UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

Towards identification of *Octopus cyanea* behaviour in its natural habitat

Martim Duarte da Costa Seco

Mestrado em Informática

Dissertação orientada por:
Professor Doutor Luís Correia, Professor Associado, Universidade de Lisboa
Professor Doutor Rui Rosa, Professor Auxiliar, Universidade de Lisboa

2019

Towards identification of *Octopus cyanea* behaviour in its natural habitat

This document was created using the (pdf)LaTeX processor, based on the "novathesis" template[1], developed at the Dep. Informática of FCT-NOVA [2].
[1] https://github.com/joaomlourenco/novathesis [2] http://www.di.fct.unl.pt

Acknowledgements

I would like to express my gratitude to Professor Rui Rosa who encouraged me to pursue a master degree in the computer science area. Also for supervising me and providing me the opportunity to develop this thesis at the *Laboratorio Maritimo da Guia*.

I would like to offer my very great appreciation to Professor Luís Correia that accepted from day one, the challenge of who supervised me on developing a tracking system able to track octopuses in its natural environment.

I am particularly grateful for the friendship and all the help that Eduardo Sampaio gave me throughout the duration of this thesis and for providing me with the opportunity to have contact with the fieldwork beyond having the video data.

I would also like to thank my colleges at *Laboratorio Maritimo da Guia* for the good work environment and all the insightful conversations.

I wish to thank to João Lourenço for designing and maintaining the LaTeX template NOVAthesis used to write this document.

I wish to thank my parents and sisters for their support, patience and encouragement throughout this period.

Finally, I would like to give my special thanks to my girlfriend Julia Mallen for encouraging me in doing what I like, for the patience during all this time and especially for never giving up.

Abstract

Octopuses are known to hunt together with fish and the quantification of such interspecific interactions has been performed with manual processing of single video recordings. Besides being a very time-consuming task it also does not provide a reliable way of registration for data analysis, especially in unstructured environments. Within this context, this dissertation aims to develop a semi-automated tracking system to observe and record cooperative behaviour events of a tropical invertebrate mollusc - the big blue octopus (*Octopus cyanea*). This dissertation also includes fieldwork for capturing stereoscopic videos of octopuses' behaviours with a pair of underwater cameras, and extracted three-dimensional location information from them with an intelligent tracking system. A central component of the solution is an already existing tracking system specialized in structured subjects, with well-defined body parts, — the DeepLabCut (DLC). DLC was never used on octopuses, a kind of animal that easily changes shape, colour, and texture. Thus, a significant part of this dissertation is directed at the identification of parameter values for the tracking algorithm. A successful usage pipeline and parametrization of a neural network (of DLC) that automatically tracks an octopus in a variety of situations in several minutes of video was achieved, requiring only the manual labelling of a small part of its frames.

Keywords: *Octopus cyanea*, Neural-Network, Tracking Systems, DeepLabCut, Underwater Video, Natural environment.

Resumo

Octopus cyanea é uma espécie de polvos cujas capacidades de camuflagem evoluiram durante 270 milhões de anos no sentido de desenvolver uma técnica de caça bastante eficaz e evitar ser detectado e perseguido por predadores. Os polvos são conhecidos por mudar de cor, forma e textura consoante as condições ambientais e o contexto (alerta, ameaça ou alimentação). Estes animais têm também uma extraordinária flexibilidade corporal, permitindos passar por orifícios tão pequenos como 17mm Embora este animais sejam conhecidos por serem solitários exibem também comportamentos cooperativos com peixes em situação de caça. Alguns dos comportamentos que podem ser observados aquando da caça cooperativa são o comportamento Web Over, o comportamento Crawl, o comportamento Jump, e o Punch (um comportamento novo observado no decurso deste trabalho). Estes comportamentos notavelmente influencia a cor, a forma e a textura dos polvos, condições que precisam de ser tidas em conta aquando da análise de imagens e vídeos.

Tanto quanto nos é dado a conhecer, não há ferramenta open-source que sejam capazes de seguir polvos de forma não intrusiva no seu ambiente natural. A maior parte dos trabalhos na literatura atual baseiam-se na interpretação e medidas feitas manualmente sobre vídeos singulares. O maior problema com este método é a sua morosidade e subjectividade. Tais métodos não fornecem uma forma fiável de registo e análise de dados. Ainda mais se a análise de depender da localização de marcos no ambiente ou na trajetória de sujeitos em ambientes não estruturados. Para além disso, as condições necessárias para observar um comportamento coletivo entre um polvo e peixes são muito difíceis de replicar em ambientes controlados como aquários. Para capturar tais comportamentos em vídeo, é necessário trabalhar no ambiente natural dos polvos, usando equipamento de mergulho e um equipamento de filmagem móvel.

No que diz respeito aos sistemas identificação e seguimento foram desenhados para trabalhar em ambientes estéreis e estruturados onde existe um fundo claro e simples e o animal pode ser seguido é escuro e com uma forma elíptica. No entanto, no seu habitat natural, muitas das características usadas na visão por computador como a cor e a forma são altamente variáveis. Ao fornecer uma ferramenta semi-automática que acelera este processo, e que automaticamente segue polvos num ambiente tridimensional, permite-se uma reconstrução de trajetória e análise de comportamentos mais rápida e automática. A

utilização de ferramentas tais a descrita neste documento poderá ajudar investigadores no estudo de animais marinhos no seu contexto natural, sem recurso a técnicas invasivas e onde a tecnologia de identificação e seguimento nunca foi aplicada. Esta solução permite que a análise do comportamento seja feita com um número muito mais reduzido de frames do que originalmente seria.

Os desafios com que nos deparamos ao longo deste trabalho advêm de estarmos a trabalhar com um animal como o polvo, um animal que se desenvolveu ao longo de milhões de anos para conseguir enganar os sistemas de identificação natural dos predadores naturais. Estes mecanismos não só enganam os predadores como também enganam as redes neuronais.

O principal objetivo deste trabalho é usar visão por computador para seguir um polvo (Octopus cyanea) no seu ambiente natural. O projeto onde este trabalho se encontra inserido tem por objetivo explorar o problema aberto de desenvolver um sistema semi-automático de seguimento que melhora o comportamento de análise de comportamento, para que este seja mais eficaz e rápido. Este objetivo é incorporado num sistema que reconstrói trajetórias de peixes e polvo, por sua vez este resultado pode ser usado como dado de entrada para uma análise de comportamento da dinâmica colaborativa do conjunto polvo, peixes. A visão final consiste num sistema completo cujos dados de entrada são vídeos tridimensionais subaquáticos e cujos resultados são etogramas representando o comportamento colaborativo de polvos e peixes.

Esta dissertação foca-se na parte inicial da análise completa descrita acima, no desenho de uma pipeline e a parametrização de uma rede neuronal que segue automaticamente um polvo numa variedade grande de situações durante alguns minutos de vídeo, requerendo para tal apenas a anotação de uma pequena parte das suas imagens. É objetivo deste trabalho a identificação de ferramentas base, e a respetiva seleção e parametrização, para se atingir os objetivos. Como resultado, obtém-se uma ferramenta, ou a configuração de uma ferramenta existente que ajudará investigadores na área da análise dos comportamentos subaquáticos, em particular, do comportamento do O. cyanea. Como um outro produto deste trabalho espera-se disponibilizar uma base robusta para desenvolvimentos futuros de sistemas de análise comportamental de animais.

No decurso deste trabalho foi efetuado trabalho de campo, onde foram utilizados dois tipos de câmaras subaquáticas, uma câmara tridimensional composta por duas câmaras num suporte fixo e uma câmara solta, potenciando vídeos com mais pormenor sobre os indivíduos. Foram registados 30 eventos isolados de comportamento cooperativo de caça, dos quais 8 foram anotados e analisados no decurso deste trabalho. Para cada um destes, a rede neuronal foi treinada e analisada tendo sido obtidos resultados que estão muito perto do erro humano existente no processo de anotação de vídeos. Foram estudadas estratégias de anotação de vídeo que vão desde anotar centenas de imagens num vídeo até a anotação de poucas dezenas de imagens para vídeos de 2 a 30 minutos. É de notar que os fatores de confiança da rede claramente apontam para resultados com muita precisão, sendo que foi também analisada a progressão de aprendizagem da rede neuronal, analisando snapshots

da rede em intervalos de algumas centenas de milhar de iterações.

Este trabalho contribui para a comunidade científica com uma base de dados de vídeos *raw* e de vídeos manualmente anotados de polvos, de polvos a cooperar com diferentes espécies de peixes diferentes. Estes vídeos poderão ser utilizados para testar novas abordagens de identificação e seguimento automático de polvos, tanto como conjunto de dados de treino mas também como conjunto de validação de verdade absoluta.

Além disto, contribuímos também com uma *pipeline* de processamento e análise de imagens e de vídeos. Esta *pipeline* é baseada na ferramenta DeepLabCut, desenhada para trabalhar com animais vertebrados em ambientes estruturados. São ainda apresentados neste documento o resultado da análise de vídeos em vários ambientes com a caracterização dos cenários onde se obtêm bons resultados e alertas para as situações onde os resultados são menos satisfatórios.

Como trabalho futuro vamos querer aplicar a rede neuronal e a *pipeline* descrita acima aos peixes que fazem parte do comportamento cooperativo alvo. Com isto pretendemos obter a identificação e o seguimento de todos os envolvidos nesta interação e futuramente através de um outro software fazer a reconstrução 3D do fundo do mar onde é decorrida esta interação. Quando obtivermos ambos vamos sobrepor os movimentos do polvo e dos restantes intervenientes, os peixes, no fundo do mar.

Pretendemos futuramente ser capazes de analisar o comportamento cooperativo como um todo e não como comportamentos individuais de cada animal. Pretendemos estudar quem é o leader da cooperação, se há preferência de indivíduos em detrimento de outros. Seria também interessante ser desenvolvido um software que fosse capaz de analisar os movimentos de todos os intervenientes e que tivesse como *output* final o comportamento de cada um dos intervenientes.

Palavras-chave: *Octopus cyanea*, Rede Neuronal, Identificação Automática , DeepLabCut, Video Subaquático, Ambiente Natural.

Contents

List of Figures xvii

List of Tables xix

Acronyms xxi

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Objectives . . . 2
- 1.3 Contributions . . . 2
- 1.4 Structure of the Dissertation . . . 3

2 State of the Art 5
- 2.1 Open Vision Control . . . 6
- 2.2 Tracktor . . . 7
- 2.3 IdTracker . . . 8
- 2.4 DeepLabCut . . . 9

3 Detailed Problem Description 11
- 3.1 Problem Statement . . . 11
- 3.2 Challenges . . . 11
  - 3.2.1 Natural environment . . . 12
  - 3.2.2 Underwater images . . . 12
  - 3.2.3 Moving camera . . . 13
  - 3.2.4 Octopus . . . 14
- 3.3 Requirements . . . 14

4 Materials and Methods 17
- 4.1 Video Acquisition and Video Analysis . . . 17
  - 4.1.1 Fieldwork Location . . . 17
  - 4.1.2 Hardware tools for video acquisition . . . 18
  - 4.1.3 Hardware tools for video analysis . . . 18
- 4.2 Tracking Tool Testing . . . 19
- 4.3 DeepLabCut based Pipeline . . . 19

4.3.1 Ffmpeg . . . 20
4.3.2 Fiji . . . 20
4.3.3 Pipeline . . . 20
4.3.4 DLC Parameters . . . 21
4.3.5 Frame Labelling Strategies . . . 25
4.3.6 Human labelling error . . . 26
4.3.7 Validation . . . 27
4.3.8 Zoomed-in Tracking . . . 27

5 Results **29**
5.1 Filmed Events . . . 29
5.2 Tracktor Results . . . 30
5.3 DeepLabCut Parameter Comparison . . . 31
5.3.1 Neural Network . . . 31
5.3.2 Global Scale . . . 31
5.3.3 Maximum Image Size . . . 32
5.3.4 Cropped Images . . . 33
5.3.5 P-Cutoff . . . 33
5.4 Results of Strategy 1 . . . 35
5.4.1 *MartimPedras1* . . . 36
5.4.2 MartimCorais . . . 38
5.5 Results of Strategy 2 . . . 40
5.5.1 *ZeSimao* . . . 40
5.5.2 *ZeSousa* . . . 42
5.5.3 *ZeManuel* . . . 44
5.5.4 *ZeMarco* . . . 46
5.6 Results of Strategy 3 . . . 49
5.6.1 *MartimPedras30* . . . 49
5.6.2 *ZeSimao30* . . . 51
5.7 Two videos, one NN . . . 52
5.8 Results of The Other Potentials Usages . . . 54
5.8.1 *MartimZoom* . . . 54
5.8.2 *ZeBrunoZoom* . . . 56

6 Conclusions **59**
6.1 Future Work . . . 60

References **61**

Annexes **65**

I Program used to extract random frames for validation **65**

II Validation Process **67**

List of Figures

2.1 Example of Open Vision Control following an object in video. 7
2.2 Example of tracktor following a fish in a noisy environment video. 8
2.3 Example of applying IdTracker on *drosophila* video. 9

3.1 Sterile Video Frame (left) vs. Non-sterile Video Frame (right). 12
3.2 Octopus occlusion due to a fish. 13
3.3 Octopus with different colours, shape and texture. 14

4.1 Stereocamera Rig used for dual acquisition of interspecies events in Eilat, Israel and in El Qoseir, Egypt. 18
4.2 Adopted DLC pipeline 20

5.1 ScreenShot of Tracktor running successfully 30
5.2 Comparison of Train and Test error of ResNet50 (left) and ResNet101 (right). 31
5.3 Comparison of Train and Test error of $GS = 0.8$ and $MIS = 1000$ (left) with $GS = 0.6$ and $MIS = 3200$ (right). 32
5.4 Comparison of Train and Test error of MIS = 1000 and MIS = 3200 33
5.5 Comparison of Train and Test error of *CROP* = TRUE (left) and *CROP* = FALSE (right) 34
5.6 Comparison of Train and Test error and Train and Test error with $p-cutoff = 0.1$ 34
5.7 Comparison of Train and Test error and Train and Test error with $p-cutoff = 0.95$ 35
5.8 Comparison between manually labelled (+) and DLC labelled (•) in *MartimPedras1* frame. The detail is zoomed in on the right. 36
5.9 Train and Test error, Train and Test error with $p-cutoff$ of *MartimPedras1*. . . 37
5.10 Comparison between manually labelled (+) and DLC labelled (•) in *MartimCorais* frame. 38
5.11 Train and Test error, Train and Test error with $p-cutoff$ of *MartimCorais*. . . . 39
5.12 Comparison between manually labelled (+) and DLC labelled (•) in *ZeSimao* frame. 41
5.13 Train and Test error, Train and Test error with $p-cutoff$ of *ZeSimao* with Stratagy 2. 41
5.14 Comparison between manually labelled (+) and DLC labelled (•) in *ZeSousa* frame. 43

5.15 Train and Test error, Train and Test error with *p–cutoff* of *ZeSousa*. 43
5.16 Comparison between manually labelled (+) and DLC labelled (•) in *ZeManuel* frame. 45
5.17 Train and Test error, Train and Test error with *p–cutoff* of *ZeManuel*. 46
5.18 Comparison between manually labelled (+) and DLC labelled (•) in *ZeMarco* frame. 47
5.19 Train and Test error, Train and Test error with *p–cutoff* of *ZeMarco*. 48
5.20 Train and Test error, Train and Test error with *p–cutoff* of *MartimPedras30*. . 50
5.21 Train and Test error, Train and Test error with *p–cutoff* of *ZeSimao30*. 51
5.22 Train and Test error, Train and Test error with *p–cutoff* of *ZeTudo*. 53
5.23 Comparison between manually labelled (+) and DLC labelled (•) in MartimZoom frame. 54
5.24 Train and Test error, Train and Test error with *p–cutoff* of MartimZoom. . . . 55
5.25 Comparison between manually labelled (+) and DLC labelled (•) in *ZeBrunoZoom* frame. 57
5.26 Train and Test error, Train and Test error with *p–cutoff* of *ZeBrunoZoom*. . . 58

List of Tables

2.1 DLC parameters 10

4.1 Summary of the different strategies of frame labelling. 26
4.2 Human Labelling Average Error in pixels. 26

5.1 Events and Corresponding MetaData. 29
5.2 Best values found for Tracktor parameters. 30
5.3 Standard Values for DLC Parameter Comparison. 31
5.4 Iteration 750000 of *MartimPedras1* - Train, Test and Validation Error Summary 37
5.5 Iteration 1030000 of *MartimCorais* - Train, Test and Validation Error Summary 39
5.6 Iteration 800000 of ZeSimao - Train, Test and Validation Error Summary. . . 42
5.7 Iteration 800000 of *ZeSousa* - Train, Test and Validation Error Summary. . . . 44
5.8 Iteration 500000 of *ZeManuel* - Train, Test and Validation Error Summary. . . 45
5.9 Iteration 1030000 of *ZeMarco* - Train, Test and Validation Error Summary. . . 48
5.10 Iteration 1000000 of *MartimPedras30* - Train, Test and Validation Error Summary. 50
5.11 Iteration 1030000 of *ZeSimao30* - Train, Test and Validation Error Summary. 52
5.12 Iteration 1030000 of *ZeTudo* - Train, Test and Validation Error Summary . . 53
5.13 Iteration 850000 of MartimZoom - Train, Test and Validation Error Summary. 56
5.14 Iteration 950000 of *ZeBrunoZoom* - Train, Test and Validation Error Summary. 57

II.1 Validation process of MartimCorais in order to explicate the process of validation of all the videos 68

Acronyms

CO - Individuals that cross and overlap each other

DLC - DeepLabCut

DNN - Deep Neural NetWork

KLT - Kanade–Lucas–Tomasi feature trackers

NN - Neural NetWork

ResNet - Residual Neural NetWork

Chapter 1

Introduction

1.1 Motivation

Octopus cyanea is an octopus species whose camouflage skills have evolved in the last 270 million years [2] towards an effective hunting technique, and to avoid being spotted and tracked by predators. Octopuses are known to change colour, shape and texture depending on the environmental conditions and its current situation (alerting, threat and feeding) [1]. Although these animals are known to be solitary animals, they are also known to cooperate with fish when hunting. Some behaviours that may be observed in cooperative hunting are the *Web Over*, the *Crawl* [10], the *Jump*, and the *Punch* (a newly observed behaviour, identified during the course of this work[1]). These behaviours notably influence the colour, shape and texture of the octopus, so we need to have them in mind when analysing images and videos.

To the extent of our knowledge, there are no open-source software tools that are able to track octopuses in its natural habitat. Most literature that targets octopuses and their behaviour is based on manual video interpretation and measurement. The main problem with this method is that manual video analysis is slow, subjective, and time-consuming. Such methods do not provide a reliable way of registration and analysis of data. More so, if the analysis depends on location of landmarks or trajectory of subjects in unstructured surroundings.

Moreover, the conditions necessary to observe a collective behaviour between an octopus and fish are very hard to replicate in the controlled environment of an aquarium. To be able to capture such behaviour events in video, it was mandatory to work in the natural environment, using scuba equipment and with a moving camera setup.

Most tracking systems have been designed to work in a sterile environment where

[1] https://youtu.be/088a8c5JJdE

there is a plain and bright background and the animal to be tracked is dark and of elliptical shape [14]. In this natural scenario, however, many of the features that are commonly used to track objects through computer vision such as colour and shape are highly variable.

By providing a semi-automated tool that speeds up this process, and automatically tracks octopuses in a three-dimensional environment, we allow for the faster and automatic reconstruction of trajectories and analysis of behaviours. The use of tools such as the one described in this document will help researchers studying marine animals behaviour in a natural context, without invasive techniques and where no tracking technology was ever applied.

1.2 Objectives

The main goal of this project is to use computer vision to track an octopus (*Octopus cyanea*) in its natural environment. The project where this work is contained aims to explore the open problem of developing a semi-automated tracking system that improves behaviour analysis work to be more effective and timely. This is embodied in a system that automatically reconstructs trajectories using a tracking system for both octopus and fish, which in turn can then be used as input for an analysis on collective dynamics, cooperative behaviour. We envision a fully functional system whose input is a three-dimensional underwater video and whose output is an ethogram representing the cooperative behaviour between octopus and fish.

This dissertation focuses on the initial part of the complete analysis, a successful usage pipeline and parametrisation of a neural network that automatically tracks an octopus in a variety of situations in several minutes of video, requiring only the manual labelling of a small part of its frames.

It is the goal of this work to identify existing tools, select the most appropriate ones, and parametrize and adapt one to be able to track the movements of an octopus, from recordings of stereo underwater camera videos.

As a result we will provide a tool (or a configuration setting for an existing tool) that will help researchers in the area of underwater behaviour, in particular, of *O. cyanea* behaviour. As a by-product of this work, we hope to provide a robust base for the future development of the animal behaviour analysis systems, in particular for octopus species.

1.3 Contributions

By providing our results to the community (raw and labelled data, tests, and results) we hope to contribute with material beneficial to communities both in the computer science to improve on video and image processing and analysis algorithms, and to the biology community with a pipeline of software tools that contains both a method and a successful starter configuration.

In summary, our contributions consist on:

- A benchmark for tracking tools, in the form of raw and labelled videos, that can be used in other related studies on tracking in unstructured environments.
- A pipeline of video and image processing and analysis tools consisting on the steps and tools used during the process. The configuration of all the tool parameters used and their refinement to analyse underwater videos obtained in similar conditions to our videos.
- Tracking information that can be used in 3D reconstruction algorithms in the subsequent task of identifying behaviours based on the movement of a single animals or groups of animals.

1.4 Structure of the Dissertation

This thesis is divided into six chapters. In this chapter (the first one) we motivate the work and describe the objectives and the contributions of the project.

Chapter 2 describes the State of the Art on tracking systems for animals.

In chapter 3 we present the main problems and challenges to be overcome or solved in order to reach the objectives laid out in section 1.2.

In chapter 4 we present the approach to the identified problem, we also present the analysis performed on the parameters and respective justification for its use. We also present the pipeline, the validation of the whole process that we use and the complementary tools that were used in the course of this project.

The chapter 5 contains an analysis and discussion of the obtained results. We present the different tests that were performed in order to choose the best parameters to the video analysis. This chapter also presents the results of the tracking system.

In chapter 6 we present the main conclusions of this work, as well as the future work that is enabled by this project.

Chapter 2

State of the Art

Video tracking systems have several usages apart from tracking animals for research reasons. Common usages go from surveillance systems based on cameras, to video games based on motion capture, to systems at the core of self-driving cars. The use of video tracking systems goes back to 1980 with simple Kanade–Lucas–Tomasi feature trackers (KLT). KLT trackers [4] simply apply a feature extraction algorithm to images and then follow those features around on every frame of a video. When compared with current tracking technology, KLT trackets are very simple. Recent approaches are usually based in DNN, which represent a more powerful and expressive response to the tracking challenge.

In the recent years, numerous benchmarks have been developed with the aim of helping visual tracking research, but it is yet very challenging to compare and evaluate tracking systems because each system is very specific to a certain type of videos. Some approaches may be well suited for objects that change colour, but cannot cope with objects that change shape, and vice-versa. Tracking systems that focus on objects that do not change form like systems that track cars will not be able to successfully track an object that changes form constantly like the octopus. Having this in mind, we have analysed tracking systems that are specifically aimed to animal video tracking. However, many of the existing tracking systems cannot be used in the area of video tracking animals in the wild, due to the requirements imposed in the surrounding environment. Our setting, composed by underwater reef images, is particularly challenging. It is composed by an unstructured background (sand, rocks or reef wall), the octopus as the subject of interest, and fish that circle the octopus and interfere with the detection process.

In order to decide which base system would be more helpful to extract octopuses' trajectories in their natural habitat from field videos, we analysed some existing open-source tracking solutions. We next present all the systems that were tested and the one that we actually used in this project.

We start by describing two representative examples of tracking systems that use colour transformations and feature extraction to convert coloured images to black and white images and detecting thresholds for certain levels. These trackers then follow the chosen black or white areas that correspond to the selected feature in a video. Open Vision Control [21] and Trackor [23] are the chosen examples for this kind of tracking algorithms.

Other tracking systems use Neural Networks (NN) and Deep Neural Networks (DNN) at their core to distinguish the animals from their background. In this category we distinguish IdTracker [14] and DeepLabCut(DLC) [13]. All solutions presented below are open-source and are developed in python, thus allowing us to modify and adapt their source code to our advantage. In our case, we needed to change some initial and default configurations that were hardcoded in the source program.

2.1 Open Vision Control

Open Vision Control was initially developed to be an auxiliary tool for diverse university projects that have the need for a tracking solution [21]. The project was developed to be a general tracking software with various features, instead of a specific tracking program. So it would be possible to adapt it and allow it to be used in several controls and automation.

However, the problem is that for each project it was needed a different control code, so this platform allows the user to write his code and use the platform as a kind of sensor that can be integrated with the users' python code. In this way, every project can use the same platform changing only the control code to solve the specific problem. The software itself is open-source, which enables the user to modify and improve the source code. The system was designed to allow to follow movements, objects, colours, to measure distances and to recognize people. It has a useful tool that can follow objects of choice, the fast and slowest objects in the video, the biggest and the smallest objects or even a specific colour of choice.

To make the software more robust, it uses Gaussian filters as size-based objects filters and region of interest filters. We give some examples below in order to illustrate this program's capabilities.

- Detect movements on a room;
- Servo motor control for automated turret type application;
- Follow an object in a video;
- Control the mouse cursor through hand movement.

The feature that we found more interesting in Open Vision Control was the capability to follow a designated object as showed[1] in Figure 2.1.

[1] https://www.youtube.com/watch?v=ECa41UD-WFs

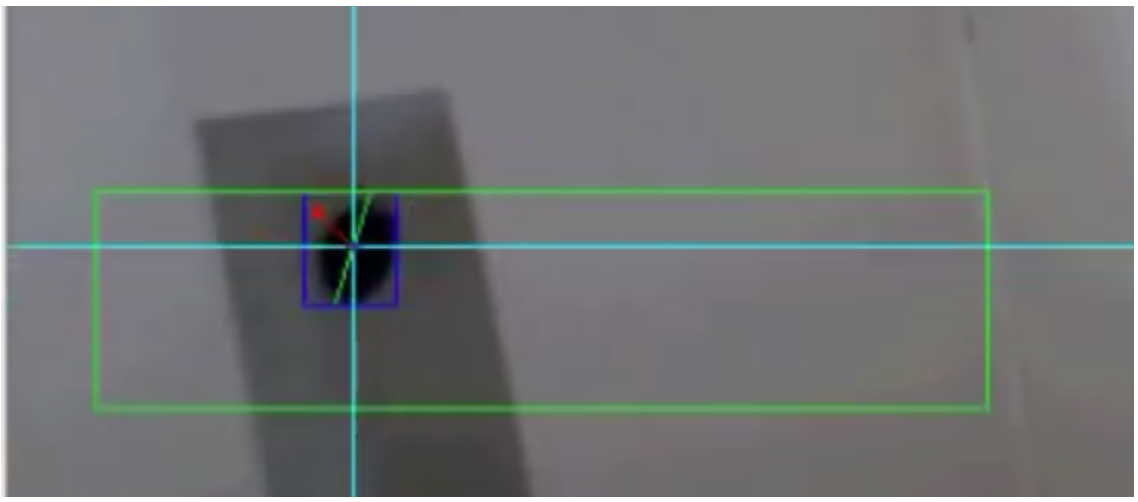

Figure 2.1: Example of Open Vision Control following an object in video.

On the other hand, the disadvantage that we found in this program was that it is relatively old since the last update was in 2011 and the last comment about the program in 2012, which means it is not supported by today's technologies. If other solution hadn't appeared and we decide to improve this program, it would be necessary to know if there is a modern version of the same tool, or we would have to adapt the program to today's technologies.

2.2 Tracktor

The Tracktor software is an object tracking system based on OpenCV code [23]. The software as presented can be used to perform single object tracking under noisy environments as showed in Figure 5.1 or multi-object tracking under uniform environments. It is able to keep individual identities while performing multi-objects tracking. The software does not have a graphical user interface, and all its functions are accessible through a command line interface. It has four key parameters whose values can be adjusted change according to the video that is being analysed. These parameters are:

- Block size → Determines the size of the neighbourhood;
- Offset → Determines where to set the threshold relative to the mean of the neighbourhood;
- Minimum area → Is the minimum area possible that the subject can occupy in any frame of the video;
- Maximum area → Is the maximum area possible that the subject can occupy in any frame of the video.

An advantage of this software is that the code is already prepared to take into account the noise of the environment. Also, although it is not able to track the octopus with all

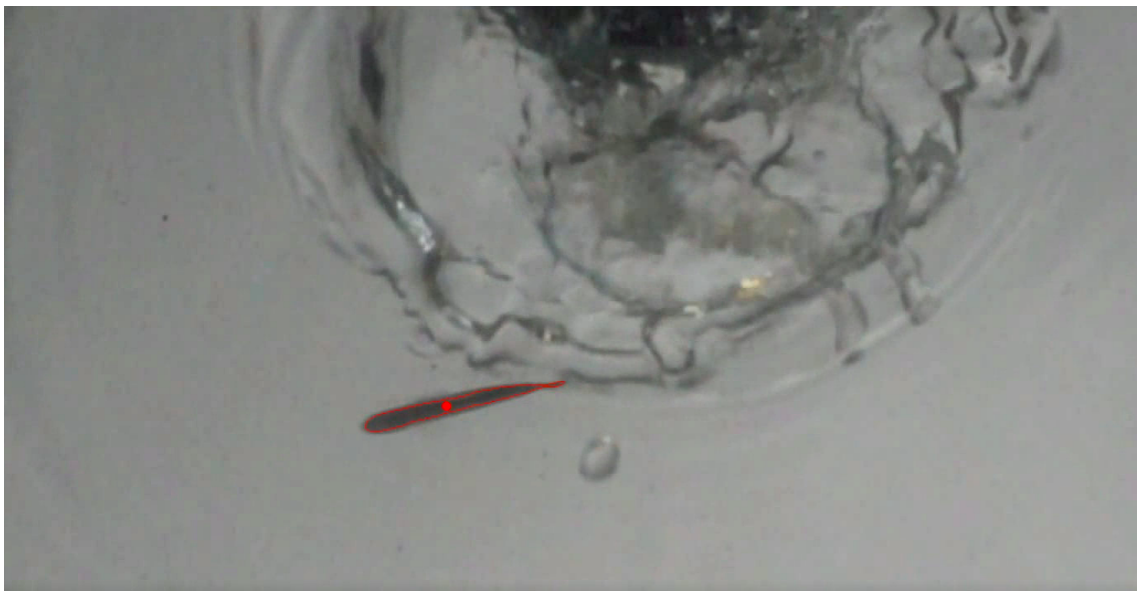

Figure 2.2: Example of tracktor following a fish in a noisy environment video.

the changes of colour and shape itself, with the right parameters, the program can follow the octopus for some moments.

Its disadvantages are that the problem of finding the right parameter values is not solved by an systematic and logical method but by successive trial and error manual processes.

2.3 IdTracker

This program [14] is used to follow schools of up to one hundred individuals. It has the particularity of identifying individuals through visual characteristics instead of just keeping track of each one individually. An example of the software is presented in Figure 2.3 where 72 *drosophila* are individually marked and tracked over a period of time[2].

This difference makes this program much more precise over big periods of time, when compared to the ones that only keep track of regions of interest. Most of the times the program can perform at an accuracy of over 99.9%. It is an advantage when you are trying to track many individuals that cross and overlap each other (CO). To achieve this, the program uses two different DNNs.

The first DNN is used to scan the video frame by frame and to determine if the "black dots" are individuals or CO, it uses the elliptical form of the individuals to distinguish between separated individuals and CO. The other DNN is used after the first one to identify the visual features of each individual, and it uses the training set to attribute an identification to all the objects. Then, the trained DNN is applied to track all the individuals identified by the first DNN.

[2] https://www.youtube.com/watch?v=_M9x14jBzVQ

Figure 2.3: Example of applying IdTracker on *drosophila* video.

2.4 DeepLabCut

DeepLabCut (DLC) [13] is an open-source toolbox that allows the user to extract poses of an animal without the use of physical markers on the individuals.

DLC was developed based on an algorithm called DeeperCut [7], one of the best algorithms for human pose estimation benchmark in our days. Particularly DLC uses DNN and Residual Neural Networks (ResNets) with both 50 and 101 layers, as well as deconvolutional layers as developed in DeeperCut. Some functions of DeeperCut were not considered to turn the process faster. Other functions were implemented so that the reliability was not affected.

To track humans there are better pose-estimation algorithms than DLC, such as DeeperCut [7] or ArtTrack [8]. But DLC stands out from the other pose-estimation of humans when we add features that are apart from the pre-train body parts. For this reason, DLC is the best and first option to pose-estimation of non-human animals. Recently another deep learning package called LEAP was described. LEAP also has good results but require a huge training data set compared to DLC to achieve the same results. Using the same training conditions LEAP relative error is documented to have 4 times bigger than DLC error [13]. In terms of inference speed, LEAP may be faster due to the difference of layers of the network compared with DLC. Apart from this, both can achieve real-time processing. The advantages of DLC compared with all the other programs presented above are the following:

1. The existing documentation guides the user through the whole process step-by-step;

2. It has been tested to a range of animals and diverse background

3. It has a minimum manual cost to achieve human-level accuracy;
4. There is no need for physical markers on the object;
5. It can be easily adapted to analyse behaviours across species;
6. It is open-source and free.

One of the major improvements of DLC compared with other non-human animal tracking programs (i.e IdTracker), is that DLC can handle an entire variety of distinct backgrounds. This allows the researcher to focus on their scientific questions, rather than constraints imposed by the tracking system. Another advantage of DLC is not having the demand for images of fixed frame size, the training process rescales the images to increase the performance of the algorithm. DLC also don't have any special camera requirement. Any commercial or scientific camera, colour or grey-scale, can capture suitable images to feed the algorithm.

DLC also does not need a supercomputer to get results. Any GPU with 8GB memory like NVIDIA GeForce 1080 is enough.

The software is robust so that it do not require any special camera or any *apriori* camera definitions require. If the features to track are visible to manual labelling, DLC should be able to label the body parts as well.

The installation of DLC is also very easy since they provide an Anaconda environment[3] with all the dependencies needed already instaled. After the installation of the packages and the environment, we can follow a step-by-step procedure, which was finaly presented in a pipeline described in chapter 4.

We need to configure several parameters that will influence the training step. The parameters that we need to configure are presented in Table 2.1. The effects of these parameters are explained in detail in subsection 4.3.4.

Table 2.1: DLC parameters

display_iters	*save_iters*	*global_scale*
scale_jitter_lo	*scale_jitter_up*	*pos_dist_thresh*
mirror	*cropping*	*cropratio*

[3]https://github.com/AlexEMG/DeepLabCut/blob/master/conda-environments/README.md

Chapter 3

Detailed Problem Description

3.1 Problem Statement

The class Cephalopoda has been targeted by scientific research for more than one hundred years [5], both in controlled environments and in its natural habitat. This class presents several interesting features, from their complex behaviour to their camouflage techniques. The focus of this work is the species *O. cyanea* and our problem statement can be summarised in the following problem question:

Is it possible to improve research methods involved in analysing the behaviour of O. cyanea, *by tracking the behaviour of an octopus in its natural environment, extracting accurate information about its position, in a three-dimensional video in a semi-automated way, with low labelling effort?*

The purpose of this chapter is to highlight all the challenges that are involved in answering the research question above. The outcome of this work is a system that aims at making the data analysis phase of experimentation an easier, faster, and more objective and precise process. From a technical point of view, we build on available open-source software tools, and provide a software pipeline, comprising tool combination and configuration data and respective data flow. By using these tools one is able to follow an, so far, 'untraceable animal'. This also constitutes a first step to track animals that naturally evolved to trick with camouflage the natural visual tracking processes, such as octopuses.

3.2 Challenges

The implementation of this project is itself a big challenge due to the novelty of the domain of application. In this section we divide it into smaller challenges that can be addressed separately, and whose (sub)solutions compose the final result.

The depicted challenges comprise data acquisition, capturing underwater images suitable for treatment. There are several factors that must be handled in this regard. The unstructured nature of the environment, the limited visibility of underwater images, the fact that we have a moving camera, and the octopus' behaviour.

3.2.1 Natural environment

Working in a natural environment is intricately difficult. The videos produced in the natural environment are not sterile and predictable like the videos produced in laboratory aquariums. Sterile videos have little noise, and have, in general, a white and well lit background with dark objects moving and no unpredictable obstacles.

On the other hand, non-sterile videos have high levels of noise and unpredictable obstacles. These is the kind of video we propose to analyse. For illustration, Figure 3.1 shows a frame of a sterile video[1] (left) and a frame of non-sterile video (right) of one of our samples. When examined, an octopus can be seen in the middle but the resemblance of the textures between rock and animal are evident.

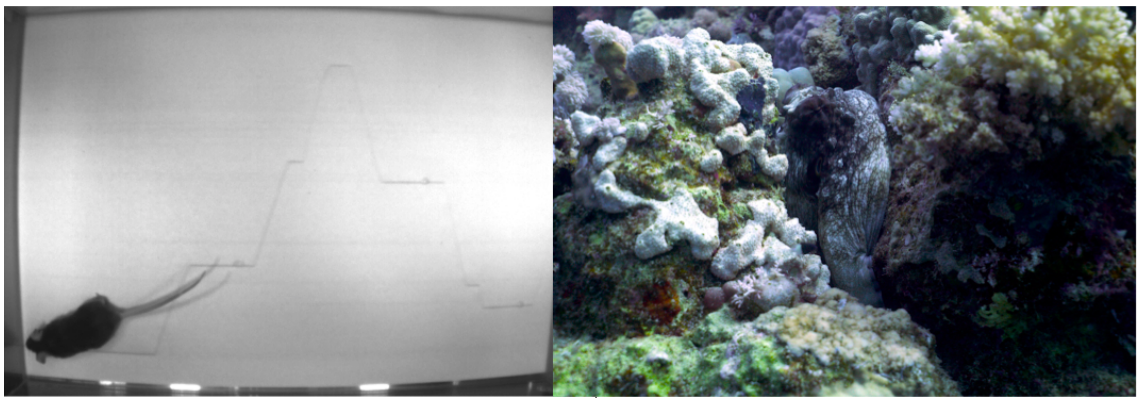

Figure 3.1: Sterile Video Frame (left) vs. Non-sterile Video Frame (right).

Our input videos are filmed in the natural environment so, we deal with a very significant amount of noise, Figure 3.1 (right). Noise can be caused by corals in the background, the sandy seabed, and objects/obstacles in the front or behind the octopus, such as fish moving and rocks. Our samples are therefore classified as non-sterile.

Figure 3.2 shows a high noise situation, where the octopus is barely visible, and for a moment there is the occlusion of the octopus by a fish passing between the camera and the octopus. Our system needs to be able to distinguish the octopus from the rest of the environment and needs to be capable of spotting the octopus after an occlusion.

3.2.2 Underwater images

Underwater images is very limited when compared to their surface equivalents. Since water absorbs different wavelengths of light depending on depth [6], the same object will

[1] Image from DLC examples, URL: https://raw.githubusercontent.com/AlexEMG/DeepLabCut/master/examples/openfield-Pranav-2018-10-30/labeled-data/m4s1/img0014.png

Figure 3.2: Octopus occlusion due to a fish.

have different apparent colours at different depths. This makes it even more difficult to track by an automatic system. The colour red is the first colour to be absorbed, followed by orange, yellow, in order according to the light spectrum. The depths where primary colours disappear are shown in the following list:

1. RED → 4.5 meters;
2. ORANGE → 7.5 meters;
3. YELLOW → 10 to 14 meters;
4. GREEN → 21 to 23 meters.

Additionally, we must take into account that the light reflected from a certain object has to travel the water mass correspondent to the object's depth, plus the water mass that separates the camera from the object, which acts as a further limitation. In essence, the closer to the object the observer is, the more colourful the object will appear.

This also indicates that the deeper the video, the harder it will be for the NN to use colour information in the training method.

3.2.3 Moving camera

Octopuses are wild animals and don't stay in a specific spot for a long time. Taking this into account, the hunt episodes are not stationary, so we adopted a search-and-follow procedure to capture these cooperative hunting events. In contrast to the stationary record, we will not be able to control on-line the movement of the camera. The diver who is filming is aware that he should have the octopus close and in the centre of the screen to help the tracking software. Nevertheless, this is not always possible because of octopus behaviours that can not be predicted beforehand. Some times the octopus may

move faster that the diver and can get too far to get a good image, or the octopus can hide behind a rock or coral faster than the diver can go around the same rock. This is one of the reasons that some videos are easier than others to analyse.

3.2.4 Octopus

Finally, the octopus itself is a challenge. Since it is being filmed in the wild we cannot control or predict his movements. Second, and most importantly, it is an animal that displays huge degrees of body shape flexibility, being able to pass through 17 millimetres holes [24] in extreme cases of whole body modification. Moreover, it can also change his texture and body colour due to chromatophores cells [12] that cover all the body. In other words, it can change all the visual features that common tracking systems use. In Figure 3.3 we have two octopus, the one on the left is presenting light spotted colours, and spiny texture, the one on the right is presenting a dark brown colour without white spots and smooth texture. This is just a example of the variation that the octopus skin can have, other combinations of these features are also possible.

In addition to this, the octopus cannot be physically marked. The skin is one of the sensory mechanisms of the octopus and it is covered with nerve cells, making impossible to mark the octopus without compromising his natural behaviour.

Figure 3.3: Octopus with different colours, shape and texture.

3.3 Requirements

The main requirement of this work is to have a tracking solution that works underwater and that can track the species *O. cyanea* in his natural environment. This solution aims to help research on decision-making, communication, and cognition during inter-specific collaborative hunting between the octopus and multiple fish species.

With this in mind, the solution for the problem described does not need to work on-line and has no real-time concern, it is only evaluated by its accuracy.

Other requirements are based on the video input. The software needs to be able to work with the videos from the stereo cameras so that after there is the possibility of overlapping the tracks with the background. Some times, the octopus is far, occluded, or change shape and colour, the tracking system needs to satisfy these requirements.

Chapter 4

Materials and Methods

In this chapter we present the methodology followed to address the problem enunciated in chapter 3. The first steps taken in this process targeted the exploring of existing tracking systems, focusing on the ones that were able to analyse videos from non-sterile environments, and the ones that use NN or DNN for tracking. These particular systems were studied with bigger detail because they analyse videos with similar features to what can be found in an underwater video.

4.1 Video Acquisition and Video Analysis

The first benchmark videos used in this work were collected images on prior fieldwork performed by Eduardo Sampaio and Simon Gingins at Eliat, Israel. All other images were produced in fieldwork during the course of this dissertation work together with Eduardo Sampaio. In this section, we describe the location, hardware of this fieldwork period.

4.1.1 Fieldwork Location

The core fieldwork for this dissertation was performed in the *Roots Red Sea* house reef, El Qoseir, Egypt, where we collected more hours of video of the collective behaviour between the octopus and the fish. This location was chosen due to is close connection to the *Open Ocean Science Centre*[1], that provided us with a proper workspace during the expedition. Additionally, this location provides diverse diving spots which allowed us to film in different backgrounds, leading to a heterogeneous collection of videos[2].

[1] https://openoceanproject.org

[2] In terms of complexity we divided the collected videos in two categories: the simple ones, that have the background all made of the same material and with the same colours (e.g. Rocks or Sand background) and the complex ones, that are the ones that the background is made of different materials and colours (e.g. Coral background).

4.1.2 Hardware tools for video acquisition

To be able to film three-dimensional images, two full-frame Sony Alpha 7SII with Zeiss Batis f/2 25mm lenses were mounted on an aluminium structure as a stereo camera setup (hereafter "Stereocamera Rig"), see Figure 4.1.

This approach allowed us to film at a mean distance of approximately 5 meters while collecting high-resolution videos. Both cameras were synchronized by sound. To film the close up videos we used one full-frame Sony Alpha 7SII with Zeiss Batis f/4 24/70mm (hereafter "Zoom Camera"). This camera was also synchronized by sound with the other two cameras and allowed us to follow the octopus movements with much more detail than the others.

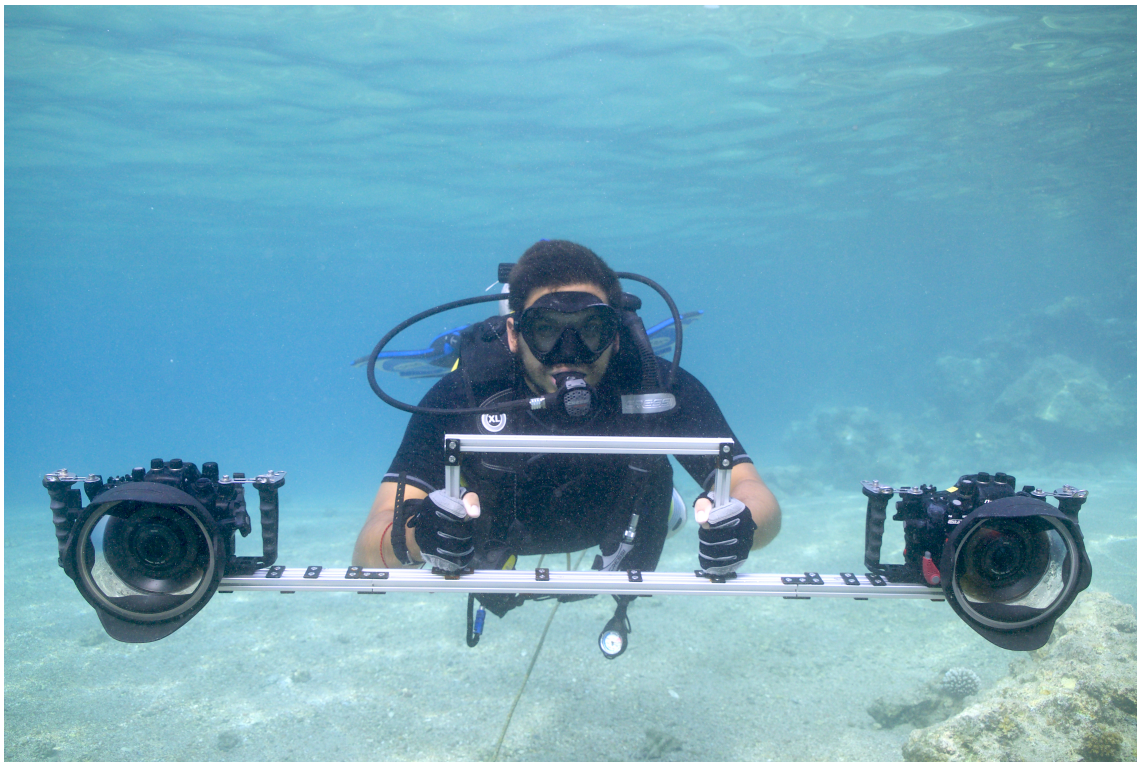

Figure 4.1: Stereocamera Rig used for dual acquisition of interspecies events in Eilat, Israel and in El Qoseir, Egypt.

4.1.3 Hardware tools for video analysis

We start by using Google Colab but this solution had several drawbacks. It only allows a low-quality images, global_scale needs to be 0.4 or low to work. Google Colab would unpredictably stop the train several times and the time needed to run the 1030000 iterations was of 10 days non-stop, because of the unpredictable Colab stops we usually needed 13 days to run the whole train.

We then used an HP Z8 G4 Workstation with 32 GB of RAM, installed with Microsoft Windows 10 Pro for Workstations. The Workstation processor is an Intel(R) Xeon(R) Gold

5120 CPU @ 2.20GHz, 2195 Mhz with 14 Core(s) and 28 Logical Processor(s). We also integrate with NVIDIA TITAN V. It has the power of 12 GB HBM2 memory and 640 Tensor Cores, delivering 110 teraflops of performance. Plus, it features Volta-optimized NVIDIA CUDA for maximum results.

This configuration allows to run the train function non-stop in approximately 5 days.

4.2 Tracking Tool Testing

The preliminary literature research indicated that the *Tracktor* [23], an adaptive threshold based tool, was probably the most suitable tool for our setting. Tracktor allows the analysis of videos based on a core configuration of four parameters. The first two, named *min_area* and *max_area*, define the area that an object to be tracked can have in the image. The remaining two are named *block size* and *offset*. The *block size* parameter determines the neighbourhood size and the *offset* determines where to set the threshold relative to the neighbourhood mean. These two parameters combined control the adaptive threshold for the grey scale obtained.

Finding the right combination of values for these parameters is a trial and error process. We started by using a video from the Zoom Camera to test Tracktor and by trying out several ranges of parameter values. We measured the area occupied by the octopus in some frames of the video and then we defined the *min_area* and *max_area* according to that. The *block size* and the *offset* were based all in trial and error.

These experiments showed to no satisfactory results, shown in chapter 5, which led to a renewed effort of looking for an alternative approach. The elected alternative is DLC which is described in the subsequent sections.

4.3 DeepLabCut based Pipeline

Our alternative, and definitive approach, is based on DLC, a DNN based tool. To perform an evaluation of this tool, we followed the prescribed processing pipeline and protocol made available by the authors in the tool website [13]. We used a selected part of a video, filmed with the Zoom Camera, which was also used to test Tracktor, as referred above. The concrete steps taken were the ones referred in section 2.4. Our work then proceeded to build a pipeline (tools and parameters) suitable to our domain. A deeper understanding of the labelling strategies and of the parameters of DLC are next presented in subsection 4.3.5 and in subsection 4.3.4.

When we understood which parameters we needed to change and their meaningful range of values, we started to systematically applying the analysis to the videos from the Stereocamera Rig to train the NN instead of the videos from the Zoom Camera, primarily used for tool assessment. These images are used in the end of the process again to obtain finer-grained results (subsection 4.3.8).

In order to preprocess videos and images we used to auxiliary tools, Ffmpeg and Fiji.

4.3.1 Ffmpeg

'FFmpeg is the leading multimedia framework, able to decode, encode, transcode, mux, demux, stream, filter and play pretty much anything that humans and machines have created.' [3] We used FFmpeg to extract the frames that we need from the videos with the necessary rate of images per second.

After a quick installation, we can choose how many frames we want per second and the quality of the image obtained.

We use this program to extract the frames for the data set and to extract the frames for the validation.

For the data set we use the following command:

```
1 ffmpeg -i input.mp4 -qscale:v 1 -r f/s img%03d.jpg
```

Being `f/s` the number of frames extracted per second and `img%03d.jpg` the formating string for producing the output file names.

For the validation, we wrote a python script that uses FFmpeg to randomly extract the number of frames we desire from both cameras. This script is available in Annex I

4.3.2 Fiji

We use Fiji [20] to label the body parts. This program has very good usability and allows us to select which body part we want to label and label it in all frames. This feature allows us to label faster since we do not need to move the zoom window a lot. Labelling an octopus that is far away in the frames is not an easy task so this kind of help reduce significantly the time spent labelling each body part. In the end, we export the labels and format into the DLC format and we can continue the training process.

4.3.3 Pipeline

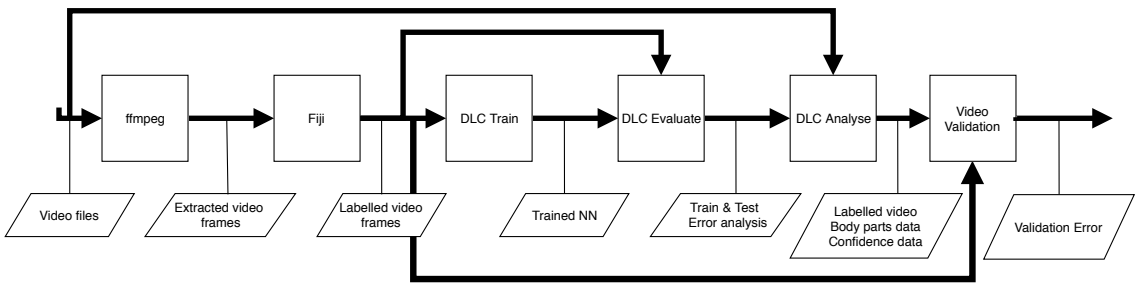

Figure 4.2: Adopted DLC pipeline

The adopted DLC pipeline follows the steps in Figure 4.2 that starts by using tools like *ffmpeg* and *Fiji* to extract the video frames from the input videos and to manually label selected frames as the source data set. Then, different DLC functions are used to train a neural network, analyse the results, and join analysis and original video in a single output. The final output of our pipeline is a labelled video where all tracked body

parts are graphically and analytically identified. The numerical data given, as output, identifies the featured body parts (their location in the picture) and the confidence level of the identification.

The first step of the whole process is to select the video files to be analysed and extract video frames from it using ffmpeg. Extraction of video frames may follow different strategies, which are described in subsection 4.3.5. The extraction process is detailed in subsection 4.3.1.

The second step is the manual labelling of frames. For this task, we choose to use tool Fiji, described in subsection 4.3.2. The output of this stage is a set of manually labelled video frames. This step also includes the visual verification of the labelling process using the appropriate DLC function. These tool choices were made due to the usability of both programs. In the case of Fiji, it allows us to label each animal body part in all the frames, individually, and DLC default functionality require us to label all the body parts in each frame at once.

The third step of the pipeline is to create the training dataset to feed to the training functionality of DLC. This step divides the dataset into train and test dataset with the percentage defined in the global configuration file. The output of this stage are the iteration snapshots of the trained NN.

The fourth step is the evaluation of the NN. It uses the train and test data sets to measure the train and test error, and the train and test error with $p-cutoff$. We run this step for all the snapshots saved in the previous step. In the end, we will analyse the values for train and test error and draw the curve of the error during the train process.

The fifth step is the video analysis and creation of a labelled video. In this step, we choose the snapshot with better results with train and test data sets and analyse the whole video with the NN trained with that snapshot weights. The result of this step is a file with the X, Y coordinates and the likelihood value of each body part in each frame. With this file, we create a labelled video that allows the visual analysis of the success of the NN.

The sixth and final step is the validation of the labelling accuracy of the NN by comparing unused labelled images and the DLC results (body part data). The output is the validation error, that certifies the final output; a completely labelled video and the body part data produced in the fifth step.

An important part of the pipeline are the configuration files that specify the whole process. The set of parameters and corresponding values are detailed in the subsequent sections below.

4.3.4 DLC Parameters

In this section we discuss the fundamental parameters of DLC configuration. As expected, there are parameters with higher impact on the results than others. The challenge here is to obtain a suitable combination of values for such parameters that better performs in the domain in hand.

4.3.4.1 Neural Network

DLC uses multiple types of neural networks. It uses DNNs, ResNets, and deconvolutional layers as initially developed in DeeperCut [7]. The initial DLC configuration is targeted to identify bodyparts in video frames, the available network configuration lets us choose between alternative ResNet configurations. DLC offers the possibility to choose between pre-weighted ResNets with 50 or 101 layers. It also allows for reusing already trained networks. We need to set the parameters *net_type* and *init_weights* (a file with a weighted network and the given layout). We tested both layouts available and decided to use ResNet 50 in most of our training and analysis work. The results supporting this decision are shown in subsection 5.3.1.

4.3.4.2 Global Scale - GS

The parameter *global_scale* defines a scale factor applied to the images that are fed to the NN. The values assigned to this parameter range between 0 and 1 (the default value for this parameter is 0.8). This basically defines the resolution of the images being analysed, the higher the value the higher the image size, the lower value the faster the NN. This parameter allows us to compromise between resource usage (mainly used memory) and the resolution of the input.

This parameter also allows to address different kinds of images fed to the NN. If, on the one hand, images fed have low-resolution this parameter needs to be higher. On the other hand, when using high-resolution images this parameter may be slightly lower, or needs to be lower because of resource exhaustion.

Since we are using 4K videos[3], we are dealing with images with very high level of detail, therefore, even if we set this parameter to 0.33(3) we still have HD videos[4]. As recommended by the authors of DLC, we try to maximise value of this parameter in order to have the best result possible.

We tested the differences in the results of training the NN with this parameter set to values 0.8, 0.7 and 0.6. Based on the results, shown in subsection 5.3.2, we decided to use the value 0.6 on all our videos.

4.3.4.3 Data Augmentation

DLC uses data augmentation methods by randomly varying the scale of images, and also randomly mirroring and cropping them.

The resolution of the images fed to the NN is defined by the global scale parameter above. However, on each iteration, the actual scale used on the images is changed according to parameters *scale_jitter_lo* and *scale_jitter_up*. The training function scales the images by the factor *global_scale* randomly modified by a factor between the values of the two parameters. The default values for these parameters are 0.5 and 1.25 respectively.

[3] frames with 4096×2160 pixels.
[4] frames with 720×1280 pixels

To configure the mirroring behaviour, we have parameter *mirror*, which is a boolean parameter that allows DLC to invert (or not) the symmetry of images in the vertical axes.

In our case, the focus of the tracking is an octopus, that itself is symmetric around the vertical axis. So, in theory, we should set this parameter to TRUE. However, we need to remember that the camera and the octopus are moving during the video in all directions. So, we already have a high level of symmetry of the octopus included in the training sets. We also identified left-hand-side and right-hand-side features (eyes) in some videos. So, we wanted to avoid interference in those results that may be introduced by this process. So, we set this parameter to FALSE in all our samples.

The boolean parameter *crop* defines the application of a cropping mechanism performed during the training, and the associated and *cropratio* parameter defines the size ratio of cropped images. We can also define the boundaries of the crop with the parameters *leftwidth*, *rightwidth*, *bottomheight*, *topheight*, and the parameter *minsize* will determine the minimum size of the cropped image. The images will be randomly cropped within these boundaries.

In our trials we tested the NN with parameter *cropping* for both TRUE and FALSE values. After this test, we determined that it had a good effect on the results, and for all the analysis therefore we set this parameter to TRUE.

4.3.4.4 Maximum Image Size - MIS

During the data augmentation phase, the size of the images is randomly changed. The *max_input_size* parameter is used to filter down the number of images fed to the NN, The square of the parameter *max_input_size* defines the maximum area that a frame can have. If a frame occupies more area than this parameter, the image will be discarded in this iteration.

The default value for this image is 1000, which is enough to accept HD images of 1280×720[5]. Since we are using 4K images, whose resolution is 4096×2160, the area considered was always bigger than the default parameter value[6]. For our experiments we needed to consider an increased value of 3200[7]. In combination with the global scale set to 0.6 we guarantee that the jitter process stays within the bounds and all images are accepted.

4.3.4.5 Iterations

We use the parameter *display_iters* to check if the training is running at the normal rate, between 2 and 5 iterations per second. At first, this parameter was set to 1, meaning that at every iteration the iteration number of the NN was printed on the command line console. When we were running sequentially all the videos we set this parameter to 1000 since we trust that the training process was occurring without errors.

[5] $1000 \times 1000 = 1000000 > 921 \times 600 = 1280 \times 720$
[6] $1000 * 1000 = 1000000 < 8847360 = 4096 \times 2160$
[7] $3200 \times 3200 = 10240000 > 8847360 = 4096 \times 2160$

We use the parameter *save_iters* to set the period of iterations in which we save the weights of the NN to a separate file, called snapshots. Allowing one to restart training at any saved point.

In some cases, we set this value to 50000, since we are running 1030000 iteration we end up with 21 snapshots of that NN to evaluate. In this way, we can observe and analyse in detail the training progress, for example analyse if there are local minimums or if there are any strange behaviour in the training.

However in the most of our cases, we set this parameter to 200000, with 6 intermediate snapshots to evaluate and choose which is the best snapshot to analyse the videos later.

4.3.4.6 Distance Threshold - DT

The parameter *pos_dist_thresh* defines the radius of the distance threshold, in pixels. This threshold determines if the training samples are considered positive for the detector or discarded.

If this parameter is set too small, too few images would be accepted for training, if the value is set too high, too much error (noise) would pass into the detector phase.

The discussion about how to get to this value is presented in [11] and we choose to accept this value based on that.

4.3.4.7 Likelihood Threshold - p-cutoff

During the training process, for each frame and for each body part, the NN produces 3 values. The X and Y values and the likelihood of the point corresponding to that body part (ranging from 0 to 1).

The likelihood value is then compared to the parameter `p-cutoff`, which defines the likelihood threshold. It helps to distinguish the uncertain body parts from the more likely ones.

In this way, we obtain a NN where the uncertain body parts can be put aside and only considered the more likely ones to construct the labelled video.

The default value for the parameter $p - cutoff$ is 0.1, which means that all frames with a likelihood bellow 0.1 are discarded.

The accepted error in each area is different: some cases, the task may be so difficult than 0.1 of error or higher is accepted and in others, errors bigger than 0.1 may be too high.

In the scientific community, there is no written consensus about what is an acceptable error value. In general an acceptable error rate is 0.05. This means that likelihood needed in our case would be 0.95.

DLC set the default of this value to 0.1 and this lack of consensus and big differences between concrete cases, lead us to compare the results of both likelihood values.

4.3.5 Frame Labelling Strategies

Labelling is an activity that identifies featured points in an image, in our case the feature were octopus and fish body parts. We identified 3 distinguished points on each eye, 3 distinguished points on the siphon[8], 4 distinguished points on the mantel and body, tail and eye points on some fish cooperating with the octopus.

The labelling process starts by selecting a small data set from the huge amount of video frames present in a full video to train the NN. In this section, we present several strategies that we use in our work. Strategies vary in terms of frequency of frames per second, in terms of the number of frames, and on the length of the video analysed. For each strategy, we reduce the number of frames manually labelled per minute of video, therefore reducing the overall time consumed per video, up to the point where results are still satisfactory. The different approaches are systematically presented in the Table 4.1 and the results of this different approaches are presented in chapter 5.

4.3.5.1 Strategy 1

At the beginning of the training, we started with small videos of 2 minutes to understand the performance of the NN. These 2 minutes are a selected part of the original videos that we found more suitable for the NN. We selected these 2 minutes because there were little or no moments where the octopus disappears or gets in low-light places. For these 2 minutes of video, we have extracted and labelled 3 frames per second, every second during the whole video. This leads in terms of the size of the data-set that we start to train the NN with approximately 750 manually labelled frames, 375 from each camera. In this way, we gather a very robust dataset with almost all poses, colourations and textures that the octopus assume during those videos.

4.3.5.2 Strategy 2

After we have good results with strategy 1 we started to train and analyse the original videos. These videos have different lengths from 20 to 30 minutes each. Since they are the original videos, they are not treated or trimmed so there are parts where the octopus is occluded, the light is low or where the angle of the cameras is not the best for analysis. Therefore these videos are harder to analyse than the ones in subsubsection 4.3.5.1. The previous data set was robust and achieved good results. But now we are dealing with videos 10 to 15 times bigger so to continue the same frequency of frames per second manually labelled, the effort would be huge[9]. Hence, we decrease the frequency of the frame extraction to 1 frame every 5 seconds instead of 3 frames each second. With this change, we moved to train the NN with datasets containing between 480 and 720 manually labelled frames each, depending on the duration of the video.

[8] A tube shaped body part of aquatic molluscs that makes water flow through their breathing system.
[9] $30min \times 60sec \times 3frames \times 2cameras = 10800$ manually labelled frames compared to 750 in strategy 1.

4.3.5.3 Strategy 3

To test the limits of the NN, we reduce drastically the number of frames manually labelled and we try to train the NN with just 60 manually labelled frames, 30 from each camera.

3.1 We first test this approach to the subsubsection 4.3.5.1 videos, and use the first 30 labelled frames of each camera, corresponding to the first 10 seconds of the video.

3.2 Then we test this approach to the subsubsection 4.3.5.2 videos, where we use as well the first 30 labelled frames of each camera, but this time corresponding to the first two and a half minutes.

Table 4.1: Summary of the different strategies of frame labelling.

Parameters	S1	S2	S3.1	S3.2
Length of the video(Minutes)	2	30	2	30
Data set size(Frames)	750	720	60	60
Frequency of frame extraction(F/S)	3	1/5	3	1/5
Frames per minute of video	375	24	30	2

In terms of human effort strategy 3.2 is without a doubt the one with more benefit. We will have to evaluate the error of DLC associate with a small dataset compared to the error associated with a big dataset, then we need to decide which is more beneficial and establish a ratio between human effort and accuracy of the NN.

4.3.6 Human labelling error

We also tested for the possibility that human labelling is introducing errors in the process. To measure it we labelled the same feature in a set of frames twice. Then, we measured the Euclidean distance between the featured points. We then compute the average error in the labelling to extrapolate the amount of error that the human labelling process may introduce in the training process.

We did the same test in five different videos, with different body parts, and with a different number of frames per video, with a total of 300 frames tested. The computed values, for the differences in the x and y axis and the average Euclidean distance, are presented in Table 4.2. This establishes a lower bound to what to expect a NN can achieve (3.16 pixels).

Table 4.2: Human Labelling Average Error in pixels.

	pixels
ΔX	2.96
ΔY	1.76
d	3.16

4.3.7 Validation

In order to validate our results we have can take into account several aspects. The first and more obvious is looking at the labelled video and check if the dots placed in the features by DLC correspond to the actual features. This is a good way to analyse the video qualitatively, give us the first impression about how good the results are. Yet we can not compare videos this way.

Another method we can use is looking at the likelihood of the DLC predictions. In each feature of each frame, DLC presents the likelihood of the label. Then we can analyse the average likelihood of each feature and assume that if all features have values over 0.95 the train was a success. If there is one feature with a value lower than that, we can remove that feature from the analysis. We can then compute the average of the likelihood of all features in that video, and extrapolate the average global likelihood of the video (for all features). We will call this measure, the confidence of the NN. Videos with high values, over 0.95 of confidence, are likely well labelled when compared to the ones with lower values, under 0.95 of likelihood.

The third and more reliable method of validation is the manual label validation. We use a python script, Annex I, that selects and extracts 30 random frames from both cameras of the original video. We then label those frames manually and save the X and Y coordinates for each frame. We next select those frames from the DLC labelled video and extract the X and Y coordinates and the likelihood for each one of them. We determine the average Euclidean distance between the DLC labelled frames and the manual labelled frames, to this distance we call DLC error. If the likelihood of DLC label in under 0.95 we consider that the label is not good enough, so we compute a DLC error with confidence higher than 0.95 to remove all the possible wrong labels. This value indicates how many pixels the correct solution is from that feature in the DLC labelled image.

Knowing this, a small value for the euclidean distance indicates that the error is small so the video was well labelled, a larger value indicates the video was badly labelled. We need to keep in mind that the error is measured in pixels and not in percentage and that the frame size is 3840 per 2160 pixels An example of how all of these values are analysed is in Annex II.

4.3.8 Zoomed-in Tracking

After several successful trials with the videos from the Stereocamera Rig, we decided that was worth it to try a neural network to analyse also the Zoom Camera videos and determine how accurate the neural network was. In order to do this test, we labelled three points in the eye, three points on the siphon, three points on the mantel and one point on the beginning of the third right arm. After we testing two videos with these characteristics, we observed that both were successful. The results of these trials are presented in section 5.8. Once this was not the focus of our study, we do not have enough data to analyse and we did not invest a lot of time and resources in these NNs. Nevertheless,

these NNs could be used in future projects to analyse, for example, breathing rates during several tests, body movements or any other specific detail observed in the octopus body.

Chapter 5

Results

In this chapter we present the results associated to the questions raised in the previous chapter (chapter 4). We follow a similar structure and provide results and answers/decisions to those questions.

5.1 Filmed Events

In the fieldwork described in section 4.1, we were able to film 30 different events, which correspond to approximately 38 hours or 2300 minutes of videos and over 1.5TB of data. In this chapter, we present results regarding 8 events representative of the whole collection.

We analyse 6 events filmed with the Stereocamera Rig and 2 events filmed with the Zoom Camera. ConsiderTable 5.1 that contains the information about the events names, corresponding video duration and source camera.

Table 5.1: Events and Corresponding MetaData.

Event Name	Duration (min)	Source Camera
MartimPedras1	2	Stereocamera Rig
MartimCorais	2	Stereocamera Rig
ZeSimao	30	Stereocamera Rig
ZeSousa	25	Stereocamera Rig
ZeManuel	13	Stereocamera Rig
ZeMarco	14	Stereocamera Rig
MartimZoom	1	Zoom Camera
ZeBrunoZoom	29	Zoom Camera

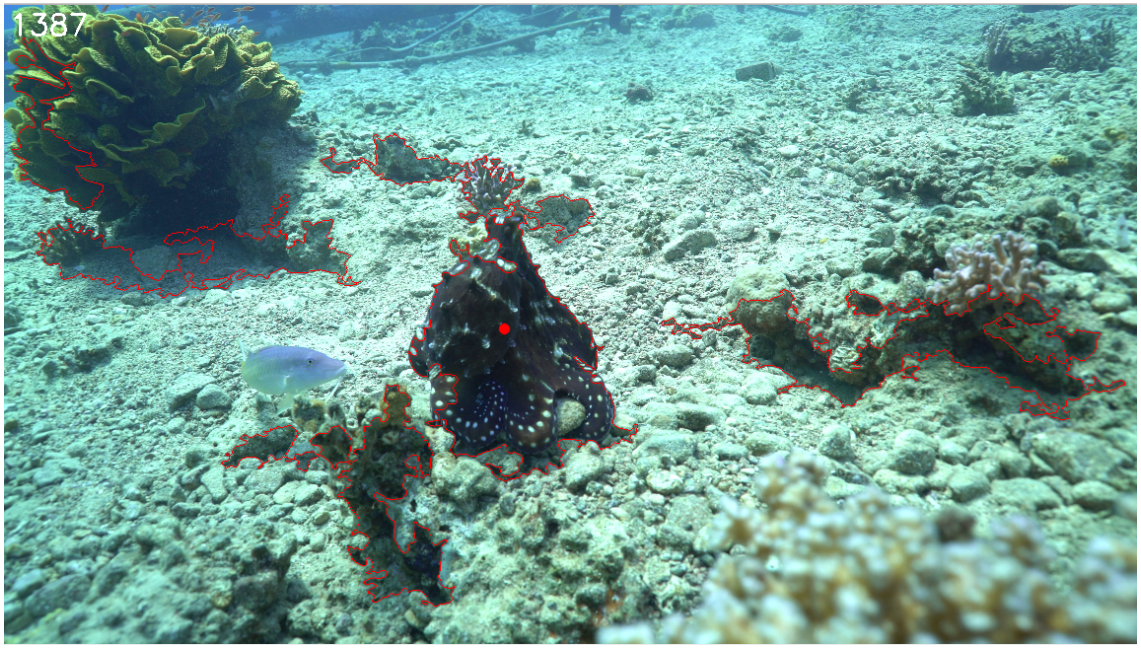

Figure 5.1: ScreenShot of Tracktor running successfully

5.2 Tracktor Results

This section describes the best results obtained on the tests described in section 4.2 on the open-source tool Tracktor, based on OPEN CV. Given the base configuration, we did trials that were able to improve the number of frames tracked by the program without any error, the precision and accuracy of the tracking. The best parameter values we could find are presented in Table 5.2.

Table 5.2: Best values found for Tracktor parameters.

Parameters	Values (px)
Block Size	121
Offset	21
Min Area	6 000
Max Area	1 000 000

This result illustrated in Figure 5.1, is much better than the result obtained with the predefined program values. However it was not totally satisfactory. We can observe in the depicted frame, the centre of mass of the identified region is on the octopus. Yet, it identifies many other regions as having the same features and during the length of video, the centre of mass changes between the octopus and those regions. After doing multiple trials with several videos and different values for the core parameters, we decided that this tool was not appropriate for the task we had in hands. Therefore we started searching for a different approach.

5.3 DeepLabCut Parameter Comparison

In this section we show a thorough comparison of parameter values, showing their impact on the results. To narrow the search space, we performed comparisons by changing just one selected parameter and maintaining all the other parameters configured with the initial values determined as a good starting point and hereafter named standard values and showed in Table 5.3. In this way, we hope to reduce the variability of the comparisons and present more objective results.

Table 5.3: Standard Values for DLC Parameter Comparison.

Parameters	NN	GS	MIS	DT	p-cutoff	MIRROR	CROP
Values	50	0.6	3200	17	0.1	False	True

5.3.1 Neural Network

To be able to compare ResNet 50 with ResNet 101 we use the video *ZeBrunoZoom* which resulted in the train and test errors compared in Figure 5.2. Both alternatives had a good results in the train error, but in the test error the ResNet50 presented better and steadier results. So, we use ResNet 50 for the rest of the trials and tests.

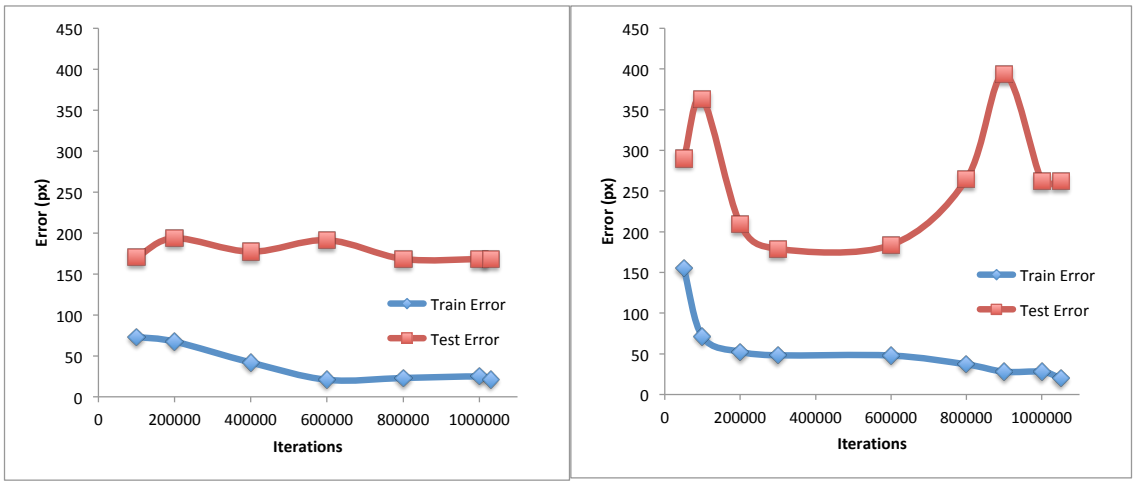

Figure 5.2: Comparison of Train and Test error of ResNet50 (left) and ResNet101 (right).

5.3.2 Global Scale

We started our tests with default value 0.8 for the *global_scale* (GS). However, values 0.8 and 0.7 easily exhaust all the resources of our Workstation and stop the training process. To avoid this from happening we use the value 0.6 and adopted it as the standard value for this parameter. In this way, we still have good quality images and we can run the train without setbacks.

However we were able to test the combination of $GS = 0.8$ with $MIS = 1000$ and compare it to the combination of $GS = 0.6$ and $MIS = 3200$. The results are presented in Figure 5.3

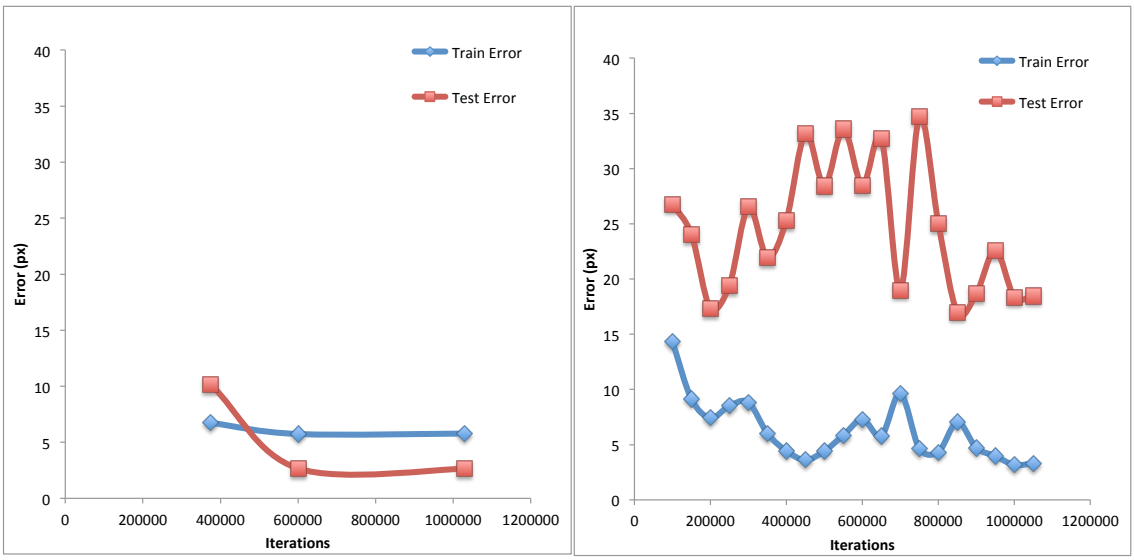

Figure 5.3: Comparison of Train and Test error of $GS = 0.8$ and $MIS = 1000$ (left) with $GS = 0.6$ and $MIS = 3200$ (right).

The video used for this test was *MartimZoom*, and it was filmed very close to the octopus (i.e, the octopus occupied most of the image). With the parameter, $GS = 0.8$, only on the 0.40 of iterations where *cropping* = True the images would be small enough to be accepted by $MIS = 1000$, so very few images would be accepted for training. Since the image selection, sizing and cropping process is random we do not know the exact number of frames that were analysed. We correlate the good (unexpected) result of the NN to the simplicity of the video and the fact that only a few frames were accepted both in train and test iterations. If the video was more complex, further away from the octopus or had a bigger duration we predict that the result would not be this good.

We want a reliable solution that is not affected this easily, so we use $GS = 0.6$, $MIS = 3200$ to run the rest of the videos.

5.3.3 Maximum Image Size

We start with this parameter set to 1000 pixels with the results, for the video *ZeBruno-Zoom*, shown in the left side of Figure 5.4. The result obtained is very bad and shows that the NN is not able to improve the performance since both train and test error remain similar in all snapshots. We determined that the bad results are caused by discarding too many frames due to the size threshold.

Then we run one other NN with this parameter set to 3200 pixels and the results were very much improved, as presented on the right hand side of Figure 5.4. As expected both train and test error decrease with the increase of the number of iterations.

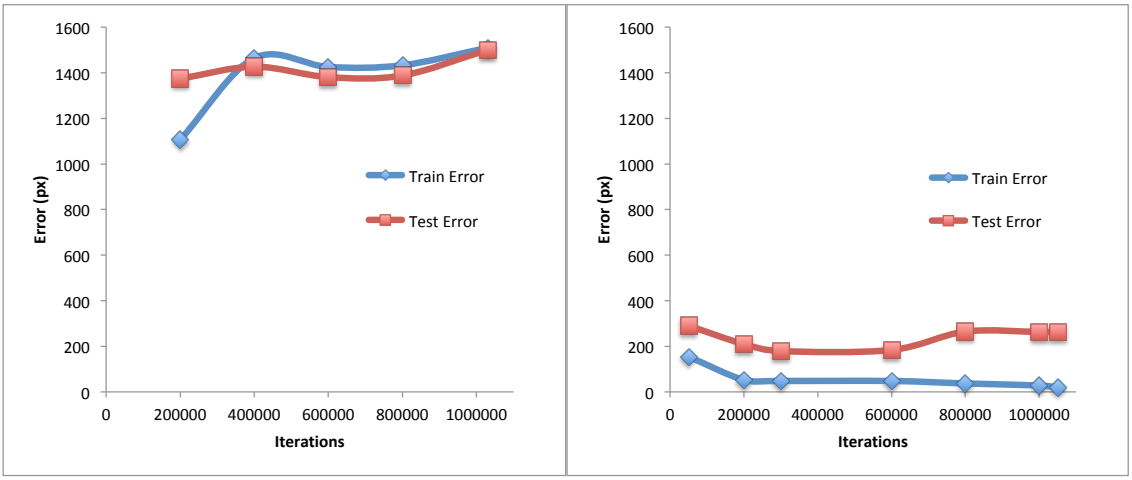

Figure 5.4: Comparison of Train and Test error of MIS = 1000 and MIS = 3200

5.3.4 Cropped Images

In theory, this parameter in our case would be beneficial to be set to TRUE as explained in subsubsection 4.3.4.3. Nevertheless we tested this theory and present here the results. We trained two different NN, one with this parameter set to TRUE and one with the parameter set to FALSE.

The results are presented in Figure 5.5 respectively. In terms of train error, the values are similar but with the parameter *crop* set to TRUE, the error decays faster and continues to decline with the increment of the iterations. This desired behaviour is not observed in the NN with the parameter *crop* set to FALSE where we observe the error to stabilize with the increment of the iterations.

In terms of test error, the differences in the results are much more visible. The NN with the parameter *crop* defined as TRUE starts with high values of error and with the increment of the iterations the error starts to decay to lower values, a result that indicates successful training. The NN with the parameter *crop* defined as FALSE starts with a medium value of error but starts to increase with the increment of the iterations.

This is a clear warning for us that the NN is not improving its performance there for the training is unsuccessful. We want to have the best performance of the NN so we set the parameter crop in all the other NN to TRUE.

5.3.5 P-Cutoff

When we evaluate each snapshot we obtain the train and test error and the error with $p\text{–}cutoff$. The default value for the parameter $p\text{–}cutoff$ is 0.1. We use this value to evaluate and analyse the results of the training in all the videos, nevertheless we evaluate one video with the value 0.95.

In Figure 5.6 we can observe that the values of both train and test decrease when the $p\text{–}cutoff = 0.1$ is applied. We can also observe that the test error with $p\text{–}cutoff$ continues

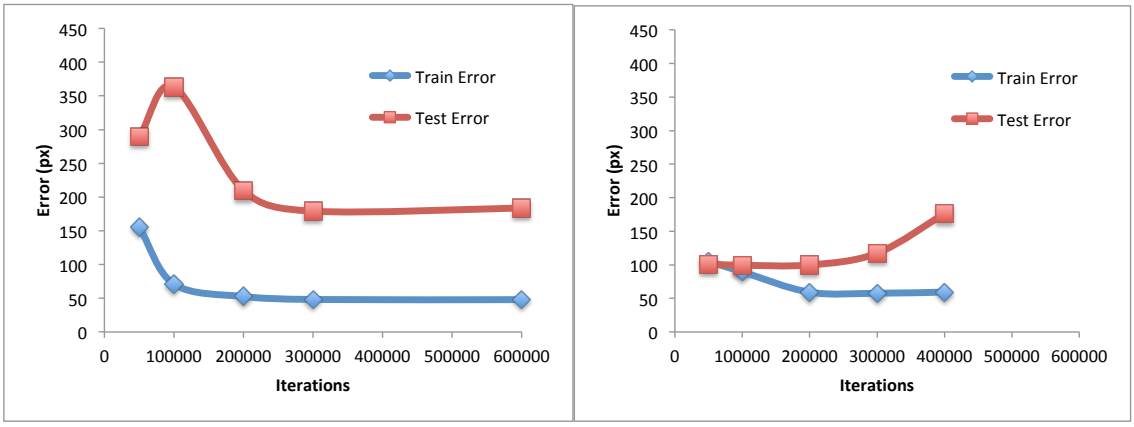

Figure 5.5: Comparison of Train and Test error of *CROP* = TRUE (left) and *CROP* = FALSE (right)

to decrease after iteration 600000, even though the test error without *p–cutoff* starts to increase. This is a clear indication that the NN with the increment of iterations is getting confident of the correctness of the results and start to exclude the body parts that it is not able to track in each frame. At the same time, it indicates that the NN is getting over fitted to the train data set. The increase of the test error after the 600000 iterations indicate that the test data set have some frames where the NN is not able to perfectly label some body parts. With the over fit of the train data set this prediction gets worse with each iteration.

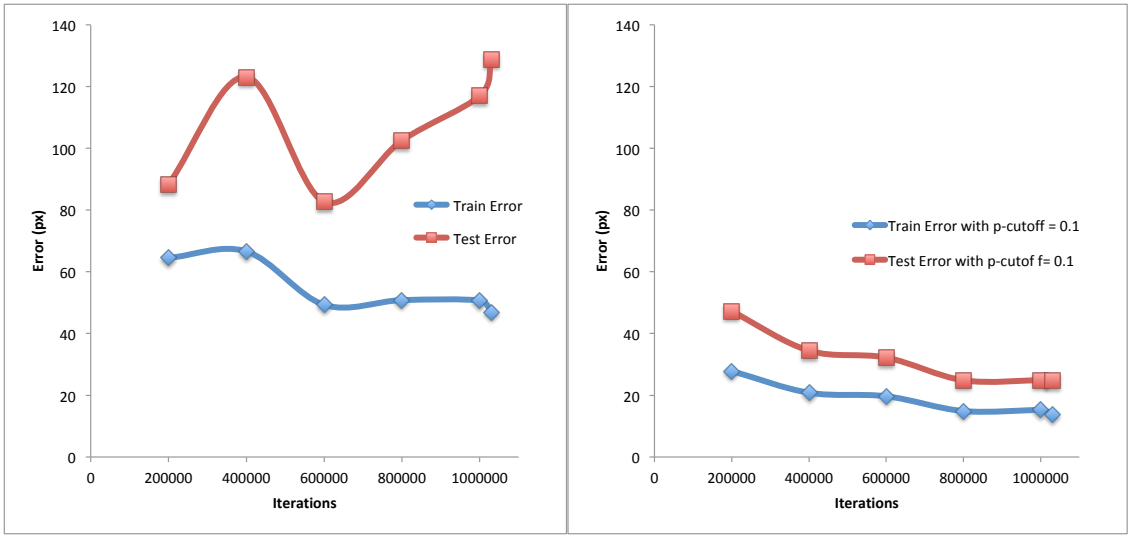

Figure 5.6: Comparison of Train and Test error and Train and Test error with *p–cutoff* = 0.1

We also run one NN with this parameter set to 0.95, meaning that only the body parts with a value higher than 0.95 of likelihood in each frame are considered likely ones to construct the labelled video.

The results of this test are presented in Figure 5.7. In the first iterations, the contrast between results could not be clearer, the error with *p–cutoff* = 0.1 decreases with the

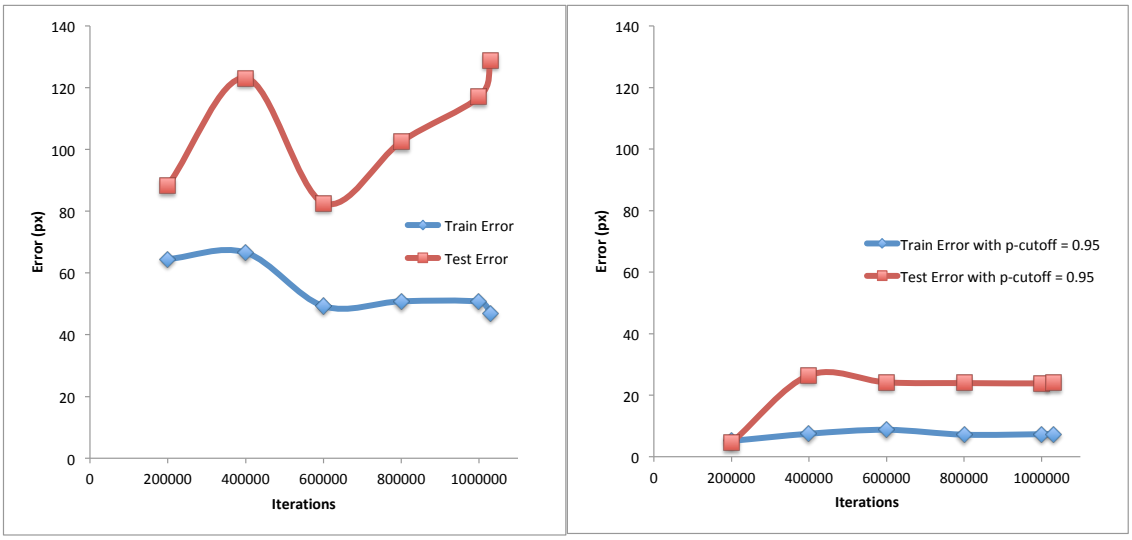

Figure 5.7: Comparison of Train and Test error and Train and Test error with $p-cutoff$ = 0.95

increment of the iterations unlike the error with $p-cutoff$ = 0.95 that increase. After iteration 800000 the behaviour of test error with both $p-cutoff$ is the same. The train error is still a bit different between values of $p-cutoff$, but the $p-cutoff$ = 0.95 had better results.

We want to analyse our videos and have the best output as possible so we could simply choose to set this parameter to 0.95 but, since the likelihood or confidence is an output of the NN the same way as the X and Y of the body part, the confidence will improve with the increment of the iterations. If we use a high value of $p-cutoff$ our train and test error with $p-cutoff$ of the first iterations would always be inaccurate and low, which could lead to mistakes.

The number of images that had been well labelled but was discarded with the $p-cutoff$ of 0.95 will be higher than the number well labelled frames discarded with the $p-cutoff$ of 0.1. To be able to fully understand the true behaviour of our NNs, regardless of with more error, we decide to keep this value in 0.1.

Notice that the $p-cutoff$ used in validation error is different from the one used in the train and test error in the strategies results sections. The $p-cutoff$ used in validation error is 0.95. The $p-cutoff$ used in train and test error is 0.1.

5.4 Results of Strategy 1

We tested two videos with Strategy 1, *MartimPedras1* and *MartimCorais*, both from the Stereocamera Rig. Both videos were analysed with this strategy were a success. We present the results of each situation individually on the following subsections.

5.4.1 *MartimPedras1*

We chose this video from the Stereocamera Rig because it was an easy video. We can notice that almost all the background is made of small identical rocks. We know from the dive that this video was filmed at low depth, which is also proven by the colour of the video. The octopus, during almost all the video, has a reddish vivid colour, which should help the NN to distinguish it from the grey rocks. In Figure 5.8 we can observe the difficulty, the features of this video and the differences between manually labelled (+) and DLC labelled (•) in *MartimPedras1* frame. The detail of the image, zoomed in, can be observed on the right.

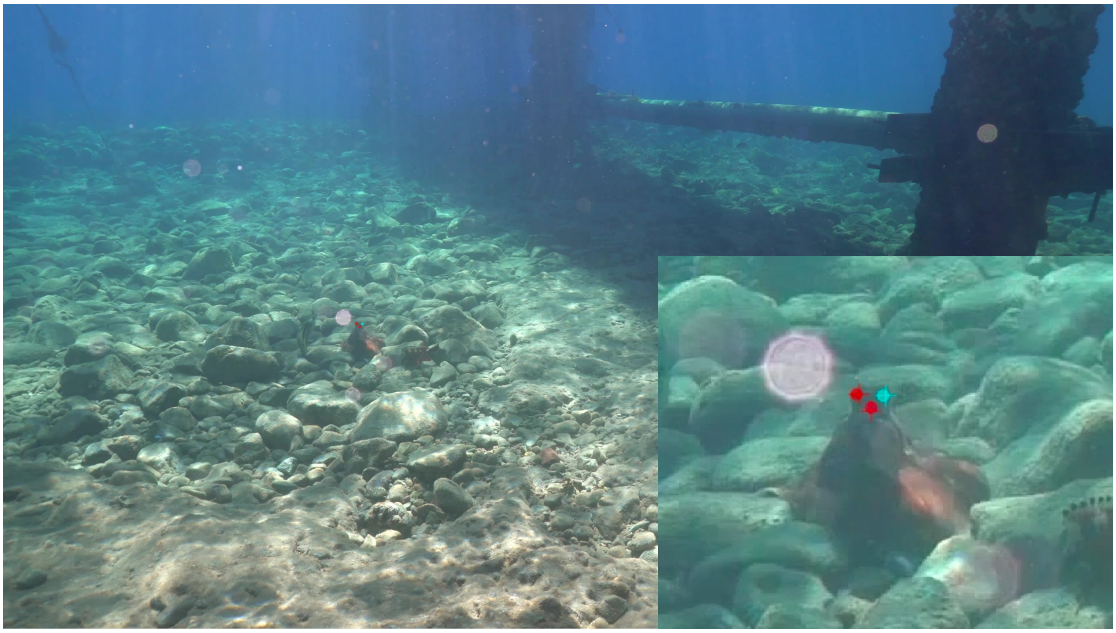

Figure 5.8: Comparison between manually labelled (+) and DLC labelled (•) in *MartimPedras1* frame. The detail is zoomed in on the right.

In Figure 5.9 we can observe the result of the training process. In this video, we save snapshots every 50000 iterations with the intention of understanding better the behaviour of the NN. We can observe that both train error and train error with *p–cutoff* behave the same way, pointing that the NN can easily identify the body parts in the train data set with high confidence. We infer that this behaviour is linked to the similarity of the frames of the video.

In terms of test error and test error with *p–cutoff*, the behaviour is more distinct. The test error in the first 350000 iterations decreases but then in the 400000 iterations the error increases dramatically. Between the 450000 and 850000 iterations, the error declines slowly until a new jump, fall and jump.

In the test error with *p–cutoff*, the error is dropping slowly with the increment of iterations. This indicates that the NN is tuning the weights of the net in order to improve the label of the body parts. The error will never drop significantly more because it had

reached the values of the human labelling error.

These sudden jumps and falls in the test error along with the slow drop of the test error with *p–cutoff* indicates over fit of the NN. To try to analyse the video without an over fitted NN we choose the iteration 750000 to do it.

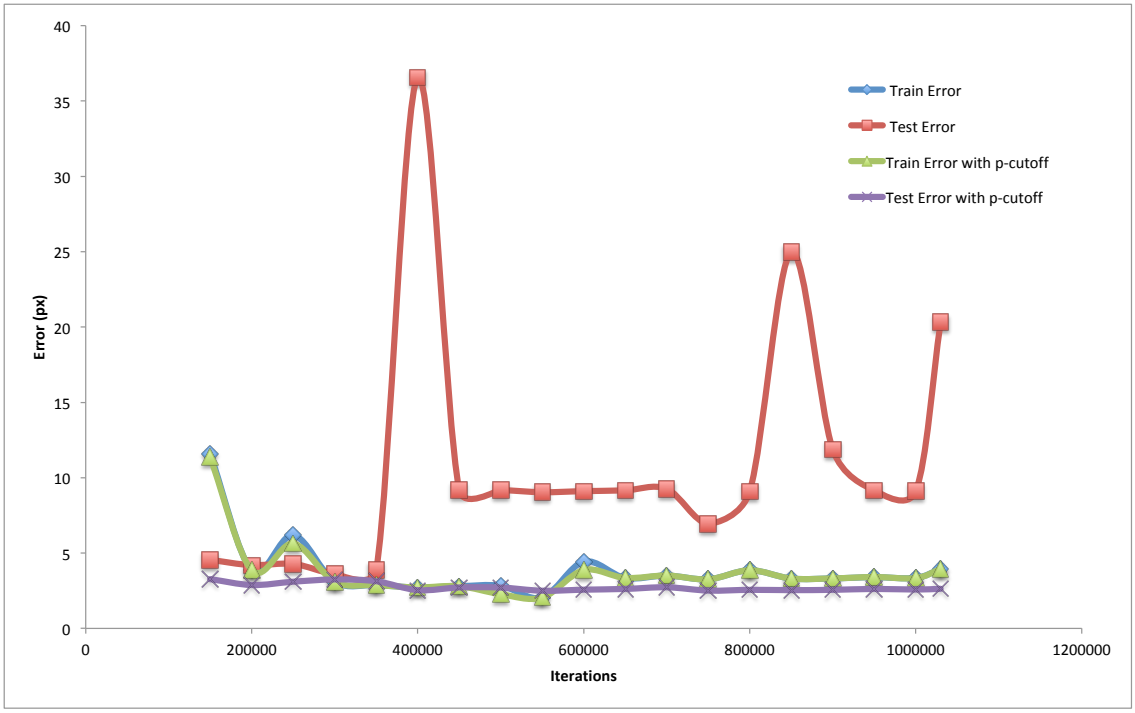

Figure 5.9: Train and Test error, Train and Test error with *p–cutoff* of *MartimPedras1*.

Then we run the DLC functions, analyse and create labelled video, and we obtained the labelled video along with the positions of each body part in each frame. After the validation process we get the results present in Table 5.4

Table 5.4: Iteration 750000 of *MartimPedras1* - Train, Test and Validation Error Summary

	Train	Test	Validation
Error	3.26	6.96	4.18
Error with *p–cutoff*	3.26	2.51	4.09

We can verify that the train error and train error with *p–cutoff* are the lower of the three categories as expected. We can compare the test error with the validation error, both values are low but the validation error is slightly better. In the end we compare the error with *p–cutoff* of both test and validation, we observe the opposite of what happened in the test and validation error. We attribute this difference to the low number of images analysed in both test and validation process. One frame with a slightly off prediction may lead to a big difference in the average error. To make this value more robust we would need to manually label even more frames, what would be a big human effort.

The average confidence during the whole video is approximately 0.99.

We visualize the labelled video and taking into account the values presented in Table 5.4 we consider that this video was labelled with human level precision.

5.4.2 MartimCorais

We chose this video from the Stereocamera Rig because it was harder than *MartimPedras1* and it has multiple fish around the octopus. We can notice that the background is diverse in colour and shape, (i.e. with increased complexity). As MartimPedras this video was filmed at low depth, we can clearly distinguish the red colour in some corals and all the other colours look lucid. The octopus during the video changes of colour and shape several times. In this video, we ask the NN to identify the 3 octopus body parts, the two eyes and the mantel but also 3 different fish in the eye and the tail. This video could be harder for the NN to label than the previous one.

In Figure 5.10 we can observe the difficulty, the features of this video and the differences between manually labelled (+) and DLC labelled (•) in *MartimCorais* frame.

Figure 5.10: Comparison between manually labelled (+) and DLC labelled (•) in *MartimCorais* frame.

In Figure 5.11 we can observe the result of the training process. In this video, we save snapshots every 200000 iterations with the intention of selecting the best one for the video analysis. We can observe that both train error and train error with *p–cutoff* behave the same way but, the train error with *p–cutoff* having lower values. This result indicates that the NN is identifying well most of the body parts but do not have high confidence in all.

In terms of test error and test error with *p–cutoff*, the behaviour is more distinct. The test error increases until reaches the 400000 iterations and then it starts to drop until

the 600000 iterations. The iteration where the NN start to over fit and the error start to increase again.

In the test error with *p–cutoff*, the error is dropping slowly with the elapse of iterations. The NN is getting more confident about the predictions of the body parts.

We assume that the best snapshot to analyse the video based on the analysis of the error is the iteration 600000, where the test error is lower

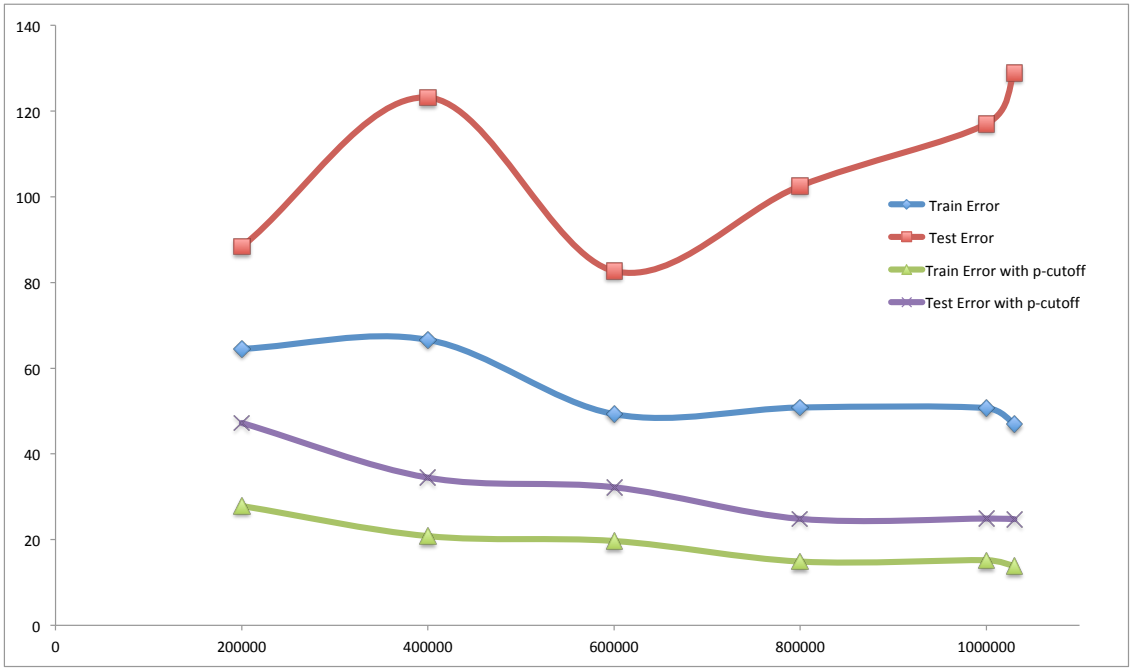

Figure 5.11: Train and Test error, Train and Test error with *p–cutoff* of *MartimCorais*.

Then we run the DLC functions, analyse and create labelled video, and we obtained the labelled video along with the positions of each body part in each frame. After the validation process we get the results present in Table 5.5

Table 5.5: Iteration 1030000 of *MartimCorais* - Train, Test and Validation Error Summary

	Train	Test	Validation
Error	49,29	82,68	128.18
Error with *p–cutoff*	19.68	32.21	4.24

We can verify that in all categories the error with *p–cutoff* is lower than the error, this is expectable since the NN is more likely to get the perditions right when having high confidence.

In the train and test the difference between errors is significant and understandable but is in the validation that the difference of errors is bigger and more interesting. This big difference is due to the variability of this video in particular. Having a high number of frames and iterations to train the NN gain a lot of confidence in its predictions. If the NN shows high values of confidence the prediction of the body part is highly probable to

be correct. So, even thought we have a big validation error if we do not count with low confidence frames so the validation error is minimal.

The average confidence during the whole video is approximately 0.56.

We also visualize the labelled video and consider that this video was also labelled with a human level precision. Both videos analysed with strategy 1 were considered well labelled, in spite of one being of low complexity and one being of high complexity. As a conclusion, we can say that Strategy 1 is a time-consuming strategy but reliable.

5.5 Results of Strategy 2

We test four videos with Strategy 1, *ZeSimao*, *ZeSousa*, *ZeManuel*, *ZeMarco*, all from the Stereocamera Rig. These videos have different background complexities and were filmed from different distances to the octopus. Some were a success others, due to complexity or distance, no so much.

We present the results of each individually in this following section.

5.5.1 *ZeSimao*

We chose this video from the Stereocamera Rig to test in first place strategy 2 because it was an easy video and similar to *MartimPedras*. By visually analysing the video we can notice that almost all the background is made of small rocks and have some small coral patches. We know from the dive that this video was filmed at low depth, so the colours of the video are very similar to the original ones. This video has the difficulty of having a lot of light reflexes and some visual effect due to temperature gradients, thermoclines. The octopus during almost all the video has a reddish vivid colour, which should help the NN to distinguish from the green water and yellowish rock sandy bottom.

In Figure 5.12 we can observe the difficulty, the features of this video and the differences between manually labelled (+) and DLC labelled (•) in *ZeSimao* frame.

In Figure 5.13 we can observe the result of the training process. In this video, we save snapshots every 200000 iterations to select the best snapshot to analyse the video with the NN. We can observe that both train error and train error with *p–cutoff* behave the same way, showing that the NN easily identify the body parts in the train dataset with high confidence. We infer that this NN behaviour is linked to the similarity in terms of colour and shape of the octopus during the whole video.

In terms of test error and test error with *p–cutoff*, the behaviour is notably more distinct. The test error in the first 800000 iterations decreases but then the error increases again without stabilize.

In the test error with *p–cutoff*, the error is steady around 5 pixels of error, with the increment of iterations. This indicates that the NN can label the likely body parts with high confidence but is having problems labelling body parts where the confidence is low. The error with *p–cutoff* will never descend significantly more because it had reached the values of the human labelling error.

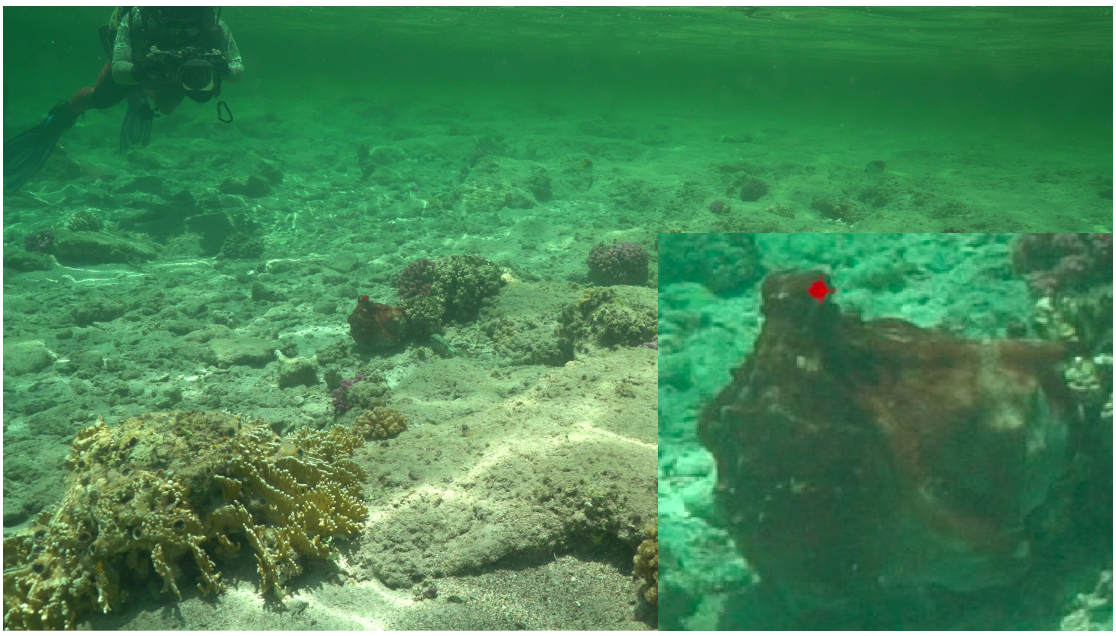

Figure 5.12: Comparison between manually labelled (+) and DLC labelled (•) in *ZeSimao* frame.

The jump after iteration 800000 in the test error indicates over fit of the NN. To try to analyse the video without an over fitted NN we choose the iteration 800000 to do it.

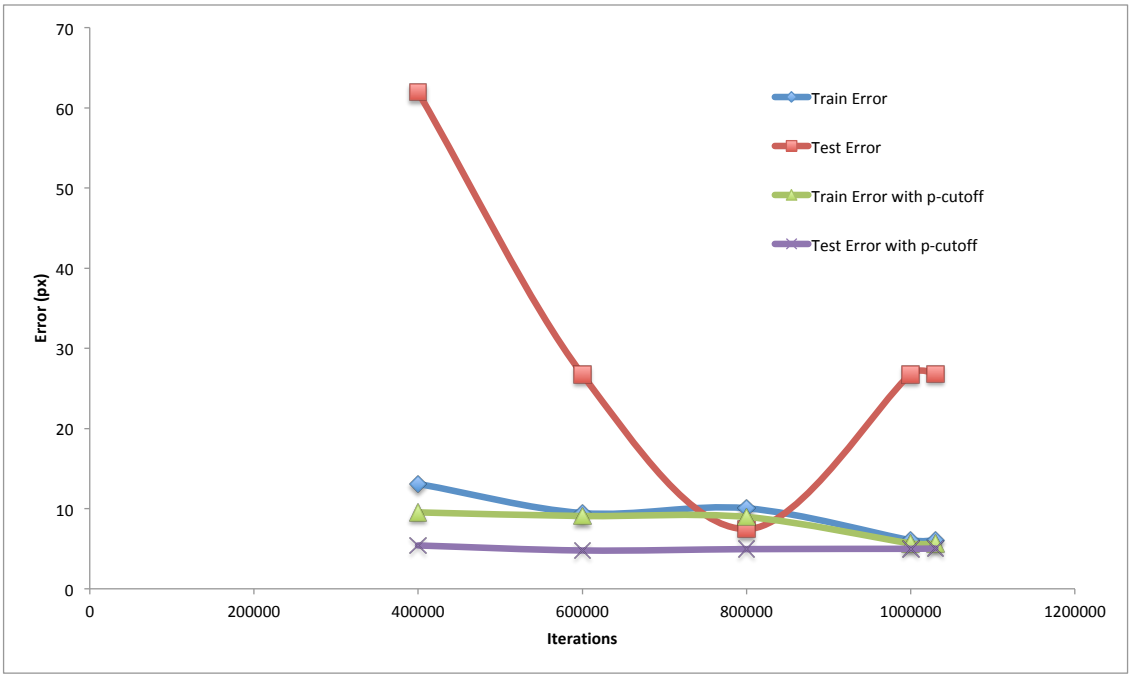

Figure 5.13: Train and Test error, Train and Test error with *p–cutoff* of *ZeSimao* with Stratagy 2.

Then we run the DLC functions, analyse and create labelled video, and we obtained the labelled video along with the positions of the body part in each frame. After the

validation process, we get the results present in Table 5.6.

Table 5.6: Iteration 800000 of ZeSimao - Train, Test and Validation Error Summary.

	Train	Test	Validation
Error	10.05	7.45	8.88
Error with *p–cutoff*	8.96	4.95	8.24

We can verify that the error with *p–cutoff* in all categories is the lower that the error as expectable. All categories have a low error with insignificant differences between them. The only value that stands is the test error with *p–cutoff*, probably due to a low number of test images with confidence over 0.95.

The average confidence during the hole video is approximately 0.96. We attribute this difference to the low number of images analysed in both test and validation process.

We visualize the labelled video and taking into account the values presented in Table 5.6 we consider that this video was labelled with human level precision.

5.5.2 *ZeSousa*

We chose this video from the Stereocamera Rig because it was similar to *MartimCorais* but it has 25 minutes. Visually analysing the video we can notice that the background is diverse in colour and shape, it is a complex background. We can also notice that there are some shadows areas and that the octopus besides changing colour and shape it also hides in those areas several times during the length of the video. This video was filmed at medium depth, approximately 10 meters, we can observe that the colour in some corals is already looking different from their original colour.

In this video, we ask the NN to identify one octopus eye. This video is harder for the NN to label, it has a more complex background than *ZeSimao*, is longer than *MartimCorais* and it only has one labelling point, the right eye.

In Figure 5.14 we can observe the difficulty, the features of this video and the differences between manually labelled (+) and DLC labelled (•) in *ZeSousa* frame.

In Figure 5.15 we can observe the result of the training process. In this video, we save snapshots every 200000 iterations to select the best snapshot for the video analysis. We can observe that train error starts to drop after the 400000 iterations and until the 600000 iteration where it stabilizes near the 130 pixels of error. Train error with *p–cutoff* behaves differently, being stable at 30 pixels with the increment of iterations. This result indicates that the NN is improving the labelling for non-likely body parts but keeping good results in the likely ones.

In terms of test error and test error with *p–cutoff*, the behaviour is similar in both. The error increases until reaches the 400000 iterations and then it starts to drop until the 600000 iterations for test error with *p–cutoff* and until the 800000 iterations for test error.

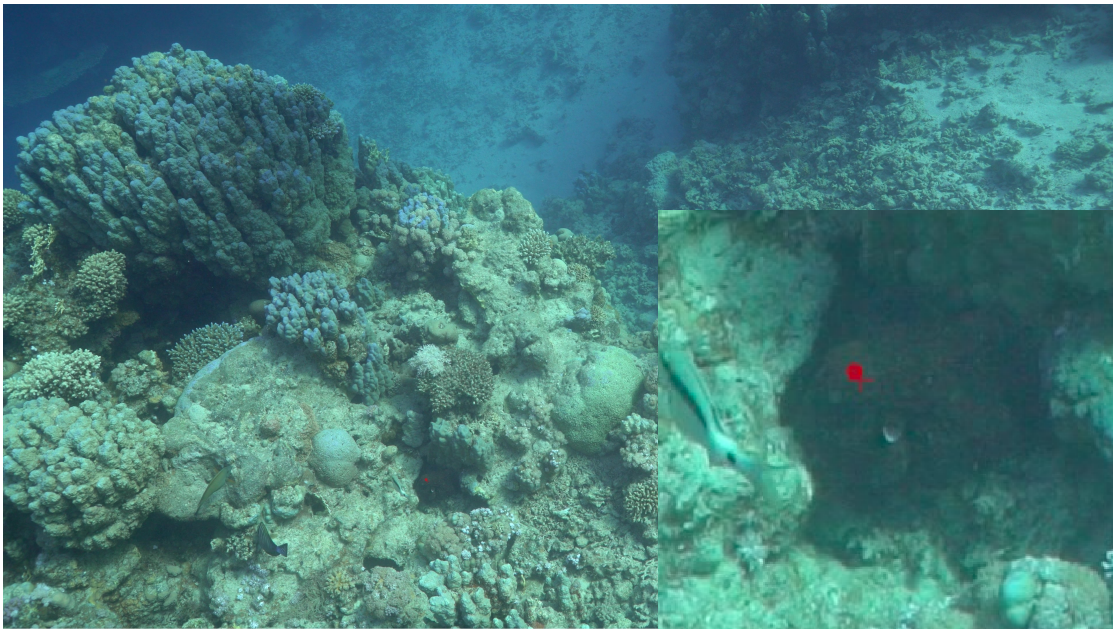

Figure 5.14: Comparison between manually labelled (+) and DLC labelled (•) in *ZeSousa* frame.

This behaviour is starting to be common in the test results, the error drops until a certain iteration, the test error with *p–cutoff* stabilizes and the test error increases again, overfitting the NN. The iteration where the NN starts to overfit and the error starts to increase again in this video is iteration 800000, so we assume that the best snapshot to analyse the video is that iteration, where the test error is lower.

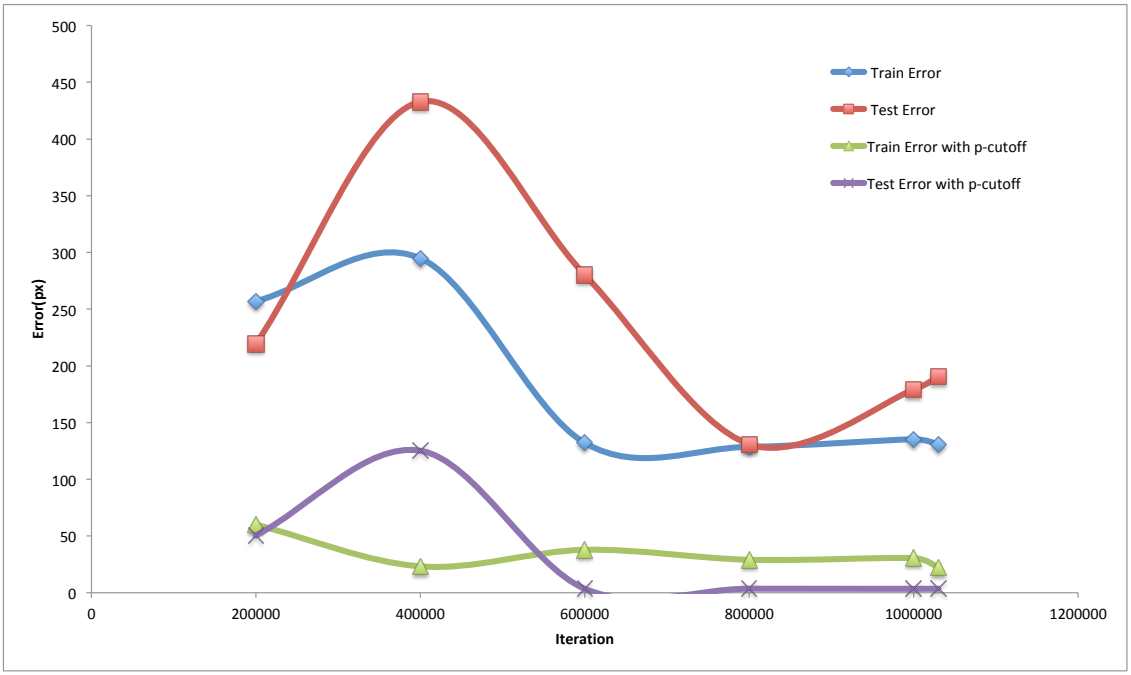

Figure 5.15: Train and Test error, Train and Test error with *p–cutoff* of *ZeSousa*.

Then we run the DLC functions, analyse and create labelled video, and we obtained the labelled video along with the positions of the body part in each frame. After the validation process we get the results present in Table 5.7.

Table 5.7: Iteration 800000 of *ZeSousa* - Train, Test and Validation Error Summary.

	Train	Test	Validation
Error	128.65	130.89	333.47
Error with *p–cutoff*	29.03	3.56	147.54

We can verify that in all categories the error with *p–cutoff* is lower than the error, this is expectable since the NN is more likely to get the perditions right when having high confidence.

In the train and test, the difference between errors is significant and understandable, the error value in both is very low for this type of complex video. Since we are dealing with a very complex video we were expecting that the validation error to be big, but the validation error with *p–cutoff* is much higher than expected.

The average confidence during the whole video is approximately 0.62.

We also visualize the labelled video and consider that this video was not labelled with a human level precision. During the video, some parts were in fact labelled with human level precision but others are not acceptable. We can compare this video to *MartimCorais* because the important features of the videos are similar. *MartimCorais* NN had very good results unlike *ZeSousa* so we can blame the Labelling strategy of the bad results. With the increase of complexity, we need to increase the number of labelled frames per video.

5.5.3 *ZeManuel*

We chose this video from the Stereocamera Rig because it was different from the other two, have both sandy areas and coral patches, and have a smaller and more difficult octopus to track. Visually analysing the video we can notice that the background is mainly sand but that the octopus spends the majority of the video moving in the coral patches, it is a complex background. We can also notice that beneath the coral there are some shadows areas and that the octopus besides changing colour and shape it also hides in those areas several times during the video. This video was filmed at medium depth, approximately 12 meters, we can observe that the colour in some corals is already looking different from their original colour and that the octopus is very dark compared to the previous analysed videos.

In this video, we ask the NN to identify both octopus eyes. This video is hard for the NN to label, it has a complex background, is filmed from a big distance and the octopus is small making the octopus occupying fewer frames than usual and the octopus spend a lot of time inside the coral patches and appearing in the opposite side of it.

In Figure 5.8 we can observe the difficulty, the features of this video and the differences between manually labelled (+) and DLC labelled (•) in *ZeManuel* frame.

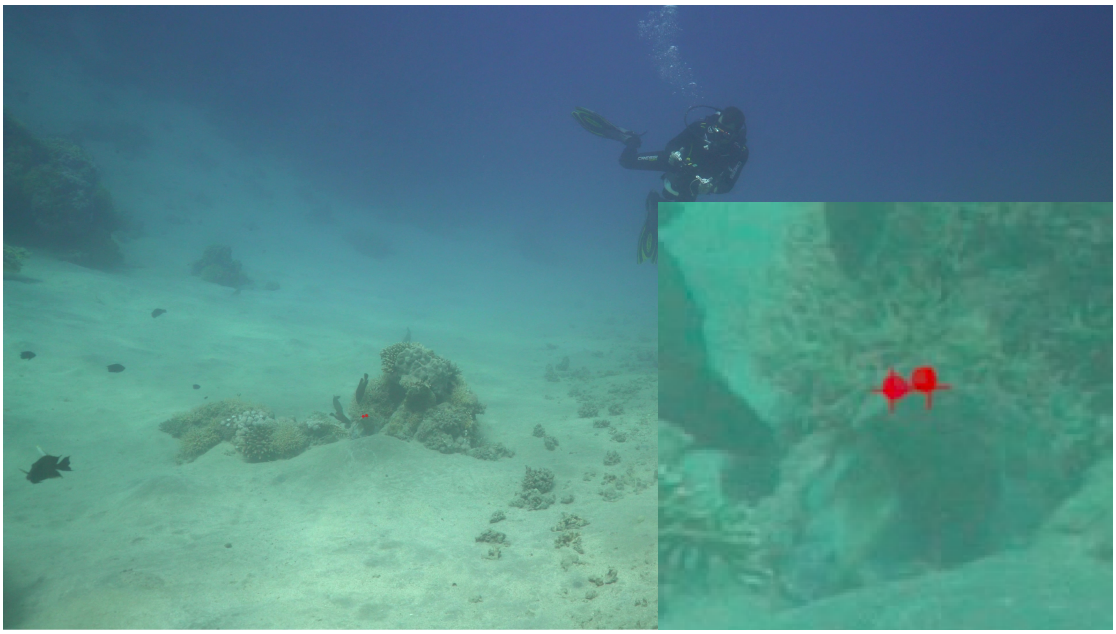

Figure 5.16: Comparison between manually labelled (+) and DLC labelled (•) in *ZeManuel* frame.

In Figure 5.17 we can observe the result of the training process. In this video, we save snapshots every 50000 iterations to select the best snapshot for the video analysis and to be able to understand the behaviour of the NN since we consider this a difficult video. We can observe that all errors are bouncing until the 450000 iterations, iteration where all start to stabilize.

After the stabilization of the error, the train error is preserved with 60 pixels of error and the train error with p-cut off approximately 20 pixels better at 40 pixels of error.

Test error is kept steady at 300 pixels of error and test error with *p–cutoff* in the 180 pixels of error.

This abrupt break in the bounce indicates that the NN reached a balanced point.

Any snapshot between the iteration 400000 and 1000000 should have similar results when analysing the video. We chose iteration 500000 to analyse our video.

Then we run the DLC functions, analyse and create labelled video, and we obtained the labelled video along with the positions of the body parts in each frame. After the validation process we get the results present in Table 5.8.

Table 5.8: Iteration 500000 of *ZeManuel* - Train, Test and Validation Error Summary.

	Train	Test	Validation
Error	94.98	313.48	490.17
Error with *p–cutoff*	42.24	170.72	92.63

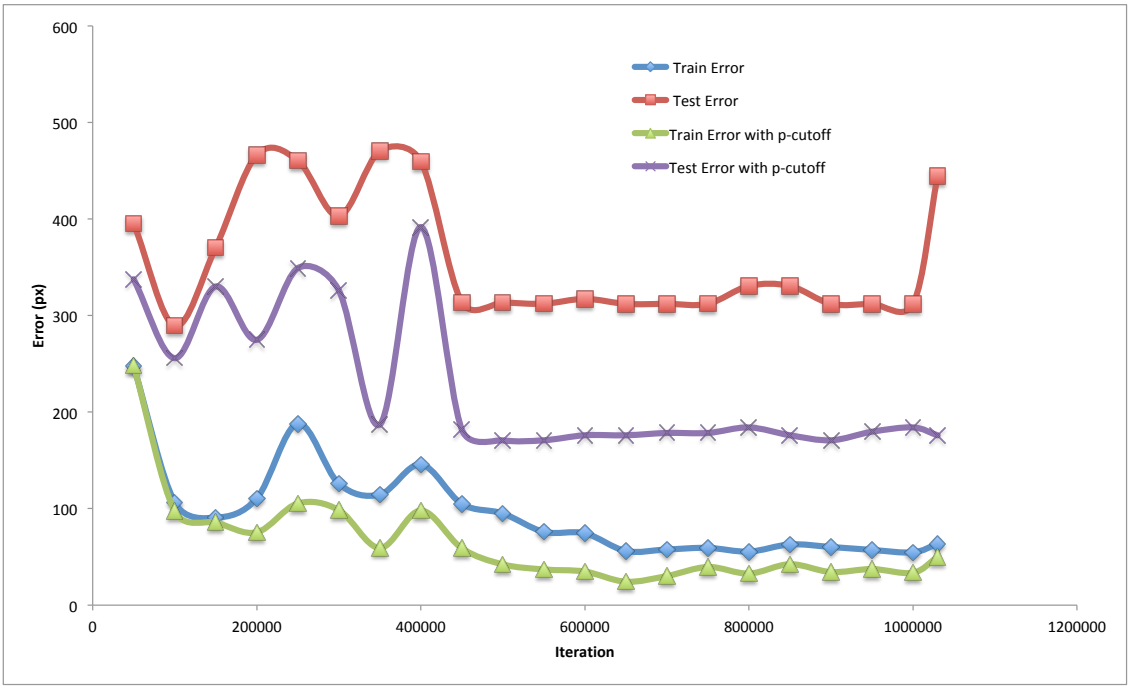

Figure 5.17: Train and Test error, Train and Test error with *p–cutoff* of *ZeManuel*.

We can verify that in all categories the error with *p–cutoff* is lower than the error, this is expectable since the NN is more likely to get the perditions right when having high confidence.

In the train and test, the difference between errors is significant and understandable, the error is in both low for this type of complex video. Since we are dealing with a difficult video we were expecting the validation error and the validation error with *p–cutoff* to be big. In this case, the validation error with *p–cutoff* is actually smaller than what we predict.

The average confidence during the whole video is approximately 0.55.

We also visualize the labelled video and consider that this video was not labelled with a human level precision. During the video, some parts were labelled with human-level precision but in particular, when the octopus is hiding the label turns very bouncing between different points.

Since we are dealing with a complex video we might have to expand the manually labelled dataset by increasing the frequency of frames per second.

5.5.4 *ZeMarco*

We chose this video from the Stereocamera Rig because it was the video where we caught one of the more interesting cooperations between the octopus and other fish.

The video background has both sandy areas and coral patches, the octopus is one of the bigger that we film in cooperation with other fish but it was filmed from a really big distance from the octopus. Visually analysing the video we understand that the

background is mainly sand but that the octopus prefers to hunt in the coral patches on top of the sand. It is a complex background. We can also notice in this video clear cooperation indications of a Yellow-edged lyretail grouper (*Variola louti*) toward the octopus and the octopus following those indications. This video was filmed at hight depth, approximately 30 meters, we can observe that the colours are already very different from the originals. Octopus, corals, stones and sand all look the same colour with different shades.

In this video, we ask the NN to identify both octopus eyes. This video is very hard for the NN to label, it has a complex background, is filmed from a big distance and the octopus is almost the same colour as the coral and rocks, making the octopus harder to identify and beyond that the octopus spend a lot of time under or inside the coral patches and does not always come out from the same spot it enter .

In Figure 5.18 we can observe the difficulty, the features of this video and the differences between manually labelled (+) and DLC labelled (•) in *ZeMarco* frame.

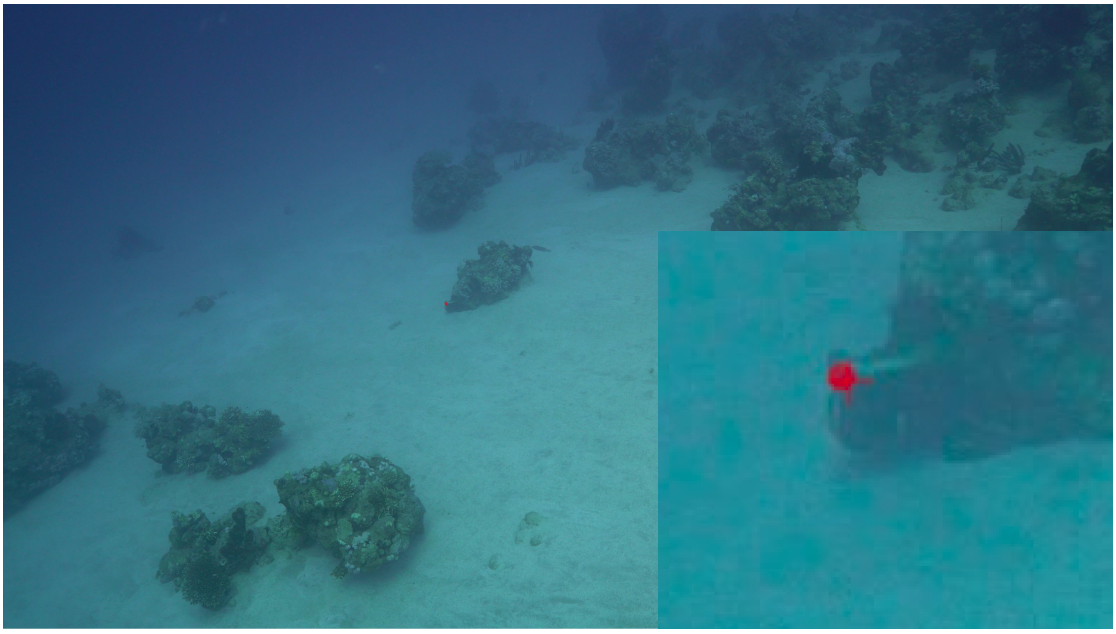

Figure 5.18: Comparison between manually labelled (+) and DLC labelled (•) in *ZeMarco* frame.

In Figure 5.19 we can observe the result of the training process. In this video, we save snapshots every 200000 iterations to select the best snapshot for the video analysis. We can observe that train error and error with *p–cutoff* have similar behaviour with approximately 100 pixels of difference. Both errors start to drop until the 600000 iteration where it stabilizes near the 200 pixels of error and near 60 pixels of error with *p–cutoff*. This result indicates that the NN reached a plateau level and that train errors will not decrease more.

In terms of test error and test error with *p–cutoff*, the behaviour very interesting. The error increases until reaches the 400000 iterations and then it starts to drop until the 600000 iterations for test error and for test error with *p–cutoff*.

This behaviour is starting to be common in the test results, the error drops until a certain iteration, the test error with *p–cutoff* stabilizes and the test error increases again, overfitting the NN. In this case, we have the particularity that the error drops again after the 800000 iteration, this difference may be due to the difficulty of the video and that the NN is still improving even at those iterations.

This video also has the particularity that until the iteration 400000 the test error with *p–cutoff* has a higher value than the test error. This may be linked to low confidence at the beginning of the iterations and that *p–cutoff* is excluding several good labelled frames because it has low confidence in them and is accepting badly labelled frames with high confidence.

We assume that the best snapshot to analyse the video based on the analysis done above is iteration 1030000, where the test error is lower.

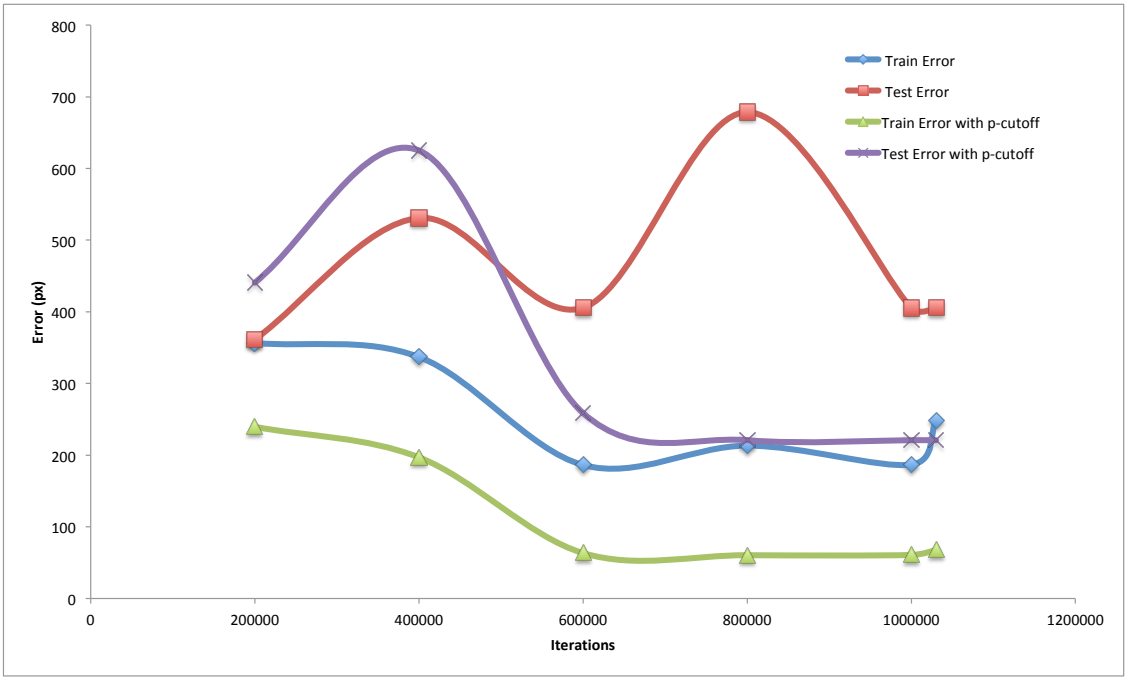

Figure 5.19: Train and Test error, Train and Test error with *p–cutoff* of *ZeMarco*.

Then we run the DLC functions, analyse and create labelled video, and we obtained the labelled video along with the positions of the body part in each frame. After the validation process we get the results present in Table 5.9.

Table 5.9: Iteration 1030000 of *ZeMarco* - Train, Test and Validation Error Summary.

	Train	Test	Validation
Error	248.1	405.57	990.29
Error with *p–cutoff*	67.7	220.91	17.15

We can verify that in all categories the error with *p–cutoff* is lower than the error, this

is expectable since the NN is more likely to get the perditions right when having high confidence.

In the train and test, the difference between errors is significant and understandable, the error value in both is very low for this type of complex video. Since we are dealing with a very complex video we were expecting that the validation error to be big, but the validation error with *p–cutoff* is much higher than expected.

The average confidence during the whole video is approximately 0.52.

We also visualize the labelled video and consider that this video was not labelled with a human level precision. During the video, some parts were well labelled but we can not say that the video is well labelled due to de presence of too much mislabelled frames. With the complexity of this video, we probably need to increase the number of labelled frames per video.

5.6 Results of Strategy 3

We test two videos with Strategy 3, *MartimPedras1* and *ZeSimao*, both from the Stereocamera Rig, we rename this videos to *MartimPedras30* and *ZeSimao30* respectively to simplify identification. These videos have similar background complexities but different lengths so different number of manually labelled frames per minute of video.

We present the results of each individually in this section.

5.6.1 *MartimPedras30*

We chose this video to retest because it had really good results with Strategy 1 so we wanted to test how much we could reduce the dataset without compromising the good results. In Figure 5.20 we can observe the result of the training process with only 30 manually labelled frames from each camera. In this video, we save snapshots every 200000 iterations since we already know what the NN is cable in this video. First of all, we verify that the result is outstanding and that the NN was able to have the same error with and without the *p–cutoff*, meaning that the NN is very confident of the predictions.

We can observe that both train and test error behave the same way, getting smaller with the increment of iterations.

Pointing that the NN was improving the label of not only the train frames but also of the test frames.

To try to analyse this video we choose the iteration 1000000 to do it.

Then we run the DLC functions, analyse and create labelled video, and we obtained the labelled video along with the positions of each body part in each frame. After the validation process, we get the results present in Table 5.10.

We can verify that the train error and train error with *p–cutoff*, are the same as explained above. The same occurs with test error and test error with *p–cutoff*. The important part, in this case, is the validation errors. We can observe that the validation errors are much higher than the train and test errors. We link this to the low number of frames to

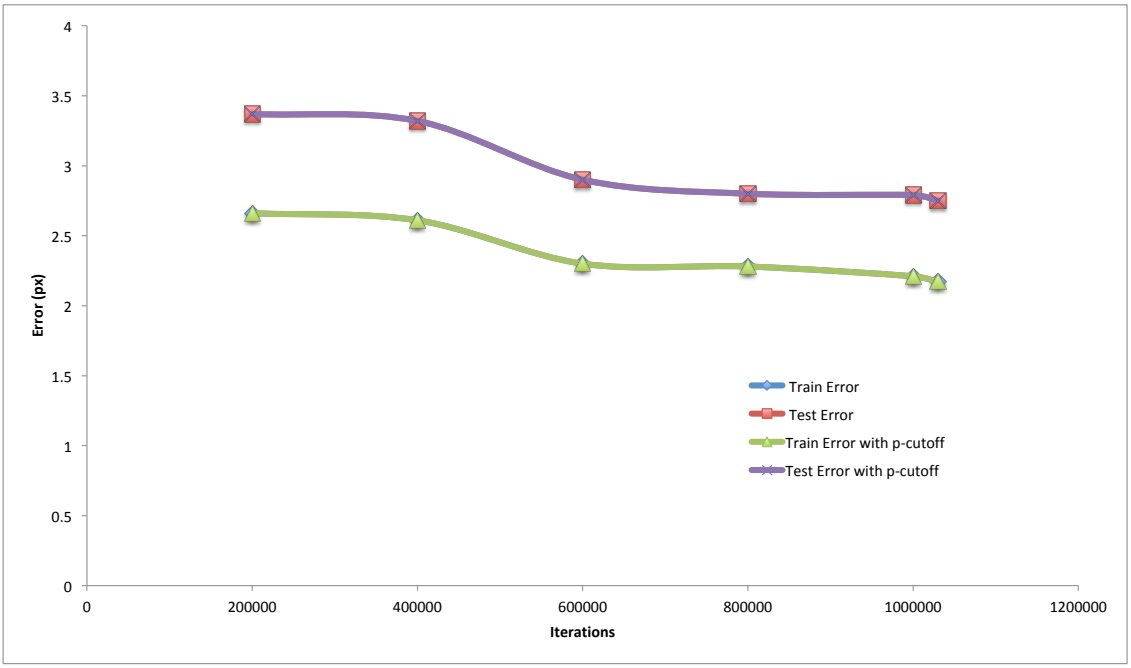

Figure 5.20: Train and Test error, Train and Test error with *p–cutoff* of *MartimPedras30*.

Table 5.10: Iteration 1000000 of *MartimPedras30* - Train, Test and Validation Error Summary.

	Train	Test	Validation
Error	2.21	2.79	53.97
Error with *p–cutoff*	2.21	2.79	19.32

evaluate, we think that the NN got really fitted to the train frames and because of that, the train errors is very small. Moreover, since we had a small data set the test dataset would be even smaller, in this case, the test dataset only had 3 images to evaluate. The validation error is high but acceptable since this NN was trained with limited manually labelled frames. The validation error with *p–cutoff* is a very good result, meaning that even that the NN is making some bad predictions the NN is able to give low confidence to those predictions.

The average confidence during the whole video is approximately 0.95.

We visualize the labelled video, compare it to *MartimPedras1* video and taking into account the values presented in Table 5.4 we consider that this video was labelled with human level precision with low manual labelled effort.

We need to keep in mind that this video is considered simple and have a short length. *MartimPedras30* was analysed with 30 manually labelled frames per minute of video. *ZeSimao* was with 24 manually labelled frames per minute of video. So with these results, we aspired to apply the same strategy to ZeSimao, the other simple video.

5.6.2 *ZeSimao30*

We chose this video to retest because it also had really good results with Strategy 2 and is very similar to MartimPedras so we wanted to test if we could reduce the dataset even without compromising the good results. In Figure 5.21 we can observe the result of the training process with only 30 manually labelled frames from each camera. In this video, we save snapshots every 200000 iterations since we also know what the NN is cable in this video. We notice that the results were very similar to the other results of strategy 3. We verify that the NN was able to have the same error with and without the *p–cutoff*, meaning that the NN is very confident of the predictions.

We can observe that both train and test error behave the same way as well, test errors had a bigger descend at the beginning but the general behaviour is getting smaller with the increment of iterations.

To try to analyse this video we choose the iteration 1030000 to do it.

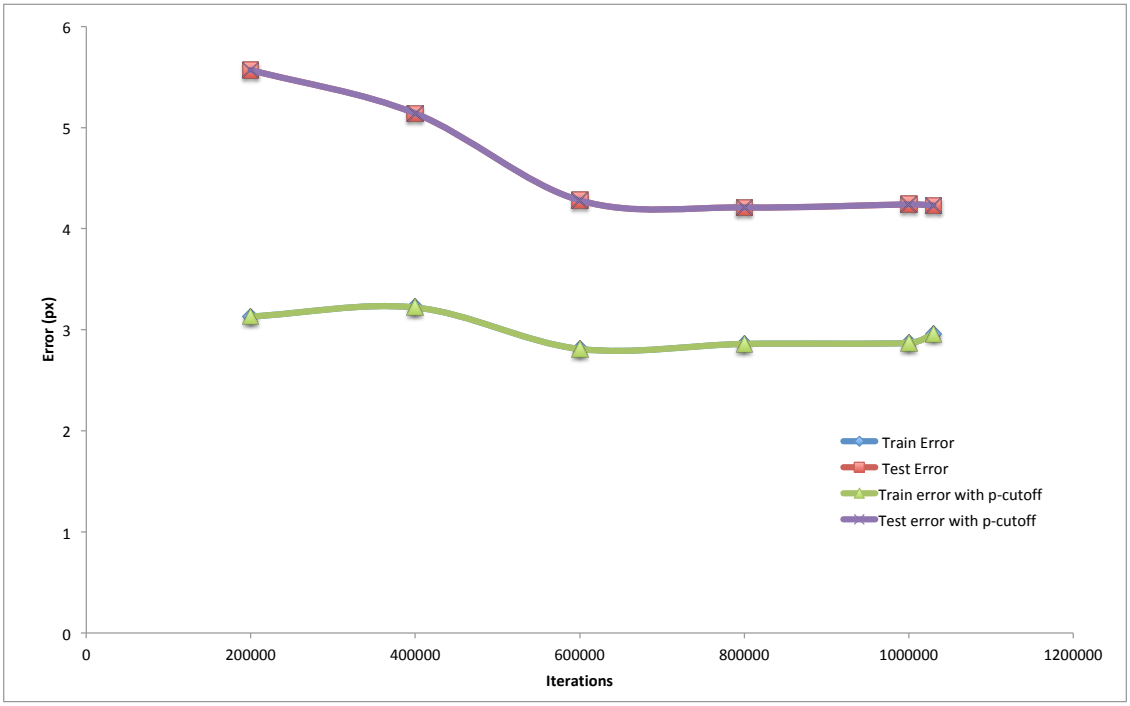

Figure 5.21: Train and Test error, Train and Test error with *p–cutoff* of *ZeSimao30*.

Then we run the DLC functions, analyse and create labelled video, and we obtained the labelled video along with the positions of each body part in each frame. After the validation process, we get the results present in Table 5.11.

We can verify that the train error and train error with *p–cutoff*, are the same as explained above. The same occurs with test error and test error with *p–cutoff*. The important part, in this case, is the validation errors. As we observed in *MartimPedras30*, we can observe that the validation errors are much higher than the train and test errors.

We give the same explanation to this video as we did to *MartimPedras30*. We link this

Table 5.11: Iteration 1030000 of *ZeSimao30* - Train, Test and Validation Error Summary.

	Train	Test	Validation
Error	2.96	4.23	971.93
Error with *p–cutoff*	2.96	4.23	845.89

to the low number of frames to evaluate, we think that the NN got really fitted to the train frames and because of that, the train errors is very small. Also since we had data set of the same size the test dataset would only have 3 images to evaluate as well. The validation error is too high to be acceptable and the validation error with *p–cutoff* is also too high, meaning that even with the *p–cutoff* the NN is making to many bad predictions.

The average confidence during the whole video is approximately 0.89

We visualize the labelled video and compared it to ZeSimao video. The differences are obvious but nevertheless, the *ZeSimao30* labelled video is not too bad as the validation values anticipate. There are several minutes where the octopus is correctly labelled but then there are some parts where the NN lost its track.

Taking into account the values presented in Table 5.11 we consider that this video was labelled reasonably taking into consideration the low manual labelled effort.

We need to keep in mind that this video is considered simple, but it is a long video. ZeSimao was analysed with 24 manually labelled frames per minute of video. *ZeSimao30* was analysed with just 2 manually labelled frames per minute of video. These results indicate us that 24 frames per minute are very good for simple videos but 2 frames per minute it too few frames, we need to find a good balance between the results and the manually labelled effort.

5.7 Two videos, one NN

In this section, we present the result of combining in the same NN two different videos.

We chose both videos from the Stereocamera Rig and from Strategy 1 so the videos chosen were *MartimPedras1* and *MartimCorais*. We chose these two videos because both were filmed at a low depth and we can clearly distinguish octopus in the image.

We needed to adjust the labels so that both have the same body parts. We use three points, both eyes and the tip of the mantel. We only use the manually labelled frames of one camera in the original data set.

We wanted to test if the NN had more difficulties to label the body parts from both videos or if on the other hand, the NN labelled the body parts with more easiness.

In Figure 5.22 we can observe the result of the training process. In this video, we save snapshots every 200000 iterations with the intention of selecting the best one for the video analysis. We can observe that both train and test error and train and test error with *p–cutoff* behave the same way but, the train errors have lower values on the first 600000 iterations.

This result indicates that the NN is identifying well almost all of the body parts and have high confidence in all.

After the 600000 iterations all four values are essentially the same and near the Human label error, around 3 pixels.

We assume that the best snapshot to analyse the video based on the analysis of the error is the iteration 1030000, where the test error is lower.

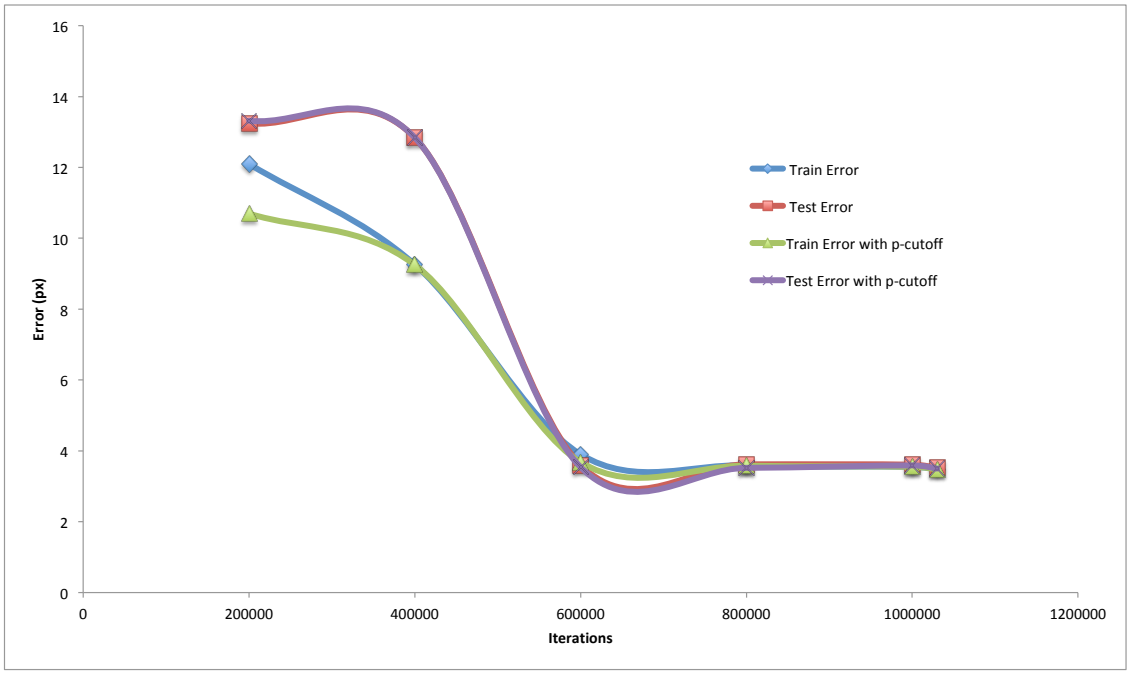

Figure 5.22: Train and Test error, Train and Test error with *p–cutoff* of *ZeTudo*.

Then we run the DLC functions, analyse and create labelled video, and we obtained the labelled video along with the positions of each body part in each frame. After the validation process we get the results present in Table 5.12

Table 5.12: Iteration 1030000 of *ZeTudo* - Train, Test and Validation Error Summary

	Train	Test	Validation
Error	3.48	3.52	16.74
Error with *p–cutoff*	3.48	3.5	16.74

We can verify that in all categories the error and the error with *p–cutoff* is essentially the same, this means that the NN has high confidence in the labelling.

Train and test errors are very small and similar to the human labelled error so this value will never drop more. The validation errors are low and very acceptable for the NN. If we compare to the validation error of the NN that was trained with just one video the value is the average of the two. This means that the NN had correctly the same performance while being trained with double the information.

The average confidence during the *MartimPedras1* whole video is approximately 0.99 and the average confidence during the *MartimCorais* whole video is approximately 0.92.

We also visualize both labelled videos and consider that these videos were also labelled with a human level precision.

Even though the same NN was able to label two different videos its was not able to correctly label a different video from the two initial ones.

We try to analyse *ZeSimao* with this NN but the labelled video was not acceptable.

5.8 Results of The Other Potentials Usages

5.8.1 *MartimZoom*

We chose this video from the Zoom Camera to test how good the NN was in small details and to see if it could follow the colour and shape changes in the octopus body. Visually analysing the video we can notice that the background is made of small rocks with some coral on top. We know from the dive that this video was filmed at low depth (5 meters), so the colours of the video are very similar to the original ones. This video has the particularity that we were filming really close to the octopus, this may interfere with his natural behaviour but for this test it was perfect. The octopus during almost all the video has a brownish colour with some white spots that appear and disappear. Its shape and texture also change several times during this video

In Figure 5.23 we can observe the difficulty, the features of this video and the differences between manually labelled (+) and DLC labelled (•) in MartimZoom frame. We can also observe which body parts we are tracking.

Figure 5.23: Comparison between manually labelled (+) and DLC labelled (•) in MartimZoom frame.

In Figure 5.24 we can observe the result of the training process. In this video, we save snapshots every 50000 iterations to select the best snapshot to analyse the video with the NN. We can observe that both train error and train error with *p–cutoff* behave in a very identically, showing that the NN easily identify the body parts in the train dataset with high confidence. There are just some minor peaks in the train error.

In terms of test error and test error with *p–cutoff*, the behaviour is notably more distinct. The test error with *p–cutoff* in the first 650000 iterations is stable ate approximately 15 pixels of error but then after a drop and a peak the error stabilize again but this time at approximately 3 pixels of error.

In the test error, the error did not stabilize during the 1030000 iterations of the training This indicates that the NN is trying to change the weights of the layers in order to try to predict better the body parts without high confidence. Since we are asking the NN to label several distinct body parts the error is the average of the error of each body parts, so if the NN is able to improve one body part but deteriorate another, the general error will remain the same.

We associate these variations on the test error due to the high number of body parts to label.

To analyse the video with the best results from the NN we choose the iteration 850000 to do it.

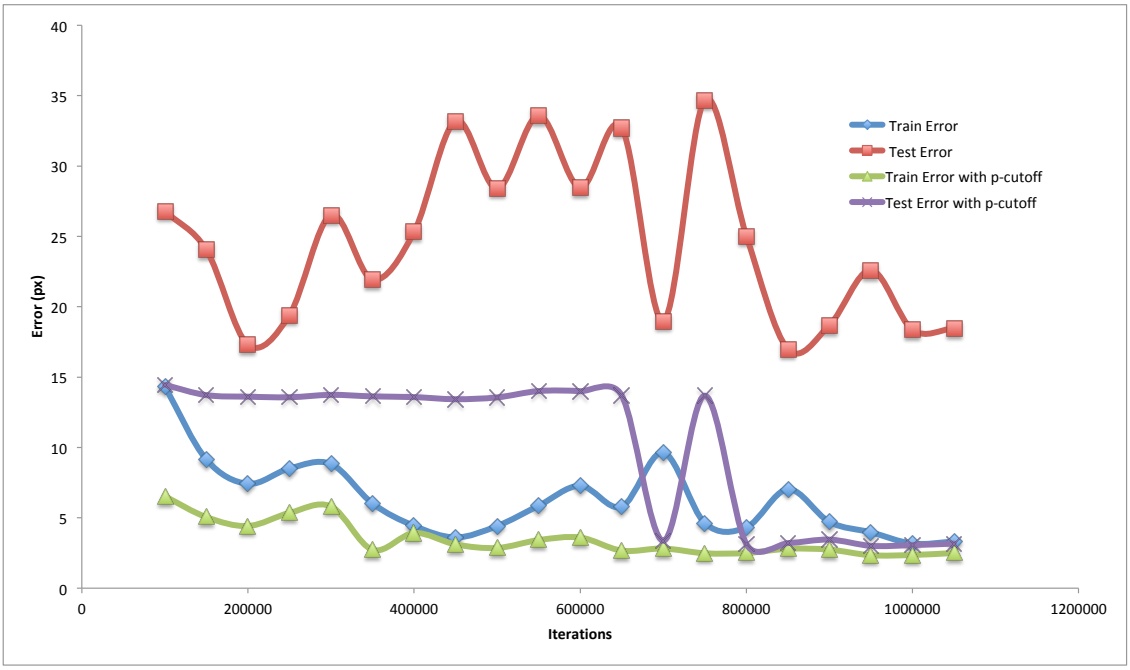

Figure 5.24: Train and Test error, Train and Test error with *p–cutoff* of MartimZoom.

Then we run the DLC functions, analyse and create labelled video, and we obtained the labelled video along with the positions of the body part in each frame. After the validation process, we get the results present in Table 5.6.

Table 5.13: Iteration 850000 of MartimZoom - Train, Test and Validation Error Summary.

	Train	Test	Validation
Error	7.03	16.99	3.04
Error with *p–cutoff*	2.9	3.2	3.04

We can verify that the error with *p–cutoff* in the train and in the test is the lower that the error as expectable. In the case of validation, the error is the same in both. All categories have a low error with insignificant differences between the error with *p–cutoff*. The average confidence during the whole video and all body parts is approximately 0.96 but the confidence of the easiest point, the centre of the eye the confidence is 0.99.

We visualize the labelled video and taking into account the values presented in Table 5.13 we consider that this video was labelled with human level precision.

5.8.2 *ZeBrunoZoom*

We chose this video from the Zoom Camera to test if the NN maintain the good results in small details if the video is a lit bit further y from the octopus body. By visual analysis of the video, we can notice that the background is slightly different, made of a big coral patch. We know from the dive that this video was filmed at medium depth (10 meters), so the colours of the video already start to change from the original ones. The octopus during a big part of the video has a brownish colour with some white spots that appear and disappear and another part where the octopus is pale.

Its shape and texture also change several times during this video usually at the same time that the colour change.

In Figure 5.25 we can observe the difficulty, the features of this video and the differences between manually labelled (+) and DLC labelled (•) in *ZeBrunoZoom* frame. We can also observe which body parts we are tracking. In the Zoomed in part we can observe that the NN is labelling with low confidence (×) the right eye with the left eye markers (green and light blue).

In Figure 5.26 we can observe the result of the training process. In this video, we save snapshots every 50000 iterations to select the best snapshot to analyse the video with the NN. We can observe that both train error and train error with *p–cutoff* behave in a very identically, showing that the NN easily identify the body parts in the train dataset with high confidence.

In terms of test error and test error with *p–cutoff*, they have the opposite behaviour.

In the test error, the error oscillates during the first 900000 iterations of the training having an average value of around 180 pixels of error, and when it stabilizes the error is still bigger than 150 pixels of error. This indicates that the NN is trying to change the weights of the layers in order to try to predict better the body parts without high confidence.

Figure 5.25: Comparison between manually labelled (+) and DLC labelled (•) in *ZeBruno-Zoom* frame.

The test error with *p–cutoff* during the 1030000 iterations have several local minimums and several peaks. We link this behaviour to the NN trials to improve the minimum error. We can also observe that the local minimums all have similar errors, around 100 pixels of error. In the 950000 iterations, we think that the test error with *p–cutoff* finally stabilizes, but since we do no have more iterations this plateau may be in fact another local minimum.

Since we are asking the NN to label several distinct body parts the error is the average of the error of each body parts, so even if the NN is able to well label one body part but there is one that the NN is not able to well label the error value will increase in general.

We associate these variations on the test error due to the high number of body parts to label.

Analysing both test errors we can assume that the best snapshot to analyse the videos is iteration 950000.

Then we run the DLC functions, analyse and create labelled video, and we obtained the labelled video along with the positions of the body part in each frame. After the validation process, we get the results present in Table 5.14.

Table 5.14: Iteration 950000 of *ZeBrunoZoom* - Train, Test and Validation Error Summary.

	Train	Test	Validation
Error	20.3	168.25	762.65
Error with *p–cutoff*	9.27	99.9	3.07

We can verify that the error with *p–cutoff* in the train and in the test is the lower

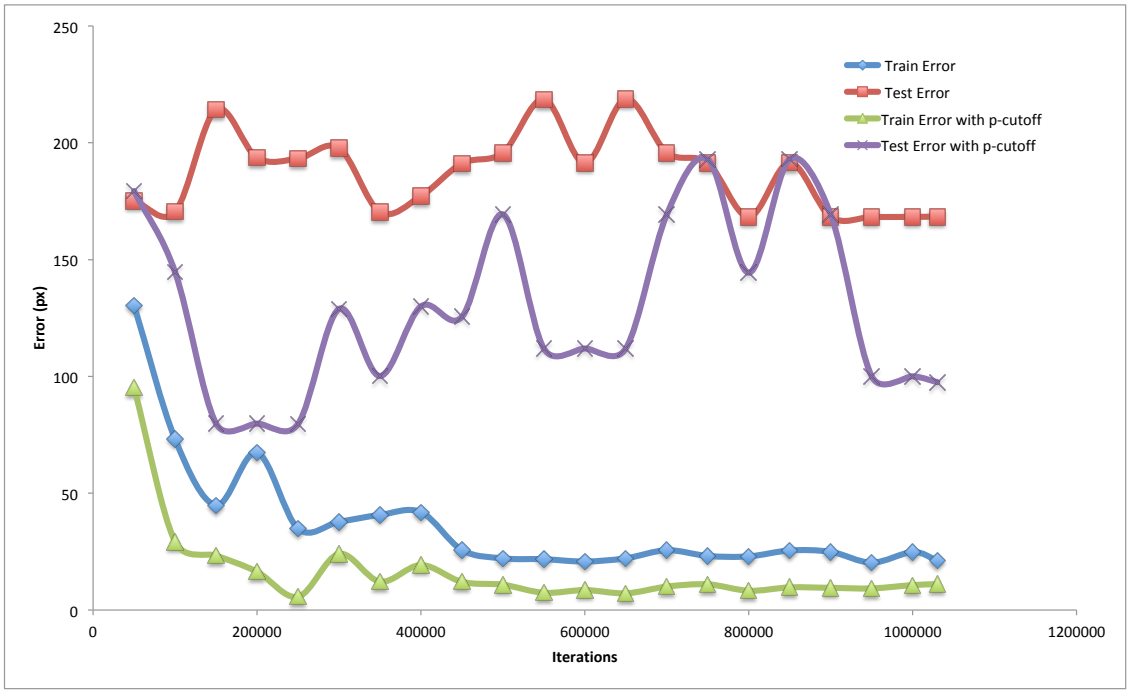

Figure 5.26: Train and Test error, Train and Test error with *p–cutoff* of *ZeBrunoZoom*.

that the error as expectable. In the case of validation, the error is the same in both. All categories have a low error with insignificant differences between the error with *p–cutoff*. The average confidence during the whole video and all body parts is approximately 0.38 but the confidence of both points of the right eye is more than 0.50, this eye is the one that is facing the camera during the majority of the video, the left eye only appear in some frames so the confidence of those two points is under 0.24. This abundance of body parts may be misleading the train of the NN.

We visualize the labelled video and taking into account the values presented in Table 5.13 we cannot consider that this video was labelled with human level precision. Even though the NN is able to mark at least one body part during almost the whole video.

Although this is a video from the Zoom Camera the complexity of the background, the occlusions of the octopus and the different luminosities during the video make us define this video as a difficult video even for manual labelling so we consider these results as very good ones.

Chapter 6

Conclusions

In this work, we were able to train different NN to achieve human level precision in the tracking of octopuses in stereoscopic videos in unstructured environments. This was a long learning process, with trial and error periods. The successful tracking of an octopus was only achieved after several unsuccessful trials. Finally, perseverance on the choice of tools and the correct pipeline and configuration lead us to the appropriate values of the parameters to use.

In the cases where we were not able to achieve human level precision in the tracking process, we still manage to get some useful results. Sometimes the algorithm loses track of the animal or some body parts, because of occlusion or dark situations, but successfully recovers, once the (correct) circumstances are regained. In other cases, the NN was not able to correctly label the video, as the case of *ZeSimao30*. In these cases, to successfully track the animal, the solution consists on having more manually labelled frames in the initial data set.

We also realised that labelling body parts that are not visible during a big part of the video, may have a negative effect on labelling confidence. In future tests, both eyes of the octopus should be labelled, and avoid to use the siphon or the tip of the mantel, since we got low confidences in these two body parts, contrary to what was verified in the eyes.

We were able to confirm our theory that the more complex the video is, the more frames per minute of video we have to label.

We can compare the case of *MartimPedras1* and *ZeSimao* with the case of *MartimCorais* and *ZeSousa*. In both pairs, the videos are very similar in all aspects of the video except the duration. The first video of each pair only have 2 minutes, the second had more than 25 minutes. In the case of *MartimPedras1* and *ZeSimao*, the videos are very simple so the difference of frames per minute does not affect the results. In the case of *MartimCorais* and *ZeSousa*, the videos are more complex and the reduction of frames per minute affect

significantly the results.

Thus we conclude that we can achieve human level precision in all videos but we would need to manually label more frames depending on the video complexity.

6.1 Future Work

After the train of the NN for one specific video, we need to calibrate the videos from both cameras by sound.

Once calibrated, the Stereocamera Rig allows for reliable and accurate 3D-tracking of overall group collective movement and 3D reconstruction of habitat features.

First, using computer-vision-based methods, videos are run through a Structure-from-Motion and Multi-View Stereo pipeline with the software colmap [22]. This allows for a high-resolution 3D spatial reconstruction, where all habitat features across hunting events are highly detailed [15]. More importantly, this approach also yields the position of the camera relative to the reconstructed habitat at all times [18], thus taking in account camera movement when tracking the animals. Second, we use the already trained NN to perform automated tracking of all players involved in the calibrated collaborative hunting events videos [17]. Third, individual positions are triangulated from the Stereocamera Rig's two viewpoints, and their movements reconstructed in 3 dimensions. After subtracting camera motion obtained during habitat reconstruction, we obtain meaningful trajectories in real-life coordinates.

Finally, these tracks are then superimposed on the 3D habitat, providing a reliable and highly quantitative reconstruction of the group's collective movement through natural habitats [19].

Concurrently, the Zoom Camera is focused on *O. cyanea* movements and close-by partners. This is essential to classify octopus behaviour since cephalopods have more flexible behavioural motor expressions than fishes [16]. We will manually or with the help of JAABA[1] [9] classify behaviours and categorize movement patterns from *O. cyanea* and fine-scale events (e.g. webbing over corals, "punch" to nearby fish), thus developing a foraging ethogram (a similar ethogram is developed for fishes, relying on body movement).

Finally, as all videos are synchronized, the outputs of both collective movement and individual behaviour are superimposed on the tracking coordinates, providing us with a highly quantitative, and accurate description of changes in movement due to specific behavioural elements.

[1] The Janelia Automatic Animal Behaviour Annotator (JAABA) is a system that allows researchers to automatically calculate interpretable quantitative statistics achieved by machine learning analysis of videos of behaving animals.

To use this system, we need to manually label small frames series and describe the animal behaviour in those frames. This will allow the system to convert these manual labels into behaviour detectors using machine learning techniques. These detectors will be used later to automatically classify, in a large data set, the animal behaviour with high throughput.

The JAABA system combines an intuitive graphical user interface, a powerful and fast machine learning algorithm and the possibility to the visualization of the classifier into an interactive, detail and usable scientifically meaningful measurements of behavioural effects in large experiments.

References

[1] J. A. Mather and R. C. Anderson. "Personalities of octopuses (Octopus rubescens)." In: *Journal of Comparative Psychology* 107.3 (1993), pp. 336–340. DOI: 10.1037//0735-7036.107.3.336.

[2] C. B. Albertin, O. Simakov, T. Mitros, Z. Y. Wang, J. R. Pungor, E. Edsinger-Gonzales, S. Brenner, C. W. Ragsdale, and D. S. Rokhsar. "The octopus genome and the evolution of cephalopod neural and morphological novelties." In: *Nature* 524 (Aug. 2015), pp. 220–224. DOI: 10.1038/nature14668.

[3] F. Bellard. *ffmpeg*. Accessed: 2019-07-22. 2006. URL: https://ffmpeg.org.

[4] T. K. Bruce D. Lucas. "An Iterative Image Registration Technique with an Application to Stereo Vision." In: *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81)*. Aug. 1981, pp. 674–679. URL: http://dl.acm.org/citation.cfm?id=1623264.1623280.

[5] G. Fiorito, A. Affuso, J. Basil, A. Cole, P. de Girolamo, L. D'Angelo, L. Dickel, C. Gestal, F. Grasso, M. Kuba, F. Mark, D. Melillo, D. Osorio, K. Perkins, G. Ponte, N. Shashar, D. Smith, J. Smith, and P. L. Andrews. "Guidelines for the Care and Welfare of Cephalopods in Research –A consensus based on an initiative by CephRes, FELASA and the Boyd Group." In: *Laboratory Animals* 49.2_suppl (Sept. 2015), pp. 1–90. DOI: 10.1177/0023677215580006.

[6] S. Gietler. *Underwater Lighting Fundamentals*. Accessed: 2018-11-22. 2018. URL: http://www.uwphotographyguide.com/underwater-photography-lighting-fundamentals.

[7] E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, and B. Schiele. "DeeperCut : A Deeper , Stronger , and Faster Multi-Person Pose Estimation Model." In: *European Conference on Computer Vision (ECCV)*. Sept. 2016, pp. 34–50. DOI: 10.1007/978-3-319-46466-4_3.

[8] E. Insafutdinov, M. Andriluka, L. Pishchulin, S. Tang, E. Levinkov, B. Andres, and B. Schiele. "ArtTrack: Articulated Multi-Person Tracking in the Wild." In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017, pp. 1293–1301. DOI: 10.1109/CVPR.2017.142.

[9] M. Kabra, A. Robie, M. Rivera-Alba, S. Branson, and K. Branson. "JAABA: Interactive machine learning for automatic annotation of animal behavior." In: *Nature Methods* 10 (Dec. 2012). DOI: 10.1038/nmeth.2281.

[10] J. Mather and J. S. Alupay. "An Ethogram for Benthic Octopods (Cephalopoda : Octopodidae)." In: *Journal of Comparative Psychology* 130(2) (Apr. 2016). DOI: 10.1037/com0000025.

[11] A. Mathis, P. Mamidanna, K. M. Cury, T. Abe, V. N. Murthy, M. W. Mathis, and M. Bethge. "DeepLabCut: markerless pose estimation of user-defined body parts with deep learning." In: *Nature Neuroscience* 21 (Aug. 2018). DOI: 10.1038/s41593-018-0209-y.

[12] J Messenger. "Cephalopod chromatophores: Neurobiology and natural history." In: *Biological reviews of the Cambridge Philosophical Society* 76.4 (Mar. 2001), pp. 473–528. DOI: 10.1017/S1464793101005772.

[13] T. Nath, A. Mathis, A. C. Chen, A. Patel, M. Bethge, and M. W. Mathis. "Using DeepLabCut for 3D markerless pose estimation across species and behaviors." In: *Nature protocols* 14.7 (June 2019), pp. 2152–2176. DOI: 10.1038/s41596-019-0176-0.

[14] F. Romero-Ferrero, M. G. Bergomi, R. Hinz, F. J. H. Heras, and G. G. de Polavieja. "idtracker.ai: Tracking all individuals in large collectives of unmarked animals." In: *Nature Methods* 16 (Jan. 2019), pp. 179–182. DOI: 10.1038/s41592-018-0295-5.

[15] E. Sampaio. *3D spatial reconstruction*. Accessed: 2019-11-07. 2019. URL: https://youtu.be/j4vV0jv44aE.

[16] E. Sampaio. *Automated tracking body features of an octopus*. Accessed: 2019-11-07. 2019. URL: https://youtu.be/K9bUkN9ODWw.

[17] E. Sampaio. *Automated Tracking octopus, wrasse, and black tip grouper through stereo-cameras*. Accessed: 2019-11-07. 2019. URL: https://youtu.be/LLzjHv18Wqs.

[18] E. Sampaio. *reconstructed habitat*. Accessed: 2019-11-07. 2019. URL: https://raw.githubusercontent.com/EduSampaio/OctoFishProject/master/3d_animation.gif.

[19] E. Sampaio. *Simplified 3D habitat, with animal tracks overlayed*. Accessed: 2019-11-07. 2019. URL: https://youtu.be/Lf-JvAbp9c0.

[20] J. Schindelin, I. Arganda-Carreras, E. Frise, V. Kaynig, M. Longair, T. Pietzsch, S. Preibisch, C. Rueden, S. Saalfeld, B. Schmid, J.-Y. Tinevez, D. J. White, V. Hartenstein, K. Eliceiri, P. Tomancak, and A. Cardona. "Fiji: an open-source platform for biological-image analysis." In: *Nature Methods* 9.7 (June 2012), pp. 676–682. DOI: 10.1038/nmeth.2019.

[21] W. Schwalb. *Open Vision Control*. Accessed: 2018-11-22. 2011. URL: http://openvisionc.sourceforge.net.

[22] J. Schönberger and J.-M. Frahm. "Structure-from-Motion Revisited." In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016. DOI: 10.1109/CVPR.2016.445.

[23] V. H. Sridhar. *Tracktor*. Accessed: 2018-10-02. 2017. URL: https://github.com/vivekhsridhar/tracktor.

[24] J. B. Wood and R. C. Anderson. "Interspecific Evaluation of Octopus Escape Behavior." In: *Journal of Applied Animal Welfare Science* 7.2 (June 2004), pp. 95–106. DOI: 10.1207/s15327604jaws0702_2.

ANNEX I

Program used to extract random frames for validation

We need to take in account the name of the video and change the input name of the video if the video name differ from aaaaa-Cx-1.mp4

a = random text

x = number of the camera (1 or 2 if stereocameras, 3 if zoomcamera)

In python :

```
from random import *
from sys import *
import os
minutes = int(input("minutes␣of␣the␣video:␣"))
seconds = int(input("seconds␣of␣the␣video:␣"))
ft = int(input("number␣of␣frames␣to␣extract:␣"))
time = minutes * 60 + seconds
nf = int(time*25)
frames = []
for frame in range(ft):
  print(frame)
  fra = randint(1, nf)
  C = randint(1,2)
  #Correct the input video name for each video
  os.system('ffmpeg␣-i␣Nameofthevideo-C' + str(C) + '-1.mp4␣-qscale:v␣1␣-vf␣"select=eq(n\,
      ↪ ' + str(fra) + ')"␣-vframes␣1␣C' + str(C) + str(fra) + '.jpg')
  frames.append(fra)
print(frames)
```

ANNEX II

Validation Process

In all the videos we use the validation process described. We start by extracting the frames and manually label them an then we search for the X, Y and confidence in the DLC output file for the video we want and we fill the spaces in the excel. We get the validation Error and the Validation Error with p-cutoff.

DLC confidence is presented with just 2 decimal numbers but the average DLC confifence was obtained with the full 9 decinal numbers and only then rounded to 2 decimal numbers for presentation purpuses.

When the octopus was not in the frame while manually labbeling we label that frame with $x = 0$ and $y = 0$.

Table II.1: Validation process of MartimCorais in order to explicate the process of validation of all the videos

MartimCorais		Manually Labelled		DLC Labelled		Euclidian Distance		Validation Error	Validation Error with p-cutoff	DLC Confidence
Frame	Camera	x	y	x	y	x	y			
554	1	1179	998	1179	999	0	1	1	1	0.99
780	1	1108	918	1108	916	0	2	2	2	0.99
1310	1	1576	900	1575	898	1	2	3	0	0.10
1353	1	1549	861	1548	855	1	6	7	7	0.99
1493	1	1620	583	1622	583	2	0	2	2	0.96
1728	1	1099	934	1101	936	2	2	4	4	0.99
1938	1	1343	1043	1349	1043	6	0	6	6	0.99
1994	1	943	1065	946	1064	3	1	4	4	0.99
2036	1	1407	1178	1409	1175	2	3	5	5	0.99
2156	1	1892	1855	1900	1856	8	1	9	9	0.99
2388	1	1419	868	1428	868	9	0	9	9	0.99
2463	1	1463	1212	1466	1211	3	1	4	4	0.99
2717	1	0	0	4	3	4	3	7	0	4.99E-05
2762	1	2071	1573	2071	1569	0	4	4	4	0.99
2777	1	1908	942	1909	940	1	2	3	3	0.99
222	2	2500	797	2499	798	1	1	2	2	0.99
305	2	2724	864	2725	867	1	3	4	4	0.99
319	2	2639	833	2638	831	1	2	3	3	0.99
432	2	2336	702	2337	698	1	4	5	5	0.99
449	2	2140	941	2137	938	3	3	6	6	0.99
471	2	1884	1004	1883	998	1	6	7	7	0.99
756	2	1550	722	1547	723	3	1	4	4	0.99
808	2	1803	916	1803	919	0	3	3	3	0.99
960	2	2045	850	2042	845	3	5	8	8	0.99
1849	2	1940	934	1939	935	1	1	2	2	0.99
2129	2	0	0	4	3	4	3	7	0	2.24E-05
2301	2	2338	858	2338	856	0	2	2	2	0.99
2344	2	2440	725	2439	724	1	1	2	2	0.99
2543	2	2723	1540	2724	1535	1	5	6	6	0.99
2657	2	2763	1938	2764	1934	1	4	5	5	0.99
Averages						2.13	2.4	4.53	4.41	0.90