00	0.00
Non-sense	0.00	0.97

nonsense

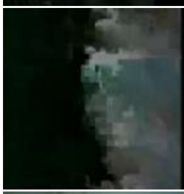
Class	Aug.	No Aug.
Fish	1.00	0.05
Net	0.00	0.00
Non-sense	0.00	0.95

nonsense

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.03
Non-sense	1.00	0.96

nonsense

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00

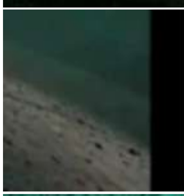


nonsense

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00

nonsense

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00



nonsense

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.02
Non-sense	1.00	0.98

nonsense

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00

nonsense

Class	Aug.	No Aug.
Fish	0.00	1.00
Net	0.22	0.00
Non-sense	0.78	0.00

fish

Class	Aug.	No Aug.
Fish	1.00	0.93
Net	0.00	0.00
Non-sense	0.00	0.07

fish

Class	Aug.	No Aug.
Fish	1.00	0.39
Net	0.00	0.00
Non-sense	0.00	0.61

fish

Class	Aug.	No Aug.
Fish	0.00	0.57
Net	0.00	0.00
Non-sense	1.00	0.43

fish

Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

fish

Class	Aug.	No Aug.
Fish	1.00	0.23
Net	0.00	0.00
Non-sense	0.00	0.77

fish

Class	Aug.	No Aug.
Fish	1.00	0.99
Net	0.00	0.00
Non-sense	0.00	0.01

fish

Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

fish

Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

fish

Class	Aug.	No Aug.
Fish	0.96	0.99
Net	0.00	0.00
Non-sense	0.04	0.01

fish

Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

net

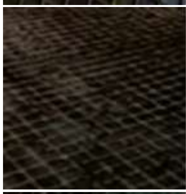
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00



net

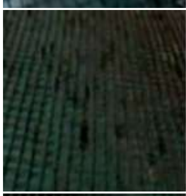
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

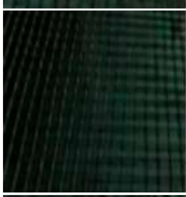
net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00



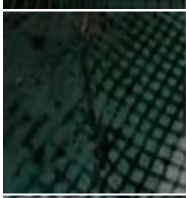
net

Class	Aug.	No Aug.
Fish	0.00	0.01
Net	1.00	0.99
Non-sense	0.00	0.00



net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00



net

Class	Aug.	No Aug.
Fish	0.00	0.01
Net	1.00	0.99
Non-sense	0.00	0.00

net

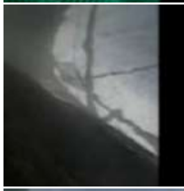
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

nonsense

Class	Aug.	No Aug.
Fish	0.95	0.12
Net	0.00	0.00
Non-sense	0.05	0.88

nonsense

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.02	0.00
Non-sense	0.98	1.00



nonsense

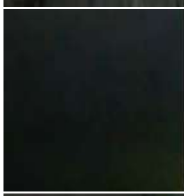
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00

nonsense

Class	Aug.	No Aug.
Fish	0.02	0.00
Net	0.00	0.00
Non-sense	0.98	1.00

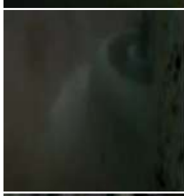
nonsense

Class	Aug.	No Aug.
Fish	0.00	0.01
Net	0.00	0.00
Non-sense	1.00	0.99



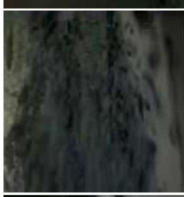
nonsense

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00



nonsense

Class	Aug.	No Aug.
Fish	1.00	0.01
Net	0.00	0.00
Non-sense	0.00	0.99

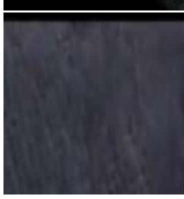


nonsense

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00

nonsense

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00



nonsense

Class	Aug.	No Aug.
Fish	0.00	0.01
Net	1.00	0.56
Non-sense	0.00	0.42

fish

Class	Aug.	No Aug.
Fish	1.00	0.66
Net	0.00	0.00
Non-sense	0.00	0.34

fish

Class	Aug.	No Aug.
Fish	1.00	0.99
Net	0.00	0.00
Non-sense	0.00	0.01

fish

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00

fish

Class	Aug.	No Aug.
Fish	1.00	0.30
Net	0.00	0.01
Non-sense	0.00	0.69

fish

Class	Aug.	No Aug.
Fish	0.67	0.96
Net	0.00	0.00
Non-sense	0.33	0.04

fish

Class	Aug.	No Aug.
Fish	0.00	0.02
Net	0.00	0.00
Non-sense	1.00	0.98

fish

Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

fish

Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

fish

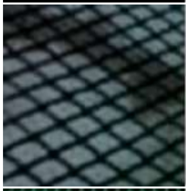
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00

fish

Class	Aug.	No Aug.
Fish	0.00	0.89
Net	0.00	0.00
Non-sense	1.00	0.11

net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00



net

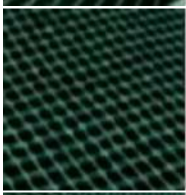
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

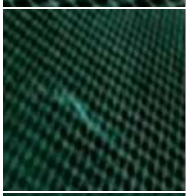
net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00



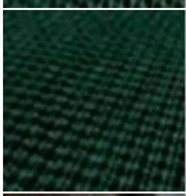
net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00



net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

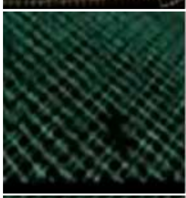


net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

net

Class	Aug.	No Aug.
Fish	1.00	0.96
Net	0.00	0.04
Non-sense	0.00	0.00



net

Class	Aug.	No Aug.
Fish	0.44	0.00
Net	0.56	1.00
Non-sense	0.00	0.00

net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

nonsense

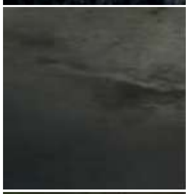
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00

nonsense

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00

nonsense

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00



nonsense

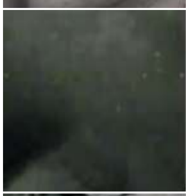
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00

nonsense

Class	Aug.	No Aug.
Fish	0.00	0.51
Net	0.00	0.00
Non-sense	1.00	0.49

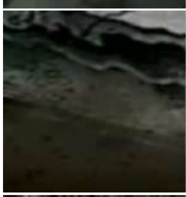
nonsense

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00



nonsense

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00

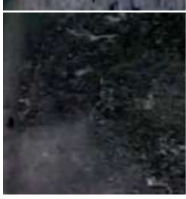


nonsense

Class	Aug.	No Aug.
Fish	0.00	0.10
Net	0.00	0.00
Non-sense	1.00	0.90

nonsense

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00



nonsense

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	0.99

fish

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00

fish

Class	Aug.	No Aug.
Fish	1.00	0.01
Net	0.00	0.00
Non-sense	0.00	0.99

fish

Class	Aug.	No Aug.
Fish	0.00	0.32
Net	0.00	0.00
Non-sense	1.00	0.68

fish

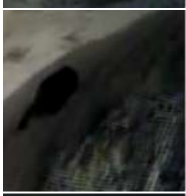
Class	Aug.	No Aug.
Fish	1.00	0.96
Net	0.00	0.00
Non-sense	0.00	0.04

fish

Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

fish

Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00



fish

Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

fish

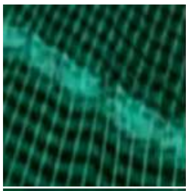
Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

fish

Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

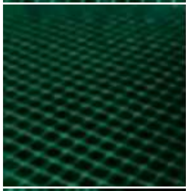
fish

Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00



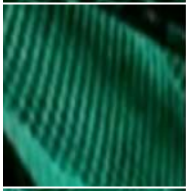
net

Class	Aug.	No Aug.
Fish	0.00	0.03
Net	1.00	0.97
Non-sense	0.00	0.00



net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00



net

Class	Aug.	No Aug.
Fish	0.00	1.00
Net	1.00	0.00
Non-sense	0.00	0.00

net

Class	Aug.	No Aug.
Fish	0.00	0.90
Net	1.00	0.10
Non-sense	0.00	0.00

net

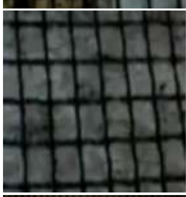
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

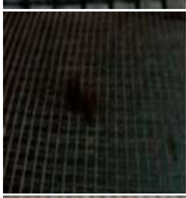
net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00



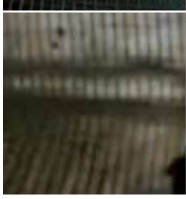
net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00



net

Class	Aug.	No Aug.
Fish	1.00	0.01
Net	0.00	0.99
Non-sense	0.00	0.00



net

Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

nonsense

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00

nonsense

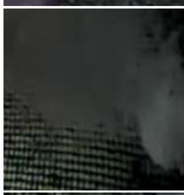
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00

nonsense

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00

nonsense

Class	Aug.	No Aug.
Fish	0.00	0.04
Net	0.00	0.73
Non-sense	1.00	0.23



nonsense

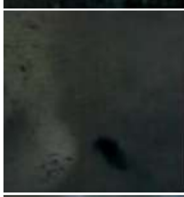
Class	Aug.	No Aug.
Fish	0.01	0.96
Net	0.99	0.01
Non-sense	0.00	0.03

nonsense

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	0.99

nonsense

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00

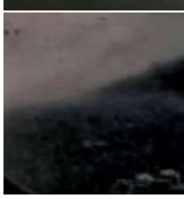


nonsense

Class	Aug.	No Aug.
Fish	1.00	0.12
Net	0.00	0.00
Non-sense	0.00	0.88

nonsense

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00



nonsense

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00

fish

Class	Aug.	No Aug.
Fish	1.00	0.95
Net	0.00	0.05
Non-sense	0.00	0.00

fish

Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

fish

Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

fish

Class	Aug.	No Aug.
Fish	1.00	0.15
Net	0.00	0.76
Non-sense	0.00	0.09

fish

Class	Aug.	No Aug.
Fish	0.03	1.00
Net	0.00	0.00
Non-sense	0.97	0.00

fish

Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

fish

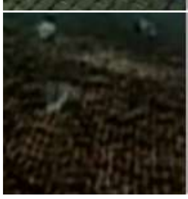
Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

fish

Class	Aug.	No Aug.
Fish	1.00	0.98
Net	0.00	0.02
Non-sense	0.00	0.00

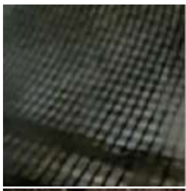
fish

Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00



fish

Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00



net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

net

Class	Aug.	No Aug.
Fish	0.02	0.08
Net	0.98	0.92
Non-sense	0.00	0.00

net

Class	Aug.	No Aug.
Fish	0.74	0.10
Net	0.26	0.90
Non-sense	0.00	0.00

net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

net

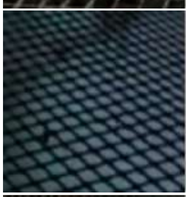
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00



net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

net

Class	Aug.	No Aug.
Fish	1.00	0.01
Net	0.00	0.99
Non-sense	0.00	0.00

net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

nonsense

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00

nonsense

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	0.00
Non-sense	0.00	1.00

nonsense

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00

nonsense

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00

nonsense

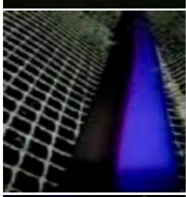
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	0.85
Non-sense	0.00	0.14

nonsense

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00

nonsense

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00



nonsense

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.11	1.00
Non-sense	0.89	0.00

nonsense

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00

nonsense

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00

fish		
Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

fish		
Class	Aug.	No Aug.
Fish	1.00	0.53
Net	0.00	0.47
Non-sense	0.00	0.00

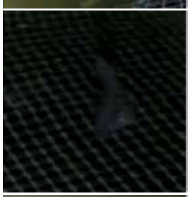
fish		
Class	Aug.	No Aug.
Fish	1.00	0.77
Net	0.00	0.23
Non-sense	0.00	0.00

fish		
Class	Aug.	No Aug.
Fish	0.73	0.85
Net	0.00	0.00
Non-sense	0.27	0.15

fish		
Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

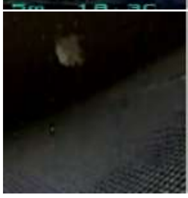
fish		
Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

fish		
Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00



fish		
Class	Aug.	No Aug.
Fish	0.39	0.00
Net	0.61	1.00
Non-sense	0.00	0.00

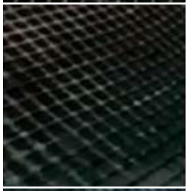
fish		
Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00



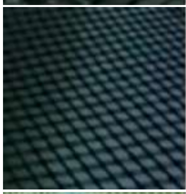
fish		
Class	Aug.	No Aug.
Fish	1.00	0.99
Net	0.00	0.00
Non-sense	0.00	0.01

net		
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

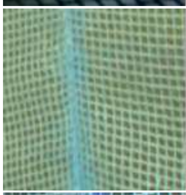
net		
Class	Aug.	No Aug.
Fish	0.00	0.04
Net	1.00	0.96
Non-sense	0.00	0.00



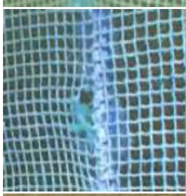
net		
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00



net		
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

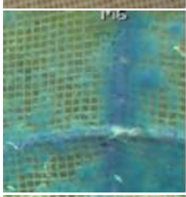


net		
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

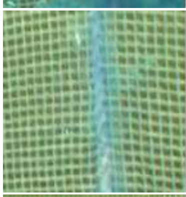


net		
Class	Aug.	No Aug.
Fish	0.01	0.12
Net	0.99	0.88
Non-sense	0.00	0.00

net		
Class	Aug.	No Aug.
Fish	1.00	0.96
Net	0.00	0.04
Non-sense	0.00	0.00



net		
Class	Aug.	No Aug.
Fish	0.00	0.01
Net	1.00	0.99
Non-sense	0.00	0.00



net		
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

net		
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

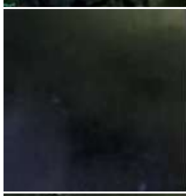
nonsense		
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00

nonsense		
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00

nonsense		
Class	Aug.	No Aug.
Fish	1.00	0.00
Net	0.00	0.00
Non-sense	0.00	1.00

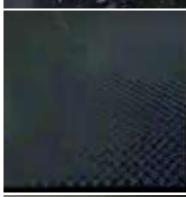
nonsense		
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00

nonsense		
Class	Aug.	No Aug.
Fish	0.00	0.01
Net	1.00	0.01
Non-sense	0.00	0.98



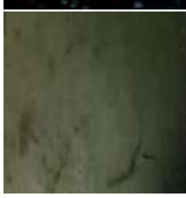
nonsense		
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00

nonsense		
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00



nonsense		
Class	Aug.	No Aug.
Fish	0.00	0.07
Net	0.99	0.89
Non-sense	0.00	0.04

nonsense		
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.05	0.00
Non-sense	0.95	1.00



nonsense		
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00

fish

Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

fish

Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

fish

Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

fish

Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

fish

Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

fish

Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

fish

Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

fish

Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

fish

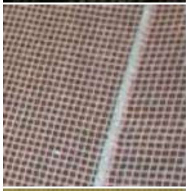
Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

fish

Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

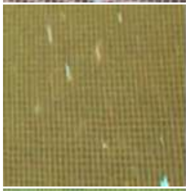
net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00



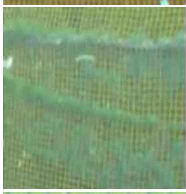
net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00



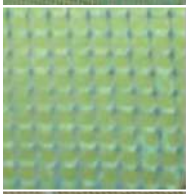
net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00



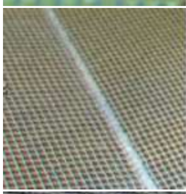
net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00



net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	0.99
Non-sense	0.00	0.01



net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

net

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

net

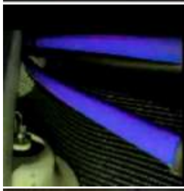
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

nonsense

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00

nonsense

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00



nonsense

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00

nonsense

Class	Aug.	No Aug.
Fish	0.00	1.00
Net	0.00	0.00
Non-sense	1.00	0.00

nonsense

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00

nonsense

Class	Aug.	No Aug.
Fish	0.00	0.03
Net	0.40	0.79
Non-sense	0.60	0.18

nonsense

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00

nonsense

Class	Aug.	No Aug.
Fish	0.00	0.01
Net	0.00	0.00
Non-sense	1.00	0.99

nonsense

Class	Aug.	No Aug.
Fish	0.00	0.02
Net	0.00	0.00
Non-sense	1.00	0.98

nonsense

Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00

fish		
Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

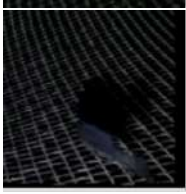
fish		
Class	Aug.	No Aug.
Fish	1.00	0.98
Net	0.00	0.00
Non-sense	0.00	0.02

fish		
Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

fish		
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

fish		
Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

fish		
Class	Aug.	No Aug.
Fish	1.00	0.91
Net	0.00	0.09
Non-sense	0.00	0.00



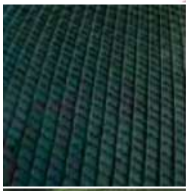
fish		
Class	Aug.	No Aug.
Fish	1.00	0.86
Net	0.00	0.14
Non-sense	0.00	0.00

fish		
Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

fish		
Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

fish		
Class	Aug.	No Aug.
Fish	1.00	1.00
Net	0.00	0.00
Non-sense	0.00	0.00

net		
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00



net		
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

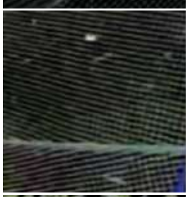
net		
Class	Aug.	No Aug.
Fish	0.00	0.90
Net	1.00	0.10
Non-sense	0.00	0.00

net		
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

net		
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

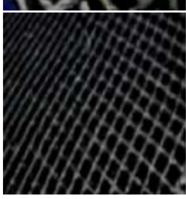
net		
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

net		
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00



net		
Class	Aug.	No Aug.
Fish	0.00	0.16
Net	1.00	0.84
Non-sense	0.00	0.00

net		
Class	Aug.	No Aug.
Fish	0.00	0.02
Net	1.00	0.98
Non-sense	0.00	0.00



net		
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	1.00	1.00
Non-sense	0.00	0.00

nonsense		
Class	Aug.	No Aug.
Fish	0.00	0.03
Net	0.00	0.01
Non-sense	1.00	0.97

nonsense		
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00

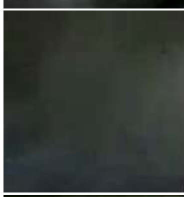
nonsense		
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00

nonsense		
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00

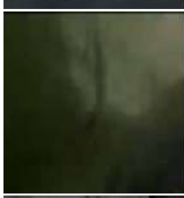
nonsense		
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00

nonsense		
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00

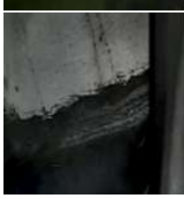
nonsense		
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00



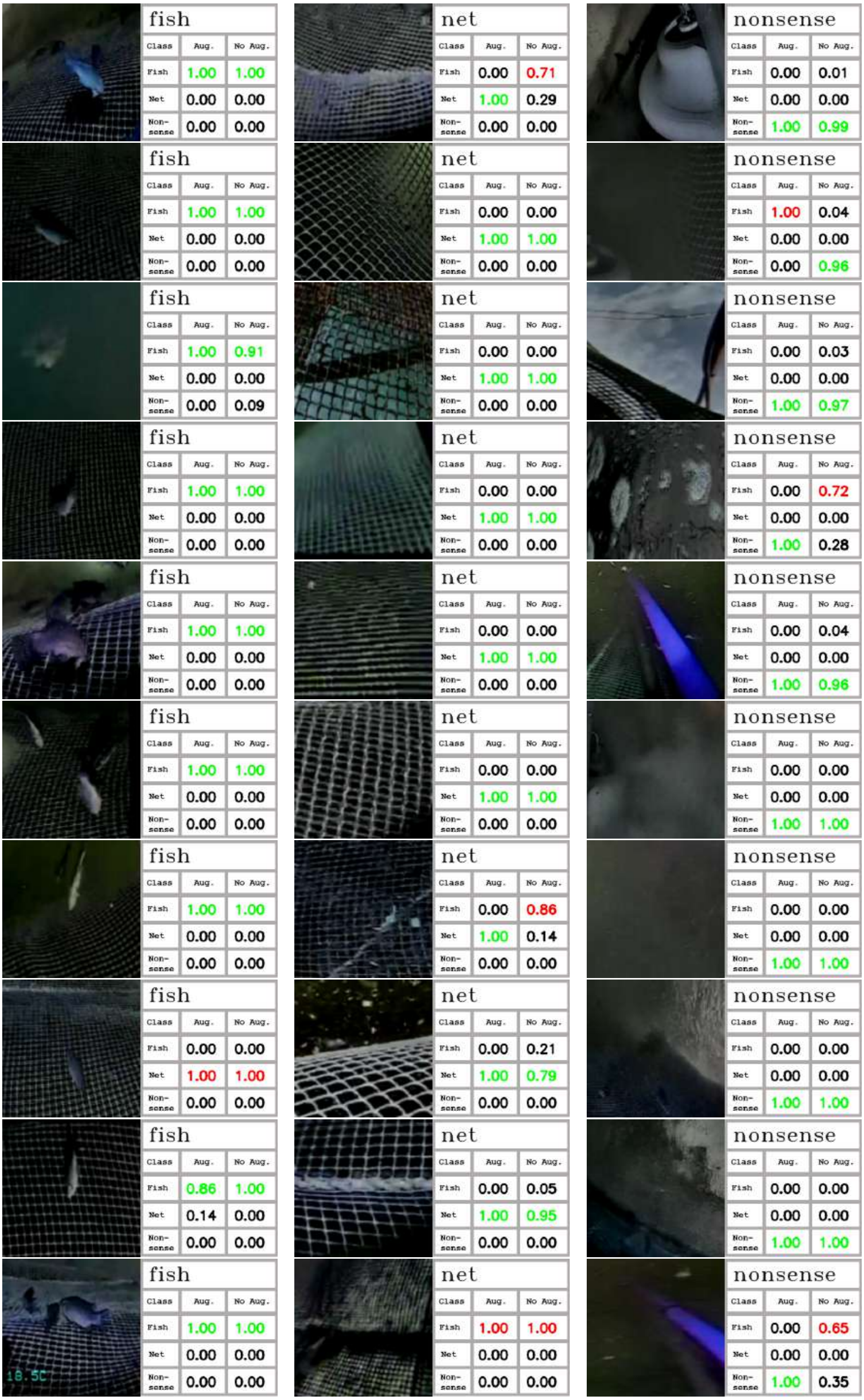
nonsense		
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00

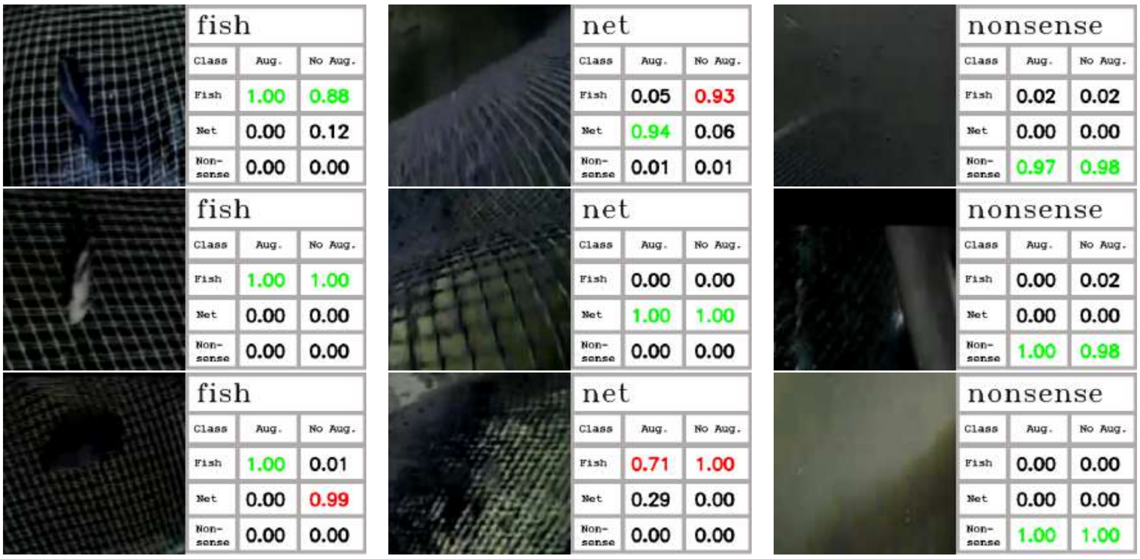


nonsense		
Class	Aug.	No Aug.
Fish	0.01	0.00
Net	0.00	0.00
Non-sense	0.99	1.00



nonsense		
Class	Aug.	No Aug.
Fish	0.00	0.00
Net	0.00	0.00
Non-sense	1.00	1.00


fish
Class Aug. No Aug.
Fish 1.00 1.00
Net 0.00 0.00
Non-sense 0.00 0.00
net
Class Aug. No Aug.
Fish 0.00 0.71
Net 1.00 0.29
Non-sense 0.00 0.00
nonsense
Class Aug. No Aug.
Fish 0.00 0.01
Net 0.00 0.00
Non-sense 1.00 0.99
fish
Class Aug. No Aug.
Fish 1.00 1.00
Net 0.00 0.00
Non-sense 0.00 0.00
net
Class Aug. No Aug.
Fish 0.00 0.00
Net 1.00 1.00
Non-sense 0.00 0.00
nonsense
Class Aug. No Aug.
Fish 1.00 0.04
Net 0.00 0.00
Non-sense 0.00 0.96
fish
Class Aug. No Aug.
Fish 1.00 0.91
Net 0.00 0.00
Non-sense 0.00 0.09
net
Class Aug. No Aug.
Fish 0.00 0.00
Net 1.00 1.00
Non-sense 0.00 0.00
nonsense
Class Aug. No Aug.
Fish 0.00 0.03
Net 0.00 0.00
Non-sense 1.00 0.97
fish
Class Aug. No Aug.
Fish 1.00 1.00
Net 0.00 0.00
Non-sense 0.00 0.00
net
Class Aug. No Aug.
Fish 0.00 0.00
Net 1.00 1.00
Non-sense 0.00 0.00
nonsense
Class Aug. No Aug.
Fish 0.00 0.72
Net 0.00 0.00
Non-sense 1.00 0.28
fish
Class Aug. No Aug.
Fish 1.00 1.00
Net 0.00 0.00
Non-sense 0.00 0.00
net
Class Aug. No Aug.
Fish 0.00 0.00
Net 1.00 1.00
Non-sense 0.00 0.00
nonsense
Class Aug. No Aug.
Fish 0.00 0.04
Net 0.00 0.00
Non-sense 1.00 0.96
fish
Class Aug. No Aug.
Fish 1.00 1.00
Net 0.00 0.00
Non-sense 0.00 0.00
net
Class Aug. No Aug.
Fish 0.00 0.00
Net 1.00 1.00
Non-sense 0.00 0.00
nonsense
Class Aug. No Aug.
Fish 0.00 0.00
Net 0.00 0.00
Non-sense 1.00 1.00
fish
Class Aug. No Aug.
Fish 1.00 1.00
Net 0.00 0.00
Non-sense 0.00 0.00
net
Class Aug. No Aug.
Fish 0.00 0.86
Net 1.00 0.14
Non-sense 0.00 0.00
nonsense
Class Aug. No Aug.
Fish 0.00 0.00
Net 0.00 0.00
Non-sense 1.00 1.00
fish
Class Aug. No Aug.
Fish 0.00 0.00
Net 1.00 1.00
Non-sense 0.00 0.00
net
Class Aug. No Aug.
Fish 0.00 0.21
Net 1.00 0.79
Non-sense 0.00 0.00
nonsense
Class Aug. No Aug.
Fish 0.00 0.00
Net 0.00 0.00
Non-sense 1.00 1.00
fish
Class Aug. No Aug.
Fish 0.86 1.00
Net 0.14 0.00
Non-sense 0.00 0.00
net
Class Aug. No Aug.
Fish 0.00 0.05
Net 1.00 0.95
Non-sense 0.00 0.00
nonsense
Class Aug. No Aug.
Fish 0.00 0.00
Net 0.00 0.00
Non-sense 1.00 1.00
18.5C
fish
Class Aug. No Aug.
Fish 1.00 1.00
Net 0.00 0.00
Non-sense 0.00 0.00
net
Class Aug. No Aug.
Fish 1.00 1.00
Net 0.00 0.00
Non-sense 0.00 0.00
nonsense
Class Aug. No Aug.
Fish 0.00 0.65
Net 0.00 0.00
Non-sense 1.00 0.35


fish
Class Aug. No Aug.
Fish 1.00 0.88
Net 0.00 0.12
Non-sense 0.00 0.00
net
Class Aug. No Aug.
Fish 0.05 0.93
Net 0.94 0.06
Non-sense 0.01 0.01
nonsense
Class Aug. No Aug.
Fish 0.02 0.02
Net 0.00 0.00
Non-sense 0.97 0.98
fish
Class Aug. No Aug.
Fish 1.00 1.00
Net 0.00 0.00
Non-sense 0.00 0.00
net
Class Aug. No Aug.
Fish 0.00 0.00
Net 1.00 1.00
Non-sense 0.00 0.00
nonsense
Class Aug. No Aug.
Fish 0.00 0.02
Net 0.00 0.00
Non-sense 1.00 0.98
fish
Class Aug. No Aug.
Fish 1.00 0.01
Net 0.00 0.99
Non-sense 0.00 0.00
net
Class Aug. No Aug.
Fish 0.71 1.00
Net 0.29 0.00
Non-sense 0.00 0.00
nonsense
Class Aug. No Aug.
Fish 0.00 0.00
Net 0.00 0.00
Non-sense 1.00 1.00

Appendix D

Siamese Network for Scene Similarity Detection

Two pre-trained VGG16 models (with frozen weights) were run in parallel on two input images. Their output layers were removed, and their feature space before that layer were compared with a Euclidean distance measure. A single sigmoid was trained to output 0 if the two input images were dissimilar, and 1 if they were similar.

A relatively small dataset of 100 scenes (with at least two snapshots from each scene) was constructed. During training, half of the images (called *anchors*) were coupled with their true counterpart (called the *positive*), and the other half, with a randomly chosen anchor (playing the role as the *negative*). See fig. D.1 for examples. Anchors from the validation set were never used as negatives for training data, and vice versa.

After 100 epochs it seemed that further training was quite futile. Some learning was probably achieved, especially in one fold which achieved a validation accuracy of 88% for some epoch (see fig. D.2). Testing was merely conducted on a preliminary stage and not further pursued.

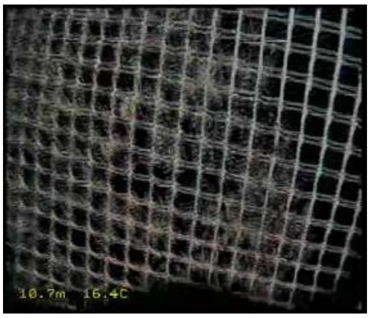

(a) Anchor


(b) Positive


(c) Negative


(d) Anchor


(e) Positive



(f) Negative

Figure D.1: Sample images from the foundation dataset.

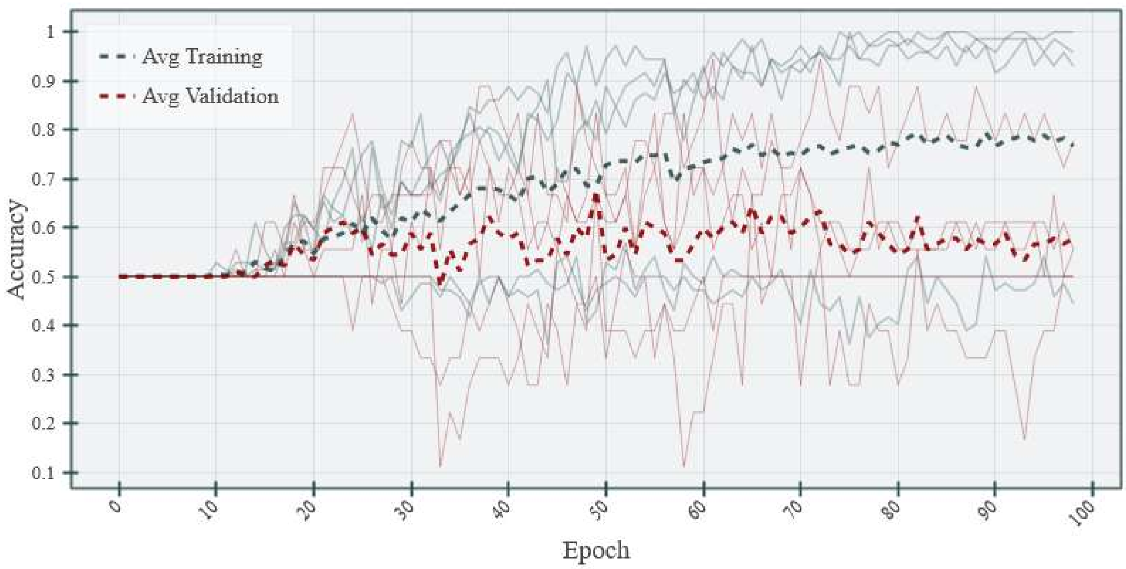


Figure D.2: Training and validation accuracies for the first 100 epochs of training indicated some learning, but not sufficient.