Development of a non-invasive motion capture system for swimming biomechanics

G ASCENSO

PhD 2021

Development of a non-invasive motion capture system for swimming biomechanics

GUIDO ASCENSO

A thesis submitted in partial fulfilment of the requirements of Manchester Metropolitan University for the degree of Doctor of Philosophy

Department of Sport and Exercise Sciences; Visual Computing Lab

Manchester Metropolitan University

2021

Abstract

Sports researchers and coaches currently have no practical tool that can accurately and rapidly measure the 3D kinematics of swimmers. Established motion capture methods in biomechanics are not well suited for underwater use, either because they i) are not accurate enough (like depth-based systems, or the visual hull), ii) would impair the movement of swimmers (like sensor- and marker-based systems), or iii) are too time consuming (like manual digitisation). The ideal for swimming motion capture would be a markerless motion capture system that only requires a few cameras. Such a system would automatically extract silhouettes and 2D joint locations from the videos recorded by the cameras, and fit a generic 3D body model to these constraints. The main challenge in developing such a system for swimming motion capture lies in the development of algorithms for silhouette extraction and 2D pose detection (i.e., localisation of joints in image coordinates), which need to perform well on images of swimmers—a task that currently available algorithms fail. The aim of this PhD was the development of such algorithms. Existing datasets do not contain images of swimmers, making it impossible to train algorithms that would perform well in this domain. Therefore, during the PhD two datasets of images of swimmers were constructed and hand-labelled: one, called Scylla, for silhouette extraction (3,100 images); and one, called Charybdis, for 2D pose detection (8,000 images). Scylla and Charybdis are the first datasets developed specifically for training algorithms to perform well on images of swimmers. Indeed, using these datasets, two algorithms were developed during this PhD: FISHnet, for silhouette extraction; and POSEidon, for 2D pose detection. The novelty of FISHnet (which outperformed state-of-the-art algorithms on Scylla) lies in its ability to predict outputs at the same resolution as the inputs, allowing it to reconstruct fine-grained silhouettes. The novelty of POSEidon lies in its unique structure, which allows it to directly regress the x and y coordinates of joints without needing heatmaps. POSEidon is almost as accurate as humans at locating the spinal joints of swimmers, which are essential constraints onto which to fit 3D models. Using these two algorithms, researchers will, in the future, be able to assemble a markerless motion capture system for swimming, which will contribute to improving our understanding of swimming biomechanics, as well as providing coaches a tool with which to monitor the technique of swimmers.

Acknowledgements

I like to think that this thesis represents not only the culmination of my achievements as a PhD student, but also a chronicle of the relationships that, in the past three years, I built, developed, and maintained with a multitude of people, whom I am delighted to thank here.

Thank you mom and dad. I rarely show it, but I love you deeply, and I missed you during these three years abroad; I hope the future will allow us to be closer. Thank you Chiara, for still loving me though I have not come to Madrid yet. I love you like a sister: I'm always happy when we talk on the phone, happier when we are together. Thank you Simone, for dismissing the doubts I had in myself and encouraging me to push through them. Thank you also Albertino: you cannot read yet, but you have already taught me that I need to work hard if I want you to be proud of your uncle. Thank you Sławek and Kasia, for giving me a home away from home, and for countless delicious meals and beers. My love for you has grown as much as my waist size.

Thank you Lorenzo, Mike, Lez, Lollo, for the many hours spent together—sometimes winning, sometimes losing, but always united by a hatred for Yasuo players. Those hours spent together helped me keep my sanity.

Thank you Lorenza, Masi, but first and foremost Luo: at least one third of this PhD was only possible thanks to your efforts. Thank you Previ, for not being mad at me when I asked you for the seventieth time how to locate the shoulder joint centre. Thank you Vale, Pisce, Marco, Luca: from sharing snobby comments about movies, to learning how to deep fry a mozzarella, to failing to get into Old Trafford (I still cannot get over the hurt...), you have made these three years more enjoyable.

Thank you Antoine, for always being happy to lend a hand and have a chat; Matthieu, for being the chillest roommate I could have asked for; Matteo, for always being glad to remind me what true science looks like, and for inspiring me to apply to a PhD in the first place; Giulia, for the pictures of your dog, the memes, and... you know what; and Maria Luisa: without your therapy sessions I might still be in a dark place.

Thank you Carl. Your unwavering confidence in me brings me great pride and motivates me to work hard to deserve it. Thank you also for having gone beyond what most people would

have to smooth out all the bumps we encountered along the road: now my Maserati can roar freely across the finish line. Thank you Moi: you took a group of colleagues and transformed them into a family, and I am glad to be a part of it. Thank you Tom, for imprinting in me a scientific rigour that I was missing, both in research and in writing. Thank you Simon, for grounding me to reality. Again. And again. Your input was always appreciated when present, and missed when absent.

Thank you Connah and Sean: you helped me enough times to be counted as my fifth and sixth supervisors. Thank you Bill, Manu, Jhan, Jireh, Xulu, Chuin Hong: the hours spent in the lab never felt like chores, as I knew I was amongst people who were working just as hard as me—but who were also open to having a chat when the work was just too dull.

To all of you: Thank you for having been a part of my life during this journey.

And to you, Ala: thank you for giving me a future to look forward to every morning. Everything else I need to say to you belongs to your ears only, and I will keep saying it to you every day, forever.

Dedication

To Marcelle.

'This universe henceforth seems to him neither sterile or futile. Each atom of that stone, each mineral flake of that night filled mountain, in itself forms a world. The struggle itself towards the heights is enough to fill a man's heart. One must imagine Sisyphus happy.'

Albert Camus

Contents

List of Tables

List of Figures

Glossary

adversarial learning In an adversarial learning scheme, a network (called a generator) learns to produce realistic outputs, and a second network (called a discriminator) learns to recognise the difference between real outputs and generated outputs. The training continues until the generator can reliably 'fool' the discriminator. Then, the generator can be used to produce new, realistic data. 17

architecture (referred to a neural network) The way in which the layers are arranged, in terms of both the depth and the width of a neural network. 31

ASPP module A component of DeepLab, a state-of-the-art semantic segmentation algorithm. It consists of parallel dilated convolutional layers with different dilation rates, and its function is to capture features at different resolutions. In FISHnet (the silhouette extraction algorithm developed during this PhD), the ASPP module was re-purposed into the SRE module. 42

batch normalisation A technique used to re-centre the weights of a neural network's layer by normalising them with respect to the mean standard deviation of the layer's inputs. 109

bottleneck layer Technically a block and not a layer, it consists of the following layers: 1 x 1 convolution, n x n (typically n=3) convolution, 1 x 1 convolution. The purpose of bottleneck layers is to reduce the number of filters (via the 1 x 1 convolutional layer) before the computationally expensive 3 x 3 convolutional layer, and then restore the number of filters via a second 1 x 1 convolutional layer. xix, 121

capacity (of a neural network) In theory, it indicates the complexity of the function that the network can learn to approximate. In practice, it is often equated to the number of

parameters of the network, and networks with more parameters are said to have more capacity. 71

colour camouflaging A phenomenon that happens when an object and its background have similar colours, making it difficult to distinguish the two and to identify their boundary. xiii, 27

computer vision The field that studies how machines can gain a high-level understanding of images and videos. 22

convolutional layers Layers of a neural network that apply small filters (usually 1 x 1, 3 x 3, or 5 x 5) to all parts of an image. Convolutional layers are more computationally efficient than fully-connected layers and are indispensable for neural networks applied to computer vision. 32

Convolutional Neural Networks (CNNs) A particular type of neural network which makes use of (not necessarily only) convolutional layers. 27

cropping-resizing The process of cropping an image around the bounding box of the object it features, and then re-sizing the cropped image to a desired resolution. 64

data augmentation The process of performing transformations on data to generate new training examples. 69

data-generating process (in the context of training neural networks) The process that generated the labels that are fed to a neural network. For example, in the case of silhouette extraction, the data-generating process is not the act of recording the images, but the act of extracting the silhouettes from the images. 98

decoder A series of blocks of convolutional layers, each followed by an upsampling layer. The purpose of decoders is to restore the spatial resolution of features that have been condensed to low resolutions by an encoder. 31

Deep Learning Part of the broader field of machine learning, deep learning deals with neural networks that are very deep (i.e. have many layers). 7

depth (of a neural network) In neural networks, computational units are arranged in consecutive 'layers'. The number of layers (or, sometimes, of blocks of layers) that are stacked together defines the depth of the neural network. 113

Dice score A metric used to evaluate semantic segmentation algorithms; strongly related to the Intersection over Union (IoU) metric. 106

dilated convolutional layers (also 'atrous convolutional layers') Convolutional layers in which the filters are 'dilated' by adding zeros between the values of the filters. 32

dilation rate The number of zeros interposed between a dilated convolutional layer's filters. 41

discriminative methods A category of image-based markerless motion capture systems, discriminative methods first extract the silhouette and/or 2D joints of a person from images from one or more camera views, and then learn a direct 2D-to-3D mapping, without using a pre-defined model. 14

encoder A series of blocks of convolutional layers, each followed by a pooling layer. The purpose of encoders is to encode a given input into low-resolution features that are high in semantic information. 31

encoder-decoder A type of neural network architecture in which an encoder first gradually reduces the spatial dimension of the input and captures higher semantic information, followed by a decoder that gradually recovers the spatial information and brings the output back to the original size of the input. 31

epochs The number of times a neural network sees the entire set of data on which it is trained. 122

fine-tuning (in the context of transfer learning) See 'pre-trained' and 'transfer-learning'. 119

foreshortening (in art) The act of drawing or photographing objects or people to make them (or parts of them) look smaller or larger than they are. 59

generative methods A category of image-based markerless motion capture systems, generative methods first extract the silhouette and/or 2D joints of a person from images from

one or more camera views, and then iteratively fits a parametric model to these these data points (which function as constraints). 13

GPU Graphics Processing Unit (i.e. a computer's graphics card). GPUs allow operations to be parallelised much more easily than on a CPU (i.e. a computer's processor). Training neural networks on GPUs (whose software was originally developed for video gaming) was one of the factors most related to the recent success of deep learning, since deep models are too big to be trained effectively on CPUs. 28

heatmaps (in the field of 2D pose detection) Given the 2D coordinates of a keypoint (i.e. joint) in an image, a heatmap is an image in which a 2D Gaussian is centred on the keypoint, and all other pixels are assigned values of zero. The resolution of a heatmap is the same as (or slightly smaller than) the resolution of the original image. 67

in-the-wild images Images recorded outside of controlled laboratory settings (but not necessarily outdoors). 15

IoU Intersection over Union: a metric used to evaluate semantic segmentation algorithms; strongly related to the Dice score metric. 49

Keras with TensorFlow backend TensorFlow is a Python library for developing deep learning models. Keras is a Python library that acts as an interface to TensorFlow, to streamline some of the aspects of model development. Since Keras used to be an interface also for Theano (another Python library for deep learning), it is customary to specify which library was used 'backend' (i.e. behind) Keras's interface for the development of a model. 122

keypoints (in the context of human motion capture) a synonym of 'joints' of the person being recorded. 13

label (verb) The act of assigning a label to a piece of data. (noun) The value that neural networks are expected to output for a given input. For example, if developing a neural network to distinguish cats and dogs from pictures, we could assign a picture of a cat a label of 1 and a picture of a dog a label of 0 (or vice versa). This would teach a neural network to output 1 when it is given a picture of a cat as input, and 0 when the input is a picture of a dog. 24

layer (of a neural network) a number of computational 'nodes' (or neurons) that operate together at a specific depth within a neural network. 31

loss function (of a neural network) The mathematical function that calculates the error between the network's prediction for the current input and the expected output (i.e. the labelled ground truth). 21

module (also 'block'; of a neural network) A series of layers arranged in a particular way, so as to impart to a section of a neural network a desired behaviour. 30

multi-scale (referred to the training of convolutional neural networks) An approach to training neural networks for which the same input is shown to the network at multiple scales, to allow the network to learn to distinguish features at multiple scales. 29

neural network Loosely based on biological neural networks, *artificial* neural networks are computing systems that learn to extract patterns from data. 5

overfitting When a neural network performs better on the training set than on the test set, it is said to 'overfit' (the training set). 30

parametric models Parametric models are generated by collecting thousands of laser-scanned 3D models of people and applying dimension-reduction techniques (like PCA) to them, condensing each model into a number of parameters. A specific model can then be extracted from this family of models by specifying the parameters that correspond to that model. 18

PCK Percent of Correct Keypoints: a metric used to evaluate 2D pose detection algorithms; strongly related to the PCKh metric. 59

PCKh Percent of Correct Keypoints (head): a metric used to evaluate 2D pose detection algorithms; strongly related to the PCK metric. 59

pooling layers Layers of a convolutional neural network that reduce the resolution of their input. For example, max pooling layers are filters of size 2 x 2 that extract only the feature with the highest value from every possible 2 x 2 patch of the input, thereby reducing the resolution of the inputs by four times. 71

pose The position in space of the centres of rotation of a person's joints. 13

pre-trained Many task-specific datasets are too small to allow deep models to learn weights that can generalise well. Therefore, models are often 'pre-trained' on large, general-purpose datasets such as ImageNet, under the assumption that if a model learns features that are good enough to perform well on large, generic datasets, those features will also be beneficial when the model is trained (or 'fine-tuned') on the small, task-specific dataset. 29

receptive field see 'resolution'. 112

residual blocks (in a neural network) Blocks defined by residual layers, which are layers in which the features at a certain depth are copied, forwarded along the network (by a couple of layers), and fused with the features at a given depth. 77

resolution (of a neural network's features; also 'spatial resolution'; also 'receptive field') The portion of the input that is observed by the features of a neural network's layer. xiii, 30

SEB module A component in ExFuse, a state-of-the-art semantic segmentation algorithm. Its function is to introduce more semantic content into the low-level features of ExFuse's encoder. In this thesis, a modified version of the SEB module is proposed. 120

semantic (as in semantic content/information) The features of a neural network are said to have high semantic content if they capture high-level (or 'complex', as in 'made by the sum of multiple simpler parts') information. For example, a layer that learns to recognise a nose from an image is said to have high semantic content. Conversely, features have low semantic content if they capture low-level (or 'basic') information. For example, a layer that learns to detect edges in an image is said to have low semantic content. 31

semantic segmentation The task of performing silhouette extraction and assigning to the extracted silhouette a label corresponding to the class of objects to which it belongs. 24

shape In the context of 3D markerless motion capture, shape refers to the dimensions of the object being reconstructed and to its outer surface. Therefore, the shape of a person could be described by their anthropometric measurements (i.e. height, lengths, and widths of segments, etc). 13

silhouette extraction (also 'silhouette/contour segmentation') The task of outlining the contour of a given object from a given image, and extracting from the picture the area within it (where by 'extracting' we mean obtaining an image whose pixels have a value of one if they belong to the silhouette, and of zero otherwise). 4

skip connections Connections between parts of a network at different depths. 71

SRE module A component of FISHnet, the silhouette extraction algorithm developed during this PhD. It is a modification of DeepLab's ASPP module. 116

test data (also 'test set') Data used to test the performance of a model. These data are not seen by the model during training, and the model is not allowed to modify its parameters when predicting outputs for test data. 58

to pad (referred to an image) To add pixels to one or both of its dimensions so that they match a desired resolution. 105

training The process of feeding to a neural network inputs with corresponding labelled ground truth outputs. The error between the network's predicted output and the ground truth outputs is calculated via the network's loss function. The weights of the neural network's parameters are then modified to minimise the loss function. This process is repeated iteratively until the network's parameters converge to values that minimise the loss function for most of the inputs on which it was trained. 15

training data (also 'training set') Data used to train a neural network (see 'training'). 15

transfer learning The act of pre-training a model on a large, generic dataset and then fine-tuning it on a small, task-specific dataset. 29

Chapter 1

Introduction

1.1 Problem Statement

The automatic extraction of kinematic data (e.g. joint angles and positions) in 3D from videos of underwater swimmers is, with current methods for motion capture[1], impractical at best, and therefore such data are seldom extracted. However, such data, which when analysed over a sequence of frames describe the movement patterns performed by the swimmer, would be valuable for sports biomechanists and practitioners for three reasons. First, such data could improve our theoretical understanding of swimming biomechanics, most of which currently comes from 2D video analysis, which conveys incomplete or difficult to interpret information about how the motion is performed. Being able to analyse in 3D the movements of swimmers could improve our understanding of how differences in the technique (i.e. movement patterns) of different swimmers relate to differences in their performance (i.e. race times); in other words, it would enable researchers to define a more precise relationship between technique and performance[2]. Second, 3D kinematic data could be used to compute kinetic parameters (joint

[1]The task of extracting the 3D joint angles and positions of a person executing a motor task is commonly referred to as *3D motion capture*.

[2]As emphasized by Lees [14], there is a clear—but often overlooked—distinction between technique and performance: *technique* indicates how the movement is performed, while *performance* indicates a measurable outcome of the athletic act. An example of technique, then, would be a baseball pitcher throwing the ball by using a shoulder abduction angle more pronounced than average; an example of performance would be how fast the ball was moving as it left the pitcher's hand.

forces and moments) using complex algorithms, which, without strong prior constraints such as a complete kinematic model as the starting point for the analysis, may be too complex to be computed [15]. And third, it would give coaches a 3D visualisation tool that they could use to evaluate the technique of their athletes after a race or during practice: The coaches, who currently evaluate their athletes on the basis mainly of simple parameters such as stroke count and frequency (thus evaluating performance, rather than technique), would be able to observe the actual movement of their swimmers from different angles and identify flaws in technique. Moreover, the coaches would be able to look at the 3D graphics generated and directly draw conclusions based on their assumptions of what correct technique ought to look like, instead of having to inspect the technical and difficult to parse graphs and tables which are the output of most commercial motion capture systems.

The above considerations can be summarised in a single statement: having access to 3D kinematic data would help improve the performance of elite swimmers — a task which, as the gap between elite swimmers shrinks every year[3], becomes more difficult. By definition, elite athletes are at, or close to, their full genetic potential, and their diet and training regimens are meticulously planned out and executed; so what margin is there to improve performance even further? The theoretical answer is simple: either reduce drag[4], or increase propulsion. The practical answer is not obvious, since little is known about which 3D movement patterns minimize drag or increase propulsion.

Having established that analysing 3D kinematic data is so valuable, it is legitimate to wonder why such data, which have been available for decades for most other sports [16–19], are currently not readily available for swimming. In reality, it is possible to obtain them, by recording videos from synchronised cameras and, for each frame of each camera, clicking on the points that correspond to the joint centres, the 3D coordinates of which are then reconstructed using trigonometry (more on this in Section 2.2). This method, however, is prohibitively time consuming, and as such it is seldom used by researchers, and even less frequently by coaches[5].

[3]For example, in the 50 m freestyle final of the Rio 2016 Olympic Games the time difference between Gold and Bronze was only 0.09 seconds, down from the 0.25 seconds that it was in the London 2012 Olympic Games.

[4]In this thesis, the word *drag* will be used to indicate the concept of *active drag*.

[5]A survey by Mooney et al. [20] revealed that, as of 2016, only 3.9% of the coaches in the USA swimming program used 3D video-based systems daily, while 54.2% of them reported never having used such systems.

Furthermore, as will be shown in Section 2.2, the motion capture systems (e.g. marker-based systems like Vicon, sensor-based systems like Xsens, or depth-based systems like Microsoft's Kinect 1 and 2) used in the analysis of other sports for the automatic extraction of 3D joint angles and positions are not well suited to swimming motion capture.

As will be discussed in Section 2.2, the ideal system for the motion capture of swimmers would be a markerless system that consisted only of a few cameras, from whose recorded videos two types of information would be extracted: the contour (also known as the *silhouette*) of the swimmer, and the locations of the swimmer's joints (referred to as *2D joints*) in camera coordinates. Then, the silhouettes and 2D joints would be fed to an algorithm capable of reconstructing the 3D object (i.e. the swimmer) that generated those silhouettes and 2D joints. For land-based applications, such systems (which can be broadly categorised as 'image-based markerless motion capture systems') have been shown to be accurate enough for biomechanics research [21, 22], but none of them have been successfully applied to swimming[6]. In this type of system, the silhouettes and 2D joints need to be extracted automatically, or the system would be as impractical as the 'traditional' systems for motion capture described in Section 2.2. However, the automatic extraction of silhouettes and 2D joints from images of underwater swimmers is challenging and has not been achieved yet: algorithms that automatically extract such information from images are susceptible to changes in lighting and background, and such changes are frequent and abundant in footage of underwater swimmers. The presence of bubbles adds an additional layer of complexity, as they often occlude parts of the swimmer, making it difficult to identify the boundary between the swimmer and the background, or the locations of joints. As shown in Chapters 5 and 7, even state-of-the-art algorithms cannot accurately extract silhouettes and 2D joints from images of underwater swimmers. Therefore, the first requirement for the development of an efficient tool for the automatic, markerless motion capture of swimmers is the development of algorithms that can accurately extract the silhouette and 2D joints of a swimmer from an image.

[6] Ceseracciu et al. were the only authors who attempted to use such a system for the motion capture of swimmers, but they obtained unsatisfactory results [23].

1.2 Aim and Objectives

The aim of this PhD was to develop two algorithms that would perform well on images of underwater swimmers: one for silhouette extraction, and one for 2D pose detection (i.e. localisation of 2D joints in an image). Due to the complexity of the problem, the analysis focused on two fundamental skills in competitive swimming: underwater[7] butterfly kicking and the breaststroke pull-out. Furthermore, the algorithms were designed to work on one frame at a time, to reduce computational requirements[8]. To reach this aim, the following objectives were set:

1. Review the literature on 3D motion capture systems, to ensure that a markerless system based on silhouettes and 2D joints is the best candidate for swimming motion capture;
2. Develop a dataset (dataset A) of swimmer-silhouette image pairs, where the first image would be an image of an underwater swimmer and the second would be a hand-traced silhouette of the swimmer;
3. Use dataset A to develop and train a silhouette extraction algorithm, which would learn to automatically segment the silhouettes[9] of swimmers from previously unseen images;
4. Create a dataset (dataset B) for 2D pose detection: for each image, a list of the 2D coordinates of each joint would be hand-labeled;
5. Use dataset B to train a pose detection algorithm, which would learn to automatically detect the joints of swimmers from previously unseen images;

1.3 Outcomes of the PhD

This PhD had the following tangible outcomes:

[7] Choosing to analyse movements that happen entirely underwater simplifies the task considerably, since the motion is only in one medium.

[8] Processing HD images using neural networks requires graphics cards with large memories. With the hardware available during this PhD, it was not feasible to train models that learned to predict results for series of frames. The potential benefits of such an approach are described in Section 8.1

[9] See the Glossary entry for 'silhouette extraction'.

1. A literature review that brings together topics from fields that are often treated in isolation: motion capture systems, silhouette extraction, and 2D pose detection. For each of these topics, the review provides not only a summary of the best current practices, but also a critique of the less explored areas of research, potential pitfalls, sources of error, and potential new solutions, contextualising the analysis in light of an application of these techniques to swimming motion capture.

2. The first qualitative analysis on the effect of silhouette accuracy on the accuracy of a visual hull.

3. A dataset (Scylla) of HD images of underwater swimmers with corresponding hand-drawn 2D silhouette annotations. The dataset comprises about 3,000 images taken by cameras at multiple angles, at different times of the day, showcasing 14 different swimmers; the 2D silhouette annotations were manually traced in PhotoShop.

4. A neural network (FISHnet) for accurate silhouette extraction on images of underwater swimmers. FISHnet outperforms state-of-the-art silhouette extraction algorithms on the Scylla dataset.

5. A dataset (Charybdis) of HD images of underwater swimmers, where for each image the 2D joint coordinates were annotated. The dataset comprises about 8,000 images taken by cameras at multiple angles, at different times of the day, showcasing twenty swimmers in two pools; the 2D joint coordinates were manually annotated using a bespoke tool.

6. A neural network (POSEidon) capable of performing accurate 2D pose detection on images of underwater swimmers.

1.4 Publications

Ascenso, G., Yap, M.H., Allen, T.B., Choppin, S.S. and Payton, C., 2020. A review of silhouette extraction algorithms for use within visual hull pipelines. *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, pp.1-22.

Ascenso, G., Yap, M.H., Allen, T.B., Choppin, S.S. and Payton, C., 2020. FISHnet: Learning to Segment the Silhouettes of Swimmers. *IEEE Access*, 8, pp.178311-178321.

1.5 Conference Talks

Ascenso, G., Yap, M.H., Allen, T.B., Choppin, S.S. and Payton, C., 2020. Non-invasive motion capture of swimmers. *12th International Conference on Methods and Techniques in Behavioural Research* 2018, Manchester, UK, June 2018. — **Prize for 'Best Presentation' in the 'Sports' category.**

Chapter 2

Literature Review and Background Theory

2.1 Structure of this Chapter

As this PhD project, which addresses gaps in sports biomechanics using Deep Learning methods, is inherently interdisciplinary, the literature review covers several fields of research. In particular, it discusses the limitations of existing methods for 3D motion capture; the characteristics of 3D markerless motion capture systems, explaining how these systems work and how to adapt them for underwater capture; and the characteristics of the state-of-the-art algorithms for silhouette extraction and 2D pose detection.

This chapter will begin by delineating three criteria by which the validity[1] of a motion capture system can be assessed. In light of these criteria, established methods for motion capture will be briefly reviewed (Section 2.2.2). The purpose of Section 2.2.2 is to provide enough information about these methods and their limitations to justify why they would be poor candidates for swimming motion capture. Due to the extent of the literature on each of these topics, it would be impracticable (and off topic) to review them thoroughly in this chapter.

[1]The term 'validity' here refers to how well suited a motion capture system would be for frequent use for underwater swimming motion capture—as a tool to assist coaches in monitoring the technique improvements of their swimmers, or to enable researchers to better study swimming biomechanics.

Further details on them (as well as rigorous, quantitative justifications for why they were not adopted for this PhD) are provided in Appendices A to D. Similarly, it would be impracticable to provide in this chapter a full description of all the technical deep learning terms that will be used throughout the thesis. Rather, these terms are explained in the Glossary at the beginning of this thesis.

Section 2.3 discusses how image-based markerless motion capture systems meet the three criteria presented in Section 2.2.1, and introduces the two types of image-based systems: discriminative, and generative. As will be discussed, both types function by first extracting from images the silhouette and/or the 2D joints of the person recorded. Consequently, the fields of silhouette extraction and 2D pose detection will be reviewed next, in Sections 2.4 and 2.5. Those two sections will include detailed discussions regarding the datasets on which the algorithms from those fields are trained. Discussing datasets in such detail serves two purposes. First, since in deep learning it is common to tailor the design of an algorithm around the characteristics of the datasets on which it will be tested, understanding the characteristics of the datasets helps understand the differences between algorithms and their strengths and weaknesses. Second, as will be discussed in Section 3.4, existing datasets could not be used to train the algorithms developed during this PhD because none of those datasets feature images of swimmers. Therefore, one of the main challenges (and achievements) of this PhD was the construction of two novel, swimming-specific datasets: one for silhouette extraction, and another for 2D pose detection. To appreciate the reason why new datasets had to be developed, and to understand the rationale behind how they were constructed, it is important to be aware of how the 'standard' datasets were constructed.

2.2 3D Motion Capture in Swimming

2.2.1 Criteria for 3D Swimming Motion Capture

We can define three criteria by which to judge the validity of a motion capture system:

1. **Criterion 1: Speed**. This indicates both that the computation of the parameters of interest should be quick, and that the set up for each recording session should take minimal time and effort. If the purpose of performing 3D kinematic analyses is to inform the coaches on the technique of their athletes and to monitor how said technique changes over time as a response to interventions, it is vital that it be possible to perform these analyses routinely. Ideally, the parameters of interest should be computed in real time[2].

2. **Criterion 2: Non-invasiveness.** The movements being recorded should be representative of the movements performed during competition. To this end, the athletes being recorded must not have their movement restricted or altered. A corollary of this criterion is that the equipment used for motion capture must not increase the drag of the swimmer.

3. **Criterion 3: Accuracy.** To improve the performance of swimmers who are close to the peak of their potential, or to detect subtle differences in the performance of different swimmers, a 3D motion capture system needs to be sufficiently accurate to measure variations of the smallest meaningful magnitude. Accuracy can be defined as the reconstruction error (in mm) of the position of the joint centres in space or as the reconstruction error (in °) of the orientation of their axes of rotation. The threshold for 'acceptable accuracy' varies depending on the joint being analysed (intuitively, joints with lower range of motion will be affected more severely by even small errors) and on the purpose of the analysis. For example, to determine if a cricket bowler's elbow exceeds the 15° of flexion allowed by the International Cricket Council, the accuracy of a motion capture system must be within 1-2° [24], but to estimate the angular velocity of knee extension of a football player kicking a ball, which can be as high as 1715°/s, errors of up to 5° of knee flexion will not influence greatly the results [25]. It seems more prudent, then, to set a threshold not for the orientation errors of the axes of the joint centres, but for their position in space, which is a parameter that affects all types of joints and analyses equally. In this sense, some authors have suggested that the threshold should be set to 5 mm [26–28].

[2] A survey conducted in 2016 revealed that the main feature swimming coaches look for in a 3D motion capture system is the ability to output results within a few hours of a recording session (or, preferably, immediately after) [20].

As discussed briefly in the following section and at length in Appendices A to D, no existing method for motion capture is guaranteed to have error measures below 5 mm. Therefore, whereas for the first two criteria a motion capture system can be said to either violate or respect them, for the third criterion it is more appropriate to speak in relative terms—for example, in terms of how accurate a system is compared to the most accurate system available. Therefore, the ideal motion capture system for studying swimming biomechanics would need to meet the first two criteria, while having higher accuracy than any other systems that meet the first two criteria.

2.2.2 Consideration of Traditional Methods

Traditional methods for motion capture can be classified into four categories: marker-based systems, sensor-based systems, manual digitisation, and depth-camera systems. The first three violate either Criterion 1 (speed) or 2 (non-invasiveness), or both:

- **Marker-based systems**[3] (also called 'optoelectronic stereophotogrammetry systems', or OSSs) like Vicon (Oxford, UK) and Qualysis (Gothenburg, Sweden) are considered to be the gold standard for human motion capture. Indeed, a common procedure to benchmark a motion capture system is to compare it against an OSS [29–31]. OSSs consist of cameras (typically 3-20) and spherical markers (which are automatically tracked by the cameras) of variable dimension (3-25 mm) that are attached to the skin or clothes of a person, on bony landmarks from whose position it is possible to reconstruct—using standard formulas and anthropometric measurements—the position of the joint centres [32]. Even though OSSs are the most accurate systems available, errors of up to 7-20 mm have been reported for them, mainly because markers could be misplaced or could move relative to the bony landmarks they are supposed to represent [33, 34]. For swimming motion capture, cameras would either need to be waterproof or they would need to be placed underwater in special waterproof housings, a task which takes about four hours for an eight-camera system. Furthermore, if several markers need to be attached to a person

[3] More details on these systems are provided in Appendix A.

in a way that will prevent them from detaching when the person is in motion, the setup time may increase substantially. Therefore, marker-based systems might violate Criterion 1, depending on the number of cameras and markers required. Furthermore, spherical markers increase the passive drag of the swimmer by almost 8%[4], violating Criterion 3. An additional downside of marker-based systems is their cost: Oqus (produced by Qualysis), the only available marker-based motion capture system for underwater use, costs over £100,000.

- **Sensor-based systems**[5] involve attaching inertial measurement units (IMUs) to a person[6]. The electronic components of each IMU record (or transmit to a receiver) the position and orientation in space of the IMU, and if enough IMUs are used, joint angles and positions can be calculated. However, like the markers of marker-based systems, IMUs can be misplaced or they can move while the person is in motion, and they increase drag during swimming—by up to 23%[7]. Furthermore, the accuracy of IMUs degrades over time, meaning they might need to be re-calibrated during long sessions [35, 36]. The main advantage of sensor-based systems over marker-based systems is that they only require sensors to be fitted to the athlete, rather than the setup and calibration of cameras. Sensor-based systems, therefore, respect Criterion 1 but violate Criterion 2. The accuracy of sensor-based systems is typically comparable to that of marker-based systems [37].

- **Manual digitisation**[8] consists of a human operator (or a software, which often has to be assisted by a human operator) digitising (i.e. clicking on) the joints of a person in an image; in practice, tape markers attached to the skin of the person are digitised instead, since they are easier to identify than bony landmarks. Manual digitisation is often used to perform analyses of 2D kinematics, but if the cameras are calibrated and at least two cameras see each joint in each frame, the 3D position of the point can be reconstructed via stereophotogrammetry—which is the same working mechanism behind OSSs, with the difference that OSSs require the presence of spherical markers. Therefore, manual digiti-

[4]This value is derived analytically in Appendix A.
[5]More details on these systems are provided in Appendix B.
[6]In some sensor-based systems, like Xsen's MVN Link, the IMUs are embedded in a Lycra suit.
[7]This value is derived analytically in Appendix B.
[8]More details on these systems are provided in Appendix C.

sation is a non-invasive technique, and respects Criterion 2. However, manual digitisation severely violates Criterion 1: to manually digitise one trial of 200 m freestyle swimming recorded by four cameras, it would take an experienced user about 27 hours [38].

- **Depth-camera systems**[9] are the only 'traditional' motion capture systems that do not violate either Criterion 1 or 2. Depth cameras (such as Microsoft's Kinect 1 and 2, or the PMD CamBoard pico flexx) are able to measure the depth—which is to say, the 3D shape—of the objects in their field of view by emitting near-infrared light, recording its reflection off objects, and measuring the depth of the objects based either on the time it took for the light to come back to the camera, or on the amount of distortion with which the light came back to the camera. Depth-camera systems do not require any markers or sensors, and therefore they are inherently non-invasive. Furthermore, they output results in real time (though these results often need to be post-processed to make them more accurate). However, depth cameras are prone to errors: from the accuracy of the depth sensors degrading over time due to overheating [13]; to 'flying pixels', which are erroneous depth estimates that occur close to discontinuities in depth [13, 39, 40]; to distortions due to ambient light [41]; to systematic errors (of up to 4 cm) for objects that are two or more metres from the camera [13]. This last source of error in particular makes depth cameras unsuitable for underwater use: since water has a higher absorbance than air, the light emitted by the camera would be attenuated more than it would in air. This means that a depth camera[10] placed underwater would need to be within two metres of the swimmer, making it impractical.

It can be concluded that the motion capture systems that are routinely employed for the analysis of land-based sports are not well-suited to the motion capture of swimmers. In the following section, an alternative to these systems is discussed: image-based (as opposed to depth-based) automatic markerless motion capture systems. Once again, the discussion will revolve around the three criteria presented at the beginning of this chapter: speed, non-invasiveness,

[9] More details on these systems are provided in Appendix D.

[10] This conclusion applies to the Kinect 1 and 2, which are the two most commonly used depth cameras in biomechanics. Whether they also extend to more modern devices, such as the Azure Kinect, remains unclear, as these devices have not been tested yet for underwater motion capture.

and accuracy.

2.3 Image-based Markerless Motion Capture

Humans do not need markers, inertial sensors, or a near-infrared projector to estimate the depth of the points in an image: they can do it innately, just by looking at the image. Image-based markerless motion capture systems try to accomplish the same feat: only using calibrated video cameras, they are able to reconstruct the 3D coordinates of a given number of keypoints[11] of one or more objects [32]. These methods can be classified based on two criteria [32,42]: the type of reconstruction algorithm they use, which leads to the distinction between discriminative (or regression-based) and generative (or model-based) methods[12]; and the number of camera views they use for each frame, which leads to the distinction between monocular methods (which estimate 3D pose from one image) and multi-view methods (which use multiple images from synchronised cameras). It is useful, at this point, to distinguish two key terms that will be used throughout this thesis: *shape*, and *pose*. In the context of 3D markerless motion capture, *shape* refers to the dimensions of the object being reconstructed and to its outer surface [21]. Therefore, the shape of a person could be described by their anthropometric measurements (i.e. height, length and width of segments, etc.) and silhouette. *Pose*, on the other hand, refers to the position in space of the centres of rotation of the person's joints [21]. The term '3D pose estimation' is therefore equivalent to the terms 'full-body 3D motion capture' and '3D joints'.

Finally, it is important to acknowledge the fully-operational commercial systems for image-based markerless motion capture; for example, Openstage 2 (Organic Motion, NY), DARI (Motion Platform, version 3.2-Denali from Scientific Analytics Inc., Kansas City, KS, USA), and SIMI Shape (SIMI Reality Motion Systems, Unterschleißheim, Germany). Because these systems usually come with many cameras (16 for Openstage 2 [43], 18 for DARI [44]) which

[11] In the context of human motion capture, 'keypoints' is synonym to 'joints'.

[12] The terms 'discriminative' and 'generative' are explained in the next sections. In short: discriminative methods extract some information (i.e. silhouettes and/or 2D joints) from an image and use it to directly estimate the 3D pose of the person; generative methods first extract information from images, and then deform an articulated, generic 3D model to match it [32].

would be impractical to set up in a swimming pool, and because these systems have not been validated sufficiently, they will not be reviewed here.

2.3.1 Discriminative Methods

Discriminative methods learn a direct 2D-to-3D mapping: they take one (or more) image(s) of an object as input, and they apply to this input transformations that allows them to estimate the 3D shape and pose of the object. Because a direct mapping of this kind is hard to learn, discriminative methods first transform the input image into 'intermediate' inputs (meaning they extract some kind of information from the images) and then use the intermediate inputs to estimate the 3D pose of the person. There are two kinds of intermediate inputs that modern discriminative methods typically extract from images: the silhouette (or contour) of the person, which needs to be segmented (or 'separated') from the background [32]; and the 2D pose of the person (i.e. the locations of the joint centres in image coordinates), which is either estimated by the method itself or given as ground truth by the dataset (as is the case with the Human3.6M dataset) [32]. Arnab et al. [45], the authors of a recent discriminative method, have argued that 2D pose is sufficient an input for 3D pose estimation. However, they base this statement on the findings of Kanazawa et al. [46], who two years before had provided evidence that generative methods (described in the next section) do not strictly require the use of silhouettes. For Arnab et al. to extend the same conclusions to discriminative methods may be inappropriate, since the transformations learned by discriminative and generative methods are, as we shall see, quite different. Indeed, Huang et al. [47] found that adding silhouettes to 2D joints as the inputs to their discriminative method increased its accuracy, on average across all joints, by 7.7 mm [47]. Nevertheless, silhouettes are sometimes difficult to obtain. The data of both HumanEva-II and Human3.6M, the two most popular datasets on which discriminative algorithms are trained and tested, were collected in highly controlled laboratories, with white walls for backgrounds [48,49]. Under such ideal conditions, where the background is monochromatic, static, and uniformly lit, extracting the silhouette of an object that contrasts with the background is a trivial task: a picture of the background is captured before the subject is against it, and this reference

image is subtracted from the image with the subject present; the only pixels that will have changed will be the ones that correspond to the person, which is thus segmented[13]. In-the-wild images do not tend to have stable backgrounds, which means their reference frame would often be corrupted by the presence of dynamic background objects (the waving grass in a field, or the turbulent water of a swimming pool); shadows that appear once the person enters the scene; or sporadic or gradual changes in lighting. It is because silhouette extraction is so difficult to perform for in-the-wild images that some authors, like Arnab et al. [45], choose not to use silhouettes as inputs to their discriminative 3D reconstruction methods. Although the Human3.6M dataset provides ground truth silhouettes, those authors argue that if silhouettes are used for training, they should always be labelled for the method to work [45,50]. One of the key contributions of this thesis, discussed in Chapter 5, is the development of a novel silhouette extraction algorithm that should enable future researchers to use silhouettes in their pipelines without being concerned about the difficulty of obtaining them.

Discriminative methods, once considered less accurate than generative ones [32], now achieve state-of-the-art results on both HumanEva-II and Human3.6M. They are also faster than generative methods (as explained in Section 2.3.2) and more robust against occlusions [32]. However, a major downside of discriminative methods is that they require a lot of training data. This problem was one of the main concerns of the authors of the Human3.6M dataset [49], who set out to build the largest, most varied dataset in the field of 3D pose estimation. And indeed, Human3.6M, with its 3.6 million images, is large enough to make it difficult even for the most advanced methods to achieve high performance on the test set. But large as it may be, Human3.6M is still representative more of ideal conditions than of realistic ones, and some authors [1,2,51] wishing to develop methods for in-the-wild 3D reconstruction have elected not to use the training data provided by Human3.6M, and instead to use only its test data, as a validation tool for their algorithms. These authors argue that using Human3.6M for training would mean having access to ground truth 3D pose data, which cannot realistically be obtained

[13]The algorithm just described is known as *basic background subtraction*, and it will be discussed in more detail in Section 2.4.1.1

for in-the-wild images[14]. A paradox seems to have emerged: discriminative methods require large amounts of data to be trained, but if they are to generalise well to in-the-wild images, they cannot use the large datasets that are available[15]. Where is the training data to be found, then, if 3D pose data for in-the-wild images cannot be assumed to be available? A solution to this seeming paradox was proposed by Drover et al. [1]. Their method, schematised in Figure 2.1, feeds 2D pose data to a neural network (which they call a 'generator'[16]) that estimates the depth of the joint centres, thus estimating 3D pose. The estimated 3D pose is then rotated by

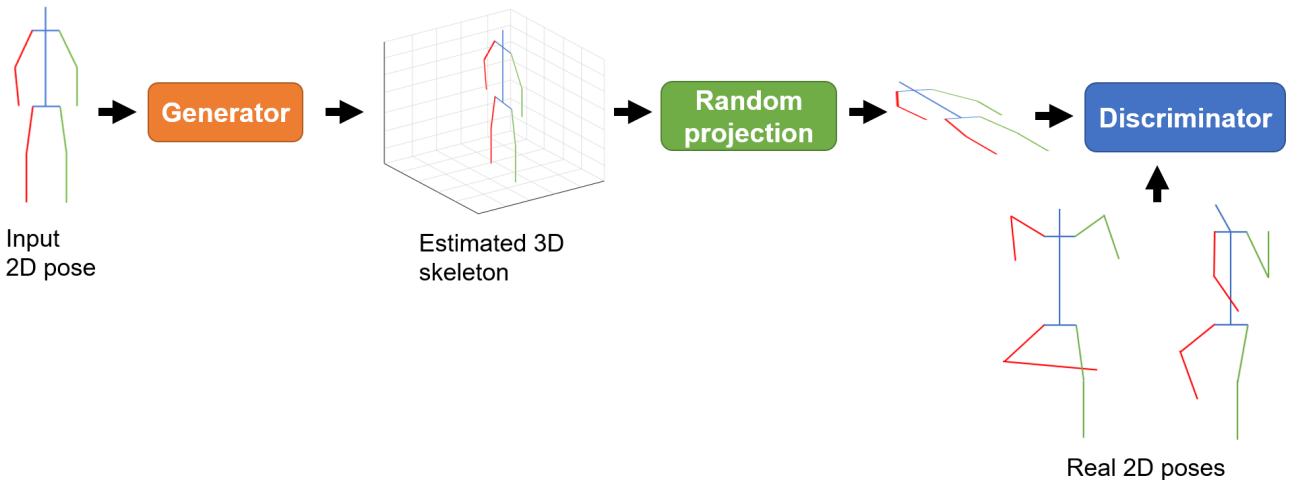


Figure 2.1: Sketch of the approach proposed by Drover et al. [1] (Image adapted from [1])

a random angle, and projected onto a new, random image. What this process achieves is to generate a new image (as if an additional camera had been used) and to project onto it a new set of 2D joint positions. The projected 2D joint positions are then evaluated by a second neural network (which Drover et al. call a 'discriminator'[17]), which has access to a database of real 2D pose data, and which therefore can tell if the projected 2D pose data fall within the distribution of valid 2D poses (i.e. if they look realistic). If the projected 2D pose looks dissimilar to known 2D pose data, it must mean that the 3D object that produced it was not a realistic 3D object.

[14]This argument is similar to the one raised by Arnab et al. [45] regarding the use of silhouettes as inputs for discriminative methods. The difference is that it is always possible (albeit slow to the point of being impractical) to label silhouettes by hand, but to obtain ground-truth 3D data special equipment (which would likely cost over £100,000) is required.

[15]Though some authors have expressed concerns relative to the use of laboratory-built datasets, many others continue using HumanEva-II and Human3.6M for training [52–60], and some even augment them with syntehtic data [61], which is likely to lower the generalisability of the method even further [50].

[16]In deep learning, the term 'generator' usually refers to a component of Generative Adversarial Networks (GANs), which are slightly different from the network developed by Drover et al.: whereas the generator of Drover et al.'s model receives 2D pose as inputs, the generators of GANs usually receive as input random noise, out of which they are asked to extract meaningful information.

[17]The discriminator of Drover et al.'s model functions almost exactly like the generator of a GAN.

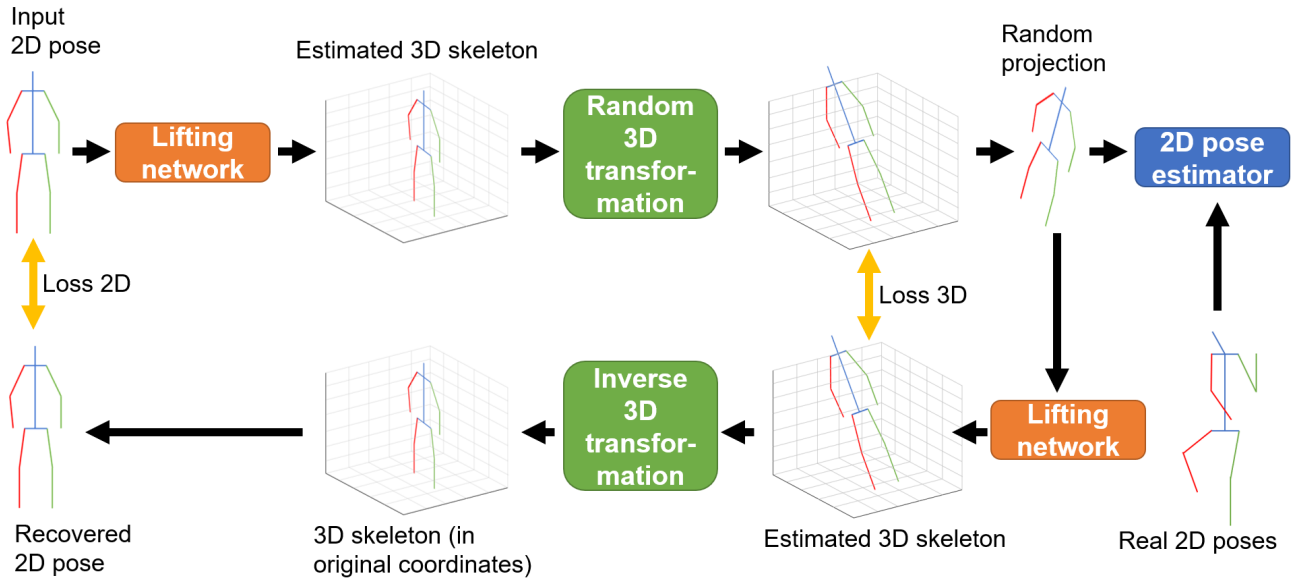


Figure 2.2: Sketch of the approach proposed by Chen et al. [2] (Image adapted from [2])

Finally, this information is used to adjust the first network (the generator) so that at its next iteration it produces more realistic 3D data. This adversarial learning scheme allows Drover et al. to skip altogether the use of 3D ground truths, instead relying entirely on 2D pose data. Chen et al. improved this idea by adding two new steps, schematised in Figure 2.2. Like in Drover et al.'s method, Chen et al. first estimate the depth of the input 2D data using a neural network (which they call the 'lifting network'[18]). The resulting 3D object is then randomly rotated and projected onto a new image frame, and the 2D pose data on the new image frame is fed to a discriminator network (which they call '2D pose estimator'[19]); this produces the first loss term that will be used to update the generator. At the same time, the projected 2D data are fed also to the generator, which thus generates a new 3D object. This new 3D object is compared to the original 3D object from the generated by the generator and then rotated, and their difference constitutes the second loss term for the update of the generator. Finally, the newly generated 3D object is subjected to the inverse of the rotation to which the original 3D object was subjected. If all the components of the algorithm worked perfectly, this should give back the original 3D object. If this was the case, then if this 3D object was projected back onto the original image frame, the two sets of 2D pose data (one from the ground truth, and one

[18]This network is essentially identical to Drover et al.'s generator network; to make it easier to compare the two methods, it will be called 'generator' here.

[19]This network is essentially identical to Drover et al.'s discriminator network; to make it easier to compare the two methods, it will be called 'discriminator' here.

from the final re-projection) should overlap perfectly. If they do not, their difference is used as a third loss term for the generator. Chen et al.'s method, though more refined and advanced, is not as accurate as Drover et al.'s method, which to date is the method with the highest accuracy on Human3.6M (Drover et al.: 34.2 mm error per joint [1]; Chen et al.: 51 mm error per joint [2]). It is unlikely that this is due to Drover et al.'s method being superior in theory. Rather, it is likely that Drover et al.'s method used more advanced components, which, had been arranged as in Chen et al.'s method, would have achieved accuracy even higher than 34.2 mm.

2.3.2 Generative Methods

Unlike discriminative methods, generative methods do not learn a direct 2D-to-3D mapping. Instead, they reconstruct 3D objects by using parametric models. Parametric models [62–65] are collections of thousands of 3D laser-scanned models (which include both shape and pose) of people. Each model is condensed into a few parameters (hence the term 'parametric model') using data reduction techniques like Principal Component Analysis (PCA) [66]. This way, to a certain set of parameters corresponds a unique 3D model[20]: some of the parameters define its shape, some its pose. Whereas discriminative methods learn to predict the exact 3D object from which the input image(s) came, generative methods learn to predict the set of parameters that will generate the model that would best match the input image(s). There are two ways to check if a generated model matches an image. The simplest way [21] is to project the 3D model back onto the image plane and then measure the overlap between its projection and the original silhouette and/or 2D pose[21]. This approach is usually chosen if the object to be reconstructed has a complex shape (like a human being's), and if the method is monocular (i.e. it only considers one camera view at a time) [67, 68]. If the algorithm is multi-view (i.e. multiple cameras are used simultaneously), or if the shape to be reconstructed is simple,

[20] A clarification of terms is warranted here: the term *parametric model* refers to a collection of several thousands of different *3D laser-scanned models*. Therefore, if a set of parameters is specified the corresponding 3D model from within the parametric model can be identified.

[21] This approach is similar to one of the steps in Dover et al.'s pipeline (described in the previous section), with two major differences: 1) the 3D model generated is confined to be one of several thousand available models; 2) the shape of the person is also reconstructed, whereas discriminative models typically only reconstruct pose.

there is another approach to check if the generated model matches the image data [22, 69]. It relies on the computation of what is known as the *visual hull*[22] [70]. The visual hull of an object is an approximation of its volume, and it is obtained by using the object's silhouettes extracted from multiple camera views; the more cameras are used, the better the visual hull will approximate the true volume of the object (but in most real scenarios, the visual hull will over-approximate the volume of the object). If the visual hull of an object can be reconstructed, a parametric model can then be fit to it by iteratively[23] changing the position of the model's joints to maximise the overlap between the model's 3D shape and that of the visual hull [22]. Since the algorithm to reconstruct an object's visual hull is relatively simple, it is the type of image-based method that has been used the most in sports biomechanics—for example, to study the tennis serve [71], [72], to perform gait analysis [69], to analyse the movement pattern of gymnasts [22], and by Ceseracciu et al. to measure arm movements during front crawl swimming [23]. However, for reasons that will be discussed in Chapter 3, Ceseracciu et al. obtained unsatisfactory results.

Fitting a parametric model to intermediate inputs (or 'constraints', i.e. silhouettes and/or 2D pose) is susceptible to local minima. If the model is projected onto the image plane, there could be multiple 3D poses that would be consistent with the same silhouette [73]. It is logical to assume, however, that the probability of convergence on an erroneous local minimum decreases as the number of cameras increases. Likewise, a visual hull reconstructed with more silhouettes is a closer approximation of the object's true volume, and thus there will be fewer local minima for the 3D model to 'find' inside the visual hull; in other words, it is more likely that the 3D model will reach the global minimum.

An aspect that is not discussed in the literature is how the type of parametric model used affects the accuracy of a generative method, in spite of the fact that there is good evidence to show that some parametric models are more accurate than others [63]. Logic dictates that the more 3D models were used to construct a parametric model, the more generalisable it will be. Therefore, the SMPL model [64], which comprises almost 4,000 3D models, should be superior

[22]Since the visual hull constituted a large part of this PhD project, a separate section (Section 3) has been dedicated to it; more details on this method can be found there.

[23]The iteration algorithm most commonly used is the Iterative Closest Point algorithm.

to the SCAPE model [65], which was built using the shapes of 37 people and the poses of a single person positioned in 60 different ways. This assumption seems to be corroborated by the fact that the SCAPE model (published in 2005) has been replaced almost ubiquitously by the SMPL model (published in 2015) in recent studies [46, 47, 55, 74, 75]. More recent and less well-known than SMPL, the Adam model [63] seems to be quite promising, since its authors claim that it is 2.95% more accurate than SMPL. This may be because Adam uses a more sophisticated technique to model hands and faces [63], thus resulting in a better approximation of the shape of the people it models. However, more accurate hands and faces are not a concern for markerless motion capture, and therefore, from the point of view of this specific application, SMPL and Adam may be equally viable.

There is a subset of generative methods that foregoes parametric models entirely. These methods use a laser-scanned 3D model (which only contains shape information) of the person who is the subject of the study, rig the model with a 3D skeleton (which gives pose information) and fit the 3D model to the silhouettes/visual hull [22, 62, 76] extracted from images of the person, recorded by multiple cameras: by deforming the 3D model to match the visual hull/silhouettes, the pose (which is defined by the skeleton rigged to the 3D shape model) is also optimised. These methods create subject-specific models and may therefore be more accurate[24]; furthermore, they do not require any learning, since they do not need to predict the parameters of a parametric model. These methods have two major drawbacks, though. First, before each motion capture session with a new participant a laser-scanned 3D model of that person is required. Second, whereas the 3D models generated by a parametric model learn pose and shape together, if a laser-scanned 3D model is used, a 3D skeleton needs to be designed by hand and then rigged to the 3D model by following hand-designed rules. For example, Corazza et al. [62] fixed a 14-joint skeleton to person-specific laser-scanned models by taking the anthropometric measurements of each participant and using them to locate the joint centres according to the guidelines specified by Andriacchi et al. [77]; the joint centres were then anchored to the surface pixels in their immediate vicinity, so that if the shape of the 3D model were deformed to match a visual hull/silhouette, the joint centres would stay in the same

[24]Though this has not been investigated in the literature, it is reasonable to assume that a person-specific model would have better shape than a generic 3D model learned statistically by a parametric model.

position relative to the body segments of the model. Using a parametric model removes the need to design a 3D skeleton and attach it to the laser-scanned 3D shape, and may therefore be a more convenient choice, especially given the lack of evidence that subject-specific models are more accurate than parametric ones.

Regardless of what type of model (parametric or subject-specific) and what type of intermediate inputs (silhouettes and/or 2D pose) are used, once the overlap between model and intermediate inputs has been measured, the model is re-parameterised to make it consistent with the intermediate inputs; in mathematical terms, an optimisation algorithm attempts to find the configuration of the model that will give a loss function between the model and the intermediate inputs that is below a given threshold. The most commonly used optimisation algorithm for this task is the Iterative Closest Point (ICP) algorithm [22].

This section has led to the following conclusions:

- Intuitively, subject-specific generative methods should be the most accurate. However, there is a lack of evidence for this being the case. Furthermore, the requirement for an easily accessible laser-scanner and for a hand-designed 3D skeleton may make methods that rely on subject-specific models impractical.
- In the absence of a laser-scanner, generative methods that use parametric models have proved to be quite accurate, but only if multiple camera views are available.
- Discriminative methods are currently the most accurate markerless 3D pose reconstruction methods, but most of them are trained on 3D ground truth data that cannot be considered to be accurate or generalisable.
- Regardless of what method is used, it seems that silhouettes are the features that are most commonly extracted from images in order to perform markerless motion capture. However, it is difficult (sometimes prohibitively so) to extract silhouettes from in-the-wild images.
- Using a combination of 2D pose and silhouettes as intermediate inputs for generative models seems to lead to better accuracy.

Given that silhouettes and 2D pose seem to be so important for markerless motion capture and that they are so difficult to extract, the next sections are dedicated to a thorough review of the best silhouette extraction and 2D pose detection algorithms available in the literature.

According to the No Free Lunch (NFL) theorem [78], all machine learning algorithms perform equally poorly when averaged over all possible cost functions. In other words, there does not exist a learning algorithm that can learn to solve *all* possible tasks better than by guessing at random. The NFL theorem implies that learning algorithms must be designed to solve one specific task, and trained on data specific to that task. The more we deviate an algorithm's design or the training data from the domain in which the algorithm is intended to operate, the less accurate we should expect the algorithm to be. For example, an algorithm trained to recognise types of birds from images would not be able to recognise types of elephants unless it was also trained on images of elephants[25].

For the purposes of this PhD, the NFL theorem implies that the algorithms described in Sections 2.4 and 2.5 could not be used off-the-shelf, because none of them were trained on images of swimmers. Figure 2.3 shows an example of this: even DeepLab v3+, the most advanced algorithm for silhouette extraction available to date, performs poorly on images of swimmers. Furthermore, we should expect that algorithms designed for general-purpose silhouette extraction and 2D pose detection would have design choices that make them sub-optimal for use on images of underwater swimmers. For example, most off-the-shelf computer vision algorithms operate on small RGB images (typically 256 x 256 pixels), while the cameras available for this PhD were greyscale cameras with a resolution of 2048 x 900 pixels. Therefore, given that the goal of this PhD was to lay the foundations for an image-based markerless system for underwater swimming motion capture, it was likely easier to develop new algorithms for 2D pose detection and silhouette extraction than it was to adapt existing ones to this specific domain. Therefore, the purpose of reviewing algorithms for silhouette extraction and 2D pose detection in Sections 2.4 and 2.5 is not just to provide a background on these topics, but also to gain insights into how to develop new such algorithms in a way that would be appropriate for the

[25]We can, however, train a model for a given task by giving it examples from that domain, and then re-train the same model for a (slightly) different task by giving it examples from the new domain; this practice is called *transfer-learning* and is a highly active area of research [79–84].

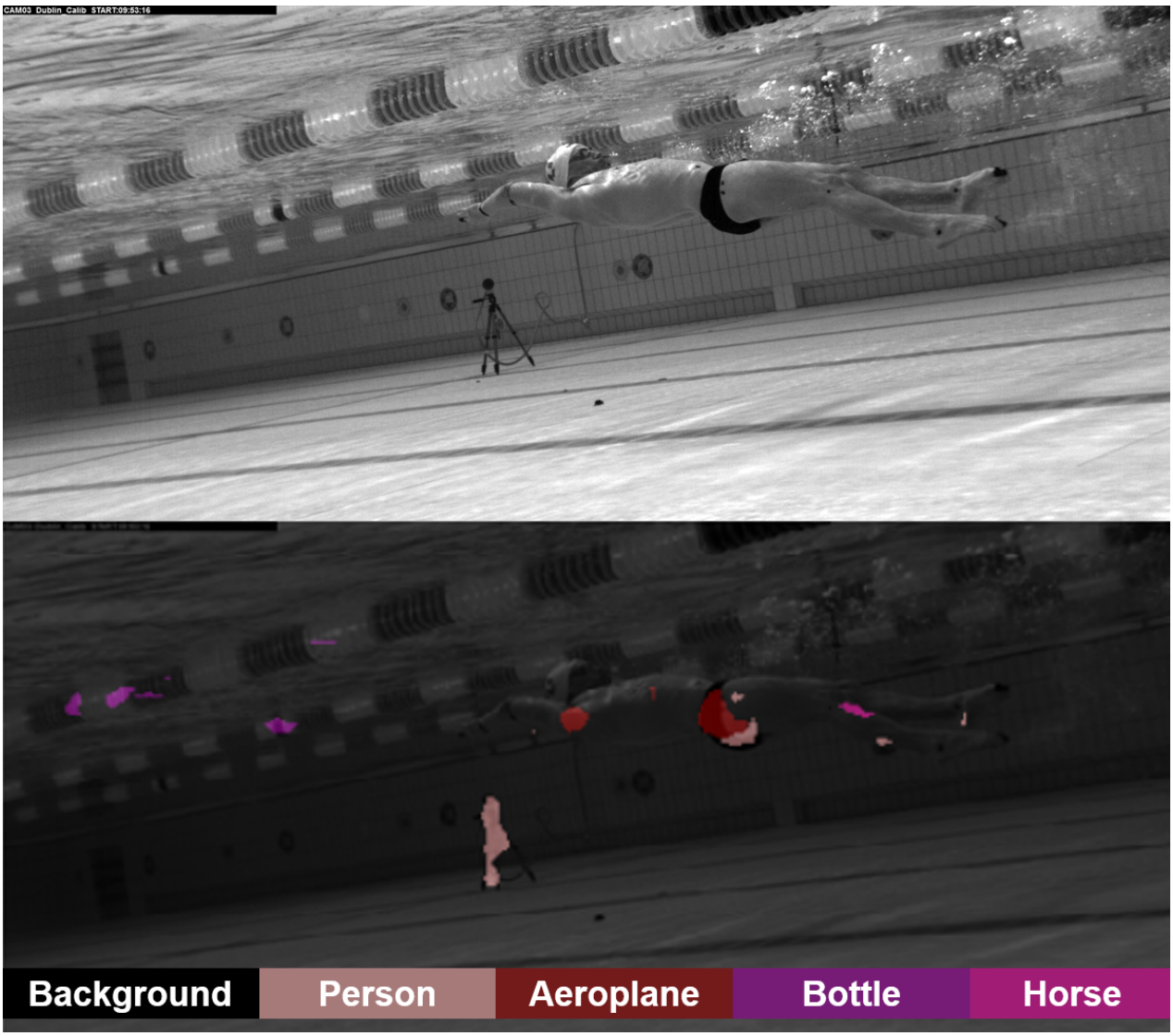


Figure 2.3: DeepLab v3+ is the most sophisticated algorithm for silhouette extraction available to date. However, even it does not perform well on images that come from a domain on which it was not trained—for examples, on images of underwater swimmers. (Image created by the author)

intended application.

2.4 Silhouette Extraction

(*Sections 2.4.1 and 2.4.2 were published in revised form in the following* Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization *paper:* [85].)

The computer vision literature offers hundreds of algorithms that enable the automatic extraction of accurate silhouettes. These algorithms can be grouped into three categories:

background subtraction, semantic segmentation, and multi-view segmentation. Multi-view segmentation algorithms will not be reviewed here, since they are a niche field that has never been used to obtain silhouettes for markerless motion capture systems. However, the state-of-the-art on multi-view segmentation algorithms was reviewed in one of the papers published during this PhD: [85].

Each category of algorithms has specific datasets used to test the algorithms and metrics used to evaluate their performance. In other words, algorithms that belong to different categories are tested on different datasets, with different metrics. This is because each category of algorithms is typically devoted to a specific task for which specialised datasets have been developed and for which the evaluation criteria are task-specific. For example, background subtraction algorithms are often used for surveillance-camera applications [86–88], while semantic segmentation algorithms not only detect the silhouette of the object in the image, but also label it as belonging to one of several possible classes (car, human, cat, dog, etc.) [89]. For the purposes of image-based markerless motion capture, any algorithm that extracts an accurate two-dimensional silhouette from an image would be applicable [70]. Nevertheless, the diversity of datasets and metrics used to evaluate methods that belong to different categories makes it difficult to identify the optimal method for the task at hand. Therefore, the main goal of this section is to clarify which silhouette extraction algorithms would be most suited for use within an image-based markerless motion capture system.

Within each category of silhouette extraction methods there are hundreds of algorithms. A detailed analysis of all of them would be prohibitively long and is therefore beyond the scope of this section. Instead, what this section endeavours to do is to give an outline of each category of algorithms in terms of what their intended application is, what the most popular and best-performing methods within the category are, what datasets they are tested on, and what metrics are used to evaluate their performance. The choice to limit the analysis to a select few algorithms is reinforced by the fact that modern silhouette extraction methods significantly outperform traditional ones under all metrics, as discussed later. Consequently, only the algorithms that achieve the best results on the datasets of their respective category will be discussed here. Review papers whose scope is limited to listing and discussing all the

algorithms, old and new, present within an individual category of algorithms already exist; for more details on a particular category, the reader is invited to refer to the corresponding reviews, which will be highlighted throughout this section.

2.4.1 Background Subtraction

2.4.1.1 Overview

Background subtraction algorithms seek to separate the moving objects (the foreground) present in an image from the static parts of an image (the background) [90,91]. Their main application is within intelligent video surveillance tasks like the automatic tracking of objects within a scene or their recognition (i.e. assigning them a class label), both of which are typically applied to videos recorded from surveillance cameras placed on roads [92], at airports [93], or within buildings [86, 87, 94]. Background subtraction algorithms are developed to meet the specific challenges of this field of computer vision, such as gradual changes in the intensity of the lighting of the scene, insertion in the scene of new background objects (a man carries a bag and then leaves it on the floor: should the bag be treated as background or as foreground?), and dynamic background objects such as waving trees or water rippling in a lake [90, 95]. Furthermore, the algorithms need to be able to model any generic object, the shape and size of which may vary considerably: from an airplane, to a person, to a bike, to a cat. Also, because background subtraction is typically only the first step within a complex computer vision pipeline (like that of an image-based markerless motion capture system), its computation should happen in real time or close to it, meaning that researchers often have to balance a trade-off between speed of execution and accuracy [90]. If a background subtraction algorithm is to be used within a markerless motion capture system, the object it needs to segment is not generic: all humans have similar shapes and sizes, and using this a priori knowledge would allow a simpler model for the background to be used [96]; this, in turn, would speed up the training and inference of the algorithm.

The most basic way to perform background subtraction is to take a reference image in which

the object of interest does not appear and subtract its pixel values from the pixel values of the image from which the silhouette is to be extracted. If the difference in intensity between the pixels of the two images is greater than an arbitrary threshold fixed a priori, the pixel is labelled as foreground; otherwise, it is labelled as background[26]. Under strictly controlled laboratory conditions where the background is completely stable, the assumption of being able to obtain a clean reference frame unobstructed from the object of interest is easily met. In real-life conditions, however, the reference frame may not be obtainable, or it may be corrupted by the presence of dynamic objects in the background (e.g. trees or moving water) or by the presence of shadows that appear once the object of interest is inserted in the scene. Furthermore, if the object and the background have similar colours (a phenomenon known as *colour camouflaging*), basic background detection will fail completely or, at best, cause numerous false negatives [90]. For these reasons, and as shown in Fig. 2.4, basic background subtraction is not adequate for use within markerless motion capture systems, in which variations of the background are, if not expected, at the very least probable.

[26]Throughout this thesis, this technique will be referred to as 'basic background subtraction'.

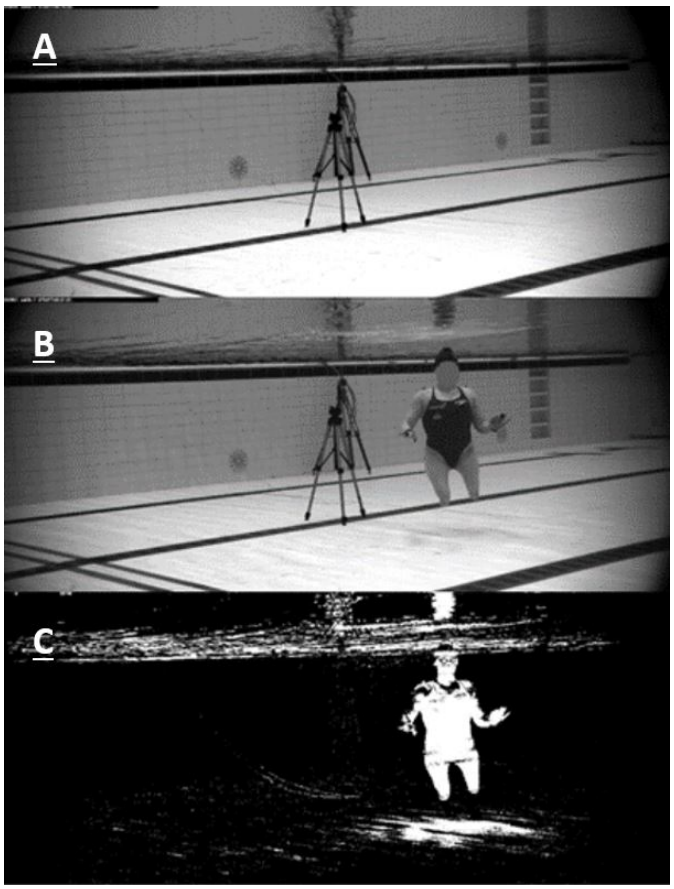


Figure 2.4: A) background initialisation frame; B) foreground object inserted into the scene; C) result of basic background subtraction. The amount of noise present in the output due to the dynamic background and the appearance of a shadow, coupled with false negatives caused by colour camouflaging, would translate into a noisy, inaccurate 2D-to-3D pipeline. (Image created by the author)

2.4.1.2 State of The Art

Traditional background subtraction algorithms define and update complex background models, and then classify pixels by using hand-crafted features and simple equations. Conversely, algorithms based on deep learning do not define nor update a background model, and instead allow the network to learn its own parameters by feeding it several labelled examples of background/foreground pixels. In other words, methods based on deep learning do not compare each pixel to a background model designed by hand: they learn to classify pixels based on examples of previously classified pixels they have seen.

A particular type of deep-learning-based algorithm, Convolutional Neural Networks (CNNs) excel at tasks in computer vision [97–99], in part because they are translation invariant, which

means that objects in new examples are recognised even if they appear in a different location than in past examples [100]; this is an important feature when dealing with dynamic backgrounds and moving objects [100]. Furthermore, the convolution operation can be easily parallelised on a GPU, making it quick to train CNNs [100]. The following sections present the most accurate CNN-based background subtraction algorithms described in the literature. The intent of these sections is not to include every CNN-based background subtraction algorithm in existence, but rather to highlight the top-performing ones that could be used in a state-of-the-art 2D-to-3D pipeline. The accuracy of each algorithm is discussed in detail in Section 2.4.1.3.

Cascade MSCNN The Cascade Multi-Scale Convolutional Neural Network algorithm, developed by Wang et al. [100], is scene-specific, meaning that the network has to be re-trained for each video being analysed. Although this training strategy makes it cumbersome to run Cascade MSCNN on multiple new videos, it enables the algorithm to exploit the high redundancy present in frames that come from the same video, thus reducing the number of training examples required: whereas image classification networks are shown tens of thousands of images during training, Cascade MSCNN only requires about 200. The $N = 200$ training examples are selected at random from the video and they are manually labelled by an experienced user, who labels each pixel as either foreground or background.

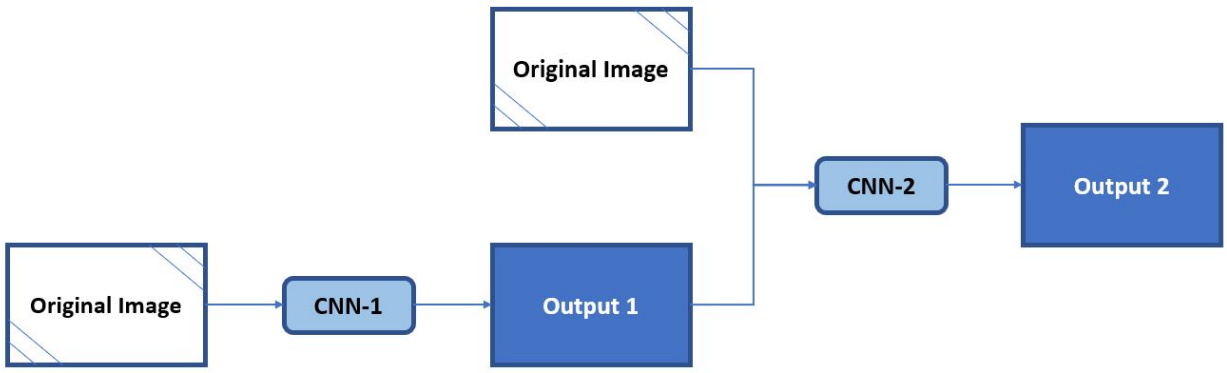


Figure 2.5: Cascaded structure of Cascade MSCNN. (Image created by the author)

CNNs were built specifically as a tool for image classification. To adapt them for background subtraction, the intent of which is to label each pixel within an image, many authors use small

patches (around 30x30 pixels) centred around the pixel to be classified, which are extracted from the image and fed to the CNN for classification. The CNN gives a label to the entire patch, and that label is attributed to the pixel in the centre of the patch [101]. This patch-based approach allows very fine-grained accuracy, but it misses global information that is important to segment large objects [102]. Imagine that a large cat (appearing on the image with size 300x300 pixels, for example) were to be segmented from the background. A 30x30-pixels patch at the centre of the cat would not be enough to tell whether those pixels belonged to the cat or not, because context and a point of reference are absent (all pixels in the patch would have similar colour properties). To address this issue, the Cascade MSCNN algorithm adopts a multi-scale approach: it resizes the original image twice, and these 3 images ($size = 1$, $size = 0.75$, and $size = 0.5$ of the original) are fed to the network separately, thus obtaining 3 separate predictions which are later averaged to produce a single output.

Because CNNs process each pixel independently, they often produce isolated false positives and false negatives [100]. To address this issue, the authors of Cascade MSCNN implemented a cascaded CNN model (illustrated in Fig. 2.5): the first part of the network, CNN-1, makes an initial prediction which is then fed, along with the original input image, to the second part of the network, CNN-2. This cascaded approach ensures that the predicted foreground mask is locally consistent (in other words, the number of isolated false positives and false negatives is reduced) without using post-processing tools like conditional random fields (CRF), which several authors have used in the past [103, 104] but which slow down training due to its computational complexity [100]. Because both CNN-1 and CNN-2 have millions of parameters to train but only 200 images for training, they were pre-trained on a generic dataset [105] for transfer learning purposes. Then, during training, the weights of CNN-1 were fixed, and only the weights of CNN-2 were allowed to learn.

3D-Net 3D-Net [49] was designed to incorporate into the evaluation of an image the information shared with temporally adjacent frames. During training, the network is shown the frame being analysed as well as the nine frames preceding it[27]. The temporal information

[27] Only the ground truth for the frame being analysed is provided to the network during training.

present in this sequence of consecutive frames is progressively encoded into denser and denser 3D convolutional modules (2D for space, 1D for time; see Fig. 2.6), until a prediction is made at the end of the last convolutional module. The redundancy present in temporally adjacent frames is leveraged to intentionally introduce bias into the system, much like Cascade MSCNN sought to exploit the redundancy present in multiple frames that came from the same video. Because 3D-Net is trained on the entire dataset being analysed, the overfitting effect of the bias introduced into the system is alleviated, and the network can generalise to new videos without requiring further training. The concept of incorporating temporal information in the pixel classification task of a neural network is similar to the idea of updating the background in traditional algorithms. In this sense, 3D-net is the only deep-learning algorithm that was explicitly designed to adapt to changes in the background. However, videos recorded with a low frequency represent an issue for 3D-net, because if the latency between consecutive frames is too large, the assumption of temporal contiguity between frames, on which the model is based, does not hold. Noticeably, the authors of 3D-Net do not mention any pre-training or weight initialisation strategies, which are universally recognised as powerful tools to boost the accuracy of CNNs [97, 106–109].

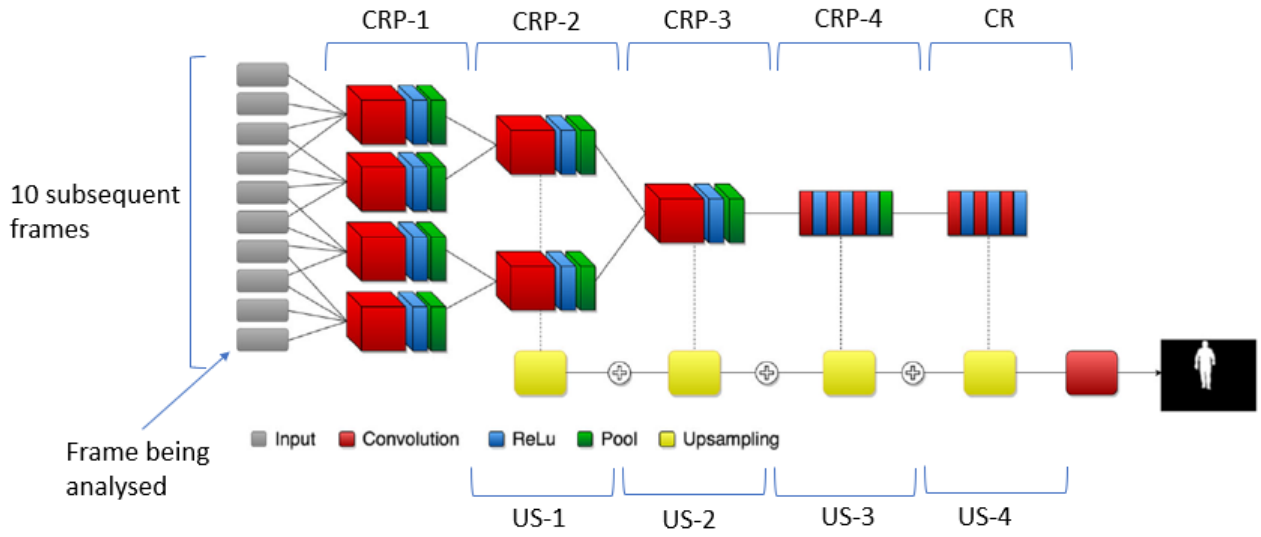


Figure 2.6: Architecture of 3D-Net. Ten consecutive frames are used for the prediction of just the last one of them. CRP-1 to CRP-3 are 3D convolutional modules, whereas CRP-4 and CR and 2D convolutional modules. The kernel size of each upsampling layer (US-1, US-2, US-3, US-4) is different, granting the network multi-scale spatial resolution. (Image adapted from [49])

BScGAN BScGAN (Background Subtraction conditional Generative Adversarial Network) was the first conditional Generative Adversarial Network (cGAN) [110] developed for background subtraction. A cGAN consists of two connected networks, called generator and discriminator. The generator learns to produce an output given an input modified by random noise[28]. The output of the generator is then fed to the discriminator, which has access to a database of real examples, to which it compares the output of the generator to determine how realistic it is. In other words, the challenge of the generator is to produce an output that is as different from random noise and as close to a realistic output as possible; the challenge of the discriminator is to determine if the output of the generator is fake or real. In the case of BScGAN, the real input shown to the generator during training consists of two images: one with the foreground object present, one with the foreground object absent (i.e. true background image). The real output shown to the discriminator is a hand-labelled mask of the object in the real input image. During testing, only the generator part of the network is active; therefore, the processing time of BScGAN is faster than that of Cascade MSCNN, since fewer components are active in BScGAN than are active in Cascade MSCNN. The processing time of BScGAN is further reduced by using entire images for testing instead of dividing them into patches.

Internally, the generator of BScGAN has an encoder-decoder[29] structure where both modules have the same architecture (based on U-net[30] [111]) but reversed layer ordering. The internal architecture of the discriminator of BScGAN is a simple series of four convolutional and four downsampling layers.

FgSegNet In 2018, Lim and Keles published three papers on three different iterations of their background subtraction algorithm, *FgSegNet* [102,112,113]; to date, these 3 algorithms occupy

[28]The distinction between a GAN and a cGAN is that the generator in a GAN is only shown random noise during the first stages of training, whereas the generator of a cGan is trained by using the random noise to modify a real input example.

[29]In an encoder-decoder network, the encoder module gradually reduces the spatial dimension and captures higher semantic information, while the decoder module gradually recovers the spatial information and brings the output back to the original size of the input. This kind of network is explained in more detail in Section 2.4.2.2.

[30]U-Net (see Figure 5.2) was designed as a biomedical image segmentation algorithm, and is an example of an encoder-decoder network (see Glossary). The name U-Net derives from the fact that its decoder has the same number and size of convolutional layers as its encoder, and thus the two blocks are symmetric, forming a U-like shape: first the encoder progressively lowers the resolution of the input, then the decoder progressively restores it.

the top three positions of the leaderboard of the ChangeDetection.net dataset described in Section 2.4.1.3 [114]. Similarly to Cascade MSCNN [100], all iterations of FgSegNet are scene-specific. The strategy used to select the N training frames from a video is identical to that of Cascade MSCNN, but, in addition, the N frames are randomly shuffled to avoid introducing excessive bias into the system by feeding it adjacent frames.

The first iteration of FgSegNet [102] achieves multi-scale spatial resolution by using as input three copies of the same image, re-scaled using Gaussian filtering [115]. The three images are then fed to a triplet of CNNs which share weights to reduce the number of training parameters. The backbone of all three CNNs is taken from the VGG-16 network [116], an object recognition network that can be adapted to background subtraction by replacing its fully connected layers with convolutional layers[31]. The outputs of the triplet of CNNs are then fed to a decoder which makes the final prediction in the same resolution as the input image.

In the second iteration of FgSegNet, called FgSegNet_S [112], the triplet of CNNs is replaced by a single-input encoder-decoder structure. Multi-scale spatial resolution is achieved by placing a Feature Pooling Module (FPM) at the end of the encoder. Within the FPM, parallel dilated convolutional layers[32] with different dilation rates allow the network to incorporate spatial information from multiple scales without re-scaling the image prior to training.

FgSegNet_v2 [113], the third iteration of FgSegNet, maintained the structure of FgSegNet_S but modified the FPM module and the decoder. The updated decoder had a significant impact (+ 1-2%) on accuracy when the number N of training examples was low (25-50), but its impact was negligible ($< 0.01\%$) for $N = 200$, which was the configuration that gave the highest accuracy. The impact of the modified FPM module cannot be evinced directly from the original paper of FgSegNet_v2.

In typical images for background subtraction, the ratio of background pixels to foreground pixels is in the order of 100:1, 1000:1, or even 10000:1 [112,113]. In supervised learning, having an imbalanced number of training examples for different class categories causes problematic

[31] For more details on this, please refer to [89].
[32] Dilated convolutions are described in detail in Section 2.4.2.2.

bias during classification [99, 117]; such bias makes it harder for the network to generalise to new data [102]. FgSegNet deals with the issue of the imbalanced data classes by penalising the loss more if a foreground pixel is classified as a background pixel than the contrary. In other words, the rare class (foreground) is given a larger weight than the dominating class (background) when computing the loss function. The weights for the two classes are derived on a frame-by-frame basis by considering the foreground/background pixel ratio for that specific frame.

2.4.1.3 Datasets

In 2014, a review paper by Bouwmans [90] on methods for background subtraction reported nine 'traditional' datasets (like the Wallflower [95] dataset) and eight 'recent' datasets. Since then, the ChangeDetection 2014 (CDnet2014) [114] and Background Models Challenge 2012 (BMC) [118] datasets, both of which appeared in Bouwmans' 'recent' category, have established themselves as the benchmark background subtraction datasets on which new algorithms are tested.

CDnet2014 contains 53 video sequences (for a total of over 11,000 frames) divided into 11 categories which reflect specific challenging scenarios: Baseline[33], Dynamic Background, Night, Shadows, etc. The videos were obtained using different cameras which had different resolution, frame rate, and compression parameters. Therefore, the resolution of the frames in CDnet2014 ranges from 320 x 240 to 720 x 576 pixels. However, because most of the images are of size 320 x 240, some authors [96, 100] resize all images to this resolution before training their networks. The ground truths made available for testing algorithms on CDnet2014 (see Fig. 2.7 for an example) were segmented manually by human operators under the following guidelines:

- A pixel should be labelled as foreground only if it is not part of the background;
- Foreground objects are people, animals, vehicles, or man-made objects;

[33]The baseline category contains a mixture of mild challenges that belong to the other categories, and therefore is the easiest category for algorithms to analyse.

- A moving object that suddenly stops (i.e. an abandoned bag) should be detected as foreground for a short period before being considered as background;
- Reflections and spotlight halos are not considered as foreground;
- Hard shadows should be manually labelled in order to enable the comparison of algorithms based on their robustness to shadows.

Figure 2.7: On the left: example of an image in CDnet2014. On the right: labelled ground truth for the image on the left.

Another popular dataset for background subtraction, the BMC dataset [118] is comprised of 20 synthetic videos rendered with the SiVIC simulator [119] and of nine real videos acquired by static surveillance cameras. The 20 synthetic videos concern two urban scenes (a roundabout and a street) under different conditions, such as bad weather, artificially added noise, or dynamic background. Though the BMC dataset is still used in the literature today [120], CDnet2014 is by far the most commonly used dataset. In particular, the BScGAN algorithm is the only algorithm of the ones discussed in Section 2.4.1.2 that reported results on the BMC dataset. Therefore, the BMC dataset is not included in Section 2.4.1.5, where methods are compared.

FgSegNet_S and FgSegNet_v2 were also tested on the SBI2015 dataset [121] and on the UCSD dataset [122]. The SBI2015 dataset, which was originally designed as a benchmark for background initialisation algorithms, contains 14 videos with ground truth labels for each frame; of the modern background subtraction algorithms listed in Section 2.4.1.2, only FgSegNet_S, FgSegNet_v2, and Cascade MSCNN were tested on the SBI2015 dataset. The UCSD dataset contains 18 videos (with ground truth labels) which showcase highly dynamic backgrounds, and therefore it constitutes an excellent tool for gauging the effectiveness of a background

subtraction algorithm in a complex environment such as those encountered during the recording of human movement activities. The only modern algorithms that were tested on the UCSD dataset were FgSegNet_S and FgSegNet_v2. Therefore, similarly to BMC, this dataset is not considered in Section 2.4.1.5.

2.4.1.4 Metrics

The CDnet2014 framework includes 7 metrics to measure the accuracy of background subtraction algorithms [114]. These are *Specificity*, *False Positive Rate* (*FPR*), *False Negative Rate* (*FNR*), *Percentage of Wrong Classification* (*PWC*), *Precision*, *Recall*, and *F-measure*:

$$Specificity = \frac{TN}{TN + TP} \tag{2.1}$$

$$FPR = \frac{FP}{FP + TN} \tag{2.2}$$

$$FNR = \frac{FN}{TP + FN} \tag{2.3}$$

$$PWC = \frac{FN + FP}{TP + FN + FP + TN} \tag{2.4}$$

$$Precision = \frac{TP}{TP + FP} \tag{2.5}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.6}$$

$$F - measure = 2\frac{Precision \cdot Recall}{Precision + Recall} \tag{2.7}$$

Recall is the ability of a model to find all the relevant cases (called *True Positives*, or *TP*) within a dataset, while *Precision* is a measure of how many of the cases labelled as relevant actually were relevant. An algorithm is considered accurate if it achieves high *recall* without sacrificing *precision*. The *F-measure* is a weighted harmonic mean of *precision* and *recall*, and as such it allows to express the accuracy of a model with a single parameter, thus enabling an immediate comparison between algorithms. For this reason, the *F-measure* is the most widely reported parameter of accuracy for background segmentation algorithms. However, since it does not incorporate true negatives in its computation, it is sensitive to imbalanced data, and background subtraction ground truths are inherently imbalanced, as mentioned in Section 2.4.1.2. Such an imbalance between the two classes is particularly problematic for deep learning methods, which suffer from bias when trained on heavily imbalanced data. For this reason, Lim and Keles [102], developers of FgSegNet, have used, along with the metrics of CDnet2014, the *Matthews Correlation Coefficient (MCC)*. The *MCC* metric is defined as:

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FN)(TP + FP)(TN + FP)(TN + FN)}} \tag{2.8}$$

where FP, FN, TN and TP denote the number of False Positives, False Negatives, True Negatives, and True Positives, respectively. The use of the *MCC* metric for imbalanced data was proposed by Boughorbel et al. [123]. Unlike the *F-measure*, which assumes values in the interval [0,1], the *MCC* takes values in the interval $[-1, 1]$, with 1 = complete agreement, -1 = complete disagreement, and 0 = no correlation between the prediction and the ground truth. Although the use of the *MCC* in background subtraction has been limited to only a few studies [102,112], it is likely that with the growing popularity of deep learning methods it will also gain popularity.

Vacavant et al. [118], authors of the BMC dataset, advocate the use of the *Peak Signal to Noise Ratio (PSNR)* metric, defined as:

$$PSNR = \frac{1}{n} \sum_{1-i}^{n} \left(10\, log_{10} \frac{m}{\sum_{j=1}^{m} ||S_i(j) - G_i(j)||^2} \right) \tag{2.9}$$

where $S_i(j)$ and $G_i(j)$ are the j^{th} pixels of image i (of size m) in the sequences S and G, respectively. Vacavant et al. also propose the use of what they call 'application quality metrics': the *Structural Similarity (SSIM)* [124] and *D-score* [125] metrics. The *D-score* evaluates the localisation of the errors (i.e. where on the image the false positives are located), whereas the *SSIM* is a perception-based metric that measures the perceived change in structural information in an image [124]. To date, no other researchers have used these metrics for background subtraction.

Wang et al. [100] suggest two metrics (*FPED = False Positive Error Distance*, and *FNED = False Negative Error Distance*) to quantify how far from the nearest foreground object the wrongly classified pixels are. Both of these metrics are conceptually similar to the *D-score* used by Vacant et al. and, similarly to the *D-score*, both of them are unused in the background subtraction literature.

Finally, authors often report the computational speed of their method [96,100,102,112,113], expressed in terms of training time or *frames per second (FPS)*. This metric is fundamental in the field of intelligent video surveillance, where real-time computation is often necessary. Therefore, in this field it is sometimes necessary to sacrifice accuracy for speed of execution.

2.4.1.5 Evaluation of Methods

Algorithm	Average F-Measure	Average Recall	Average Precision	Average PWC	FPS(320x240)	Average MCC
Cascade MSCNN	0.9209	0.9506	0.8997	0.4052	12.5 (GPU)	0.9274
3D-Net	0.9507	0.9609	0.9499	0.2650	-	-
FgSegNet	0.9770	0.9836	0.9758	0.0559	17.99 (GPU)	0.9863
FgSegNet_S	0.9804	0.9896	0.9751	0.0461	21 (GPU)	-
FgSegNet_v2	0.9847	0.9891	0.9823	0.0402	23 (GPU)	-
BScGAN	0.9339	0.9476	0.9232	0.3281	400 (GPU)/10 (CPU)	-

Table 2.1: Results of Background Subtraction Algorithms Tested on CDNET2014

Table 2.1 compares the accuracy of the algorithms presented in Section 2.4.1.2, in terms of the metrics discussed in Section 2.4.1.4, on the CDnet2014 dataset. Since they are not part of the CDnet2014 challenge, the *FPED*, *FNED*, and *D-score* metrics were omitted from Table 2.1; the *MCC* metric, due to its relevance for deep-learning-based methods, was included, although most authors do not report this metric. The results in Table 2.1 were collected by reviewing

the actual papers and by browsing the online leaderboard for CDnet2014[34]. The CDnet2014 leaderboard was particularly useful for determining the processing speed of the algorithms, which not all authors reported in their papers.

Table 2.1 shows that FgSegNet_v2 is the most accurate method under all metrics considered, except for the *MCC* which was not reported by the authors and is not one of the metrics available on the CDnet2014 website. However, the processing speed of FgSegNet_v2 is far slower than that of BScGAN, which is orders of magnitude faster than any other method reported in Table 2.1.

2.4.2 Semantic Segmentation

2.4.2.1 Overview

Garcia-Garcia et al. [89] defined semantic segmentation as the task of 'assigning each pixel in an image to an object class'. A more precise definition is given by Thoma [126], who defines semantic segmentation as 'the task of clustering together parts of images which belong to the same object class'. Zhu et al. [127] give yet another definition, arguing that semantic segmentation is the task of 'dividing a natural image into some non-overlapped meaningful regions'. Using a clear vocabulary is essential in order to avoid confusion of terms. For instance, the meaning of the terms 'object' and 'meaningful' in Zhu et al.'s definition is subjective: if told to segment the 'meaningful' parts of the 'objects' in an image, different people will most likely segment different things [128]. This means that, if the definition of what is to be segmented is not clear, semantic segmentation is an ill-posed problem. To clarify the terminology used in the rest of this section, I propose the following categorisation of semantic segmentation algorithms (following [126]), based on four criteria:

- Operation state (interactive vs passive). Examples of interactive algorithms are the segmentation tools present in Adobe Photoshop and MATLAB [98], which require the user to click on the background to mark it or to provide a coarse initial segmentation which

[34] http://changedetection.net/

the algorithm will then refine [126]. Passive algorithms, on the other hand, do not allow manipulation of the input image;

- Allowed classes (multiple vs binary). Most algorithms fix a priori the number of classes that will be segmented, which can be multiple (cat, house, car, person, etc) or binary (e.g. foreground vs background; cat vs non-cat). In this sense, by taking the definition of semantic segmentation of Garcia-Garcia et al. [89] reported above, background subtraction could be interpreted as a sub-category of semantic segmentation in which only the foreground and background classes exist. However, as pointed out by Zhu et al. [127] and Thoma [126], semantic segmentation clusters together groups of pixels, thus nullifying the issue of isolated false positives/negatives found in background subtraction (see Section 2.4.1.2) and distinguishing the two fields;

- Type of input data (greyscale vs coloured; 2D vs 3D data; including vs excluding depth data);

- Degree of supervision (unsupervised vs weakly-supervised vs fully-supervised). Unsupervised methods do not have access to a label or ground truth during training, whereas fully-supervised algorithms have at their disposal a ground truth for each image to be segmented [127]. Weakly-supervised methods use partial or coarse annotations, and as such often belong to the 'interactive' operation state category.

Recent semantic segmentation research is focused on the multi-class category, because it has more widespread applications [89, 129]. Nevertheless, multi-class algorithms can easily be retrained for a binary classification problem such as the one encountered during the first step of a 2D-to-3D pipeline (i.e. segmenting the foreground, class1, from the background, class0). For this reason, this section of the review will not be restricted to binary classification algorithms. However, because 3D data and depth data cannot be integrated into an image-based markerless motion capture system, algorithms that deal with such data will not be analysed here.

2.4.2.2 State of the Art

Traditional approaches to semantic segmentation relied heavily on domain knowledge to build an algorithm with domain-specific features [126], colour being the feature used most commonly [126, 127]. Although no colour space has been proven to be superior to all others in all contexts [130], RGB is often chosen due to its simplicity and support by programming languages; occasionally, the HIS colour space is chosen due to its property of being invariant to illumination [131–134]. An example of domain-specific features are the 'poselets' introduced by Bourdev and Malik [135] for human pose estimation. Poselets are manually-added extra keypoints such as 'left shoulder' or 'right shoulder' which aid in the task of detecting the poses of people in a scene [136, 137].

As was the case for background subtraction, the field of semantic segmentation was revolutionised by the advent of Deep Learning, and in particular of Fully Convolutional Neural Networks (FCNs, which are a particular kind of CNN): a glance at the most popular semantic segmentation datasets [138, 139] will reveal that almost the entirety of the new research in semantic segmentation adopts networks based on the FNC architecture. For this reason, and because traditional semantic segmentation algorithms are not used in the 2D-to-3D literature like traditional background subtraction algorithms were, this review will omit the analysis of traditional methods for semantic segmentation; for a review of this category of algorithms, please refer to [126, 127].

CNNs applied to semantic segmentation face two challenges: 1) because their structure was originally designed for the task of image classification, they lose feature resolution at each layer (in other words, deeper layers learn highly complex features but forget the global information, which is essential in semantic segmentation because of the requirement for the segmentation to be locally and globally consistent); 2) objects may exist at multiple scales, thus requiring the network to have both large and small fields of view. Both of these challenges have been addressed extensively by the algorithms discussed in this section.

Including in the following analysis every algorithm that uses CNNs for semantic segmenta-

tion would not be pertinent to the objective of this section, which is to describe state-of-the-art silhouette extraction methods that may be used in 2D-to-3D pipelines. As was the case for Section 2.4.1.2, then, this section focuses on the top-performing methods in this category of algorithms. The algorithms discussed in this section were chosen by looking at those with an *Average Precision* (described in Section 2.4.2.4) on the PASCAL VOC 2012 dataset [138] (described in Section 2.4.2.3) of at least 90% in the 'person' category. Although the threshold of 90% was chosen arbitrarily, it allowed me to single out the top ten algorithms for the semantic segmentation of humans.

DeepLab Iterations of DeepLab, an algorithm developed by researchers at Google, have been at the top of semantic segmentation leaderboards for years [89, 129, 138]. One of the main features of DeepLab is its use of dilated convolutions to solve the issue of the loss of feature resolution in the deep layers of CNNs. Dilated convolutions (also known as 'atrous' convolutions, from the French *a trous*, with holes), expand the resolution of the filter in the convolutional layer according to a parameter called *dilation rate*; in practice, this process fills with zeros the empty elements of a dilated filter (see Fig. 2.8). Dilated convolutions allow to control the resolution at which features are computed within the network, and to scale their resolution multiple times without having to learn new parameters like in an encoder-decoder structure. For a two-dimensional signal, for each location i on the output feature map y and a convolution filter w, a dilated convolution is applied over the input feature map x as follows:

$$y[i] = \sum_{k} x[i + r \cdot k] \cdot w[k] \tag{2.10}$$

where the dilation rate d determines the stride with which the input signal is sampled. During training, DeepLab uses patches cropped from the original image. Because of the dilated convolutions present in DeepLab, a large crop size is required to avoid that filter weights with large dilation rates be applied to the zero-padded region (i.e. the empty space within the dilated filter).

DeepLabv3 was introduced in 2017 [140] and has undergone several iterations since. Its

backbone is ResNet-101 [99], a network for image recognition pre-trained on the ImageNet dataset [141]. In DeepLabv3, the last blocks of ResNet are re-purposed using Atrous Spatial Pyramid Pooling (ASPP). First proposed by [142] (and similar to the FPM module of FgSegNet_v2), the ASPP module consists of four cascaded dilated convolutional layers with different dilation rates that allows DeepLab to capture multi-scale information at the level of the features learned by the network [140], instead of at the level of the input features (like in Cascade MSCNN). In two separate experiments, DeepLabv3 was pre-trained on the MS-COCO dataset [110] and on the JFT-300M dataset[35] [114]; both pre-training regimens noticeably improved the performance of DeepLabv3 [3], as we discuss in Section 2.4.2.5.

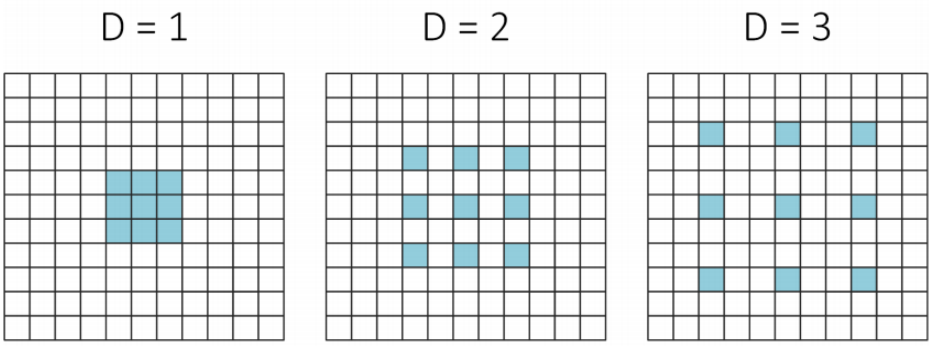


Figure 2.8: Dilated convolutions allow the network to obtain varying spatial resolution by changing the dilation rate (D). The number of parameters to learn (in this figure, 9, one for each coloured square) does not change, since the empty elements within the filter are filled with zeros. (Image adapted from [3])

DeepLabv3+ [143] further improved upon DeepLab3 by implementing an encoder-decoder structure that uses DeepLab3 as the encoder module and a simple series of upsampling layers and convolutional layers as the decoder module. Furthermore, it reduced the computational complexity of the algorithm by adopting depthwise separable convolutions: the standard convolution operation is factorised into a depthwise convolution, which performs a spatial convolution independently for each input channel, and a point-wise convolution, which combines the output from the depthwise convolution step. DeepLabv3+ further improved upon DeepLab3 by adopting the more powerful Xception [144, 145], instead of ResNet-101, as the backbone network. Furthermore, it was pre-trained on ImageNet and JFT-300M in two separate instances.

[35]The version of DeepLab3 pre-trained on the JFT-300M dataset takes the name of 'DeepLab3-JFT', while the version of DeepLab3 pre-trained on the MS-COCO dataset simply takes the name of 'DeepLabv3'.

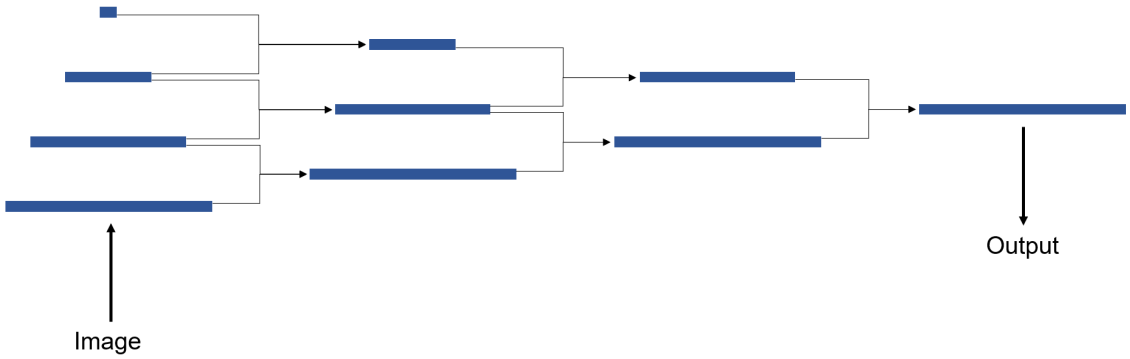


Figure 2.9: Architecture of the MSCI network. On the left, a traditional convolutional neural network structure encodes features into layers of progressively smaller reception fields. Each pair of adjacent layers is connected 'horizontally' using bi-directional connections, which progressively incorporate information from different receptive fields into the final prediction (to the right). (Image adapted from [4])

MSCI In order to capture the multi-scale information present in the data, the MSCI algorithm [4] combines the outputs of pairs of adjacent layers, as shown in Fig. 2.9. Whereas in most multi-scale architectures the information flows in a unidirectional fashion, in MSCI the connections are bi-directional, with long short-term memory (LSTM) chains connecting feature maps of different resolutions. These intertwined connections are present both horizontally and vertically in the network structure (illustrated in Fig. 2.9), granting exceptional integration of the different levels of information present at each layer. Using the structured edge detection toolbox [146], MSCI divides the input image and each subsequent feature map into sets of non-overlapping regions called super-pixels. The neurons that correspond to neighbouring super-pixels are densely connected with bi-directional LSTM connections, thus allowing great local consistency to the segmentation. MSCI uses ResNet-152 [64] as its backbone network and the model is pre-trained on the MS-COCO dataset [139] following [147, 148].

ExFuse Networks that use an encoder-decoder architecture gradually fuse the information from the bottom layers, which is low-level[36] but high-resolution, with the information from the top layers, which is high-level but low-resolution. It is the case, for example, of U-Net [111], which was adopted by several authors as the backbone network for their semantic segmentation algorithm [147, 149–151]. Zhang et al., authors of ExFuse [11], argue that fusing 'pure' low-level

[36]Information is progressively encoded as the layers get deeper. Therefore, the first layers will contain information that is scarcely encoded, and which is consequently defined as low-level. An example of a low-level feature is an edge map of the original image, which requires little encoding to obtain.

and high-level features is inefficient and leads to inaccurate results, and propose to introduce more semantic information in low-level features and more spatial information in high-level features, thus increasing the content overlap between distant layers that will be fused together. They introduce more semantic information into low-level features using three strategies:

- they rearrange the layers of the backbone network (ResNeXt 101 [152]) to be more evenly distributed instead of being clumped up in the deep blocks;
- they use auxiliary supervision at the early stages of the encoder, a practice inspired by Deeply Supervised Learning [153, 154];
- they do not fuse layers in a binary fashion as in U-Net [111]; instead, before being fused with its high-level counterpart, each low-level layer is combined with the ones directly above it using a novel module called 'semantic embedding branch', the purpose of which is to embed more semantic information into low-level features before they are fused with high-level features.

They introduce more spatial information into high-level features using two strategies: 1) they use auxiliary supervision on the first deconvolutional module of the decoder; furthermore, the original deconvolution of the module is replaced with Sub-Pixel Upsample [155], which enlarges the feature map just by reshaping the spatial dimensions; 2) they introduce the 'Densely Adjacent Prediction' mechanism, which enables feature points of the decoder to estimate the semantic information of adjacent points; the final segmentation for each point is obtained by averaging all the associated scores.

DPC Instead of relying on human expertise a neural network, Chen et al. [156], the authors of the Dense Prediction Cell (DPC) method, constructed a space of possible network architectures and used an optimisation tool [157] to select the most optimal architecture within the space. They populated the space with most of the state-of-the-art semantic segmentation algorithms, such as [140, 142, 145, 148]. As the optimisation tool selects random architectures from within

the search space to evaluate them, the selected architecture has to be trained. However, training large networks for semantic segmentation is a time-consuming task, and iterating through a search space of several architectures would be prohibitively expensive in terms of computational time. Therefore, the authors developed a proxy task on which to train the candidate architectures. The objective of a proxy task is to provide the candidate architecture a task that is quick to evaluate and that gives an output that is easily relatable to the large-scale task [156]. To achieve this goal, the authors employed, as a proxy task, a smaller network backbone and cached the feature maps produced by the network backbone on the training set, and then directly build DPC on top of it. The optimisation tool is then run on the architecture search space using the proxy task; after optimisation has ended, the selected architecture is tested on the large-scale task.

2.4.2.3 Datasets

The dataset most referenced in recent semantic segmentation research is undoubtedly the PASCAL VOC 2012 dataset [138]. The PASCAL Visual Object Classes (VOC) project ran challenges evaluating performance on object class recognition algorithms from 2005 to 2012, each year developing a new or modified dataset; starting from 2007, a semantic segmentation challenge was added. The 2012 dataset consists of 28,952 images split into 50% for training-validation (with public ground truths) and 50% for testing (with private ground truths); of these 28,952 images, only 9,993 are labelled for segmentation (see Fig. 2.10 for an example of an image in PASCAL VOC 2012). For each image in the dataset, the bare-minimum label consists of bounding boxes that surround objects that belong to one of the following twenty-one categories: person, bird, cat, cow, dog, horse, sheep, aeroplane, bicycle, boat, bus, car, motorbike, train, bottle, chair, dining table, potted plant, sofa, tv/monitor, and background. All human labellers (the number of which is not shared by PASCAL VOC) were provided with the same guidelines for the segmentation of the ground truths:

- labelled pixels MUST be the object; pixels outside a 5-pixel border area MUST be background. Border pixels can be either;

- pixels which are mixed (e.g. due to transparency, motion blur or the presence of a border) should be considered to belong to the object whose colour contributes most to the mix;
- aim to capture thin structures where possible, within the accuracy constraints. Structures of roughly one-pixel thickness can be ignored e.g. wires, rigging, whiskers;
- if a number of small objects are occluding an object (e.g. cutlery/silverware on a dining table), they can be considered part of that object. The exception is if they are sticking out of the object (e.g. candles) where they should be truncated at the object boundary.

PASCAL VOC also provides a public leaderboard that reports the accuracy of the methods submitted to the website. The size of the images in PASCAL VOC 2012 varies but is generally within 500 x 500 pixels.

Figure 2.10: Example of an image (left) and corresponding label (right) from the PASCAL VOC 2012 dataset. [http://host.robots.ox.ac.uk/pascal/VOC/]

The Microsoft Common Objects in Context (MS-COCO) dataset [139] is also widely referenced in the semantic segmentation community. For the object segmentation task, MS-COCO includes over 200,000 images (all of which are fully annotated) split into 80 categories. However, MS-COCO focuses on instance segmentation[37] rather than on semantic segmentation. In instance segmentation, all objects in the image that belong to different instances of the same object class must be labelled separately (see Fig. 2.11 for an example). Therefore, the methods

[37]This section focuses on methods for silhouette extraction that can be applied to markerless motion capture. Because such a silhouette extraction algorithm would only have to deal with a single object in the image (i.e. the human subject being recorded), instance segmentation algorithms, which focus on the segmentation of multiple objects of the same class, were not considered in this review.

reported in Section 2.4.2.2, which are semantic segmentation algorithms, were never tested on MS-COCO. Nevertheless, most of the algorithms in Section 2.4.2.2 use MS-COCO to pre-train their network, under the assumption that a network pre-trained on a large dataset like MS-COCO will perform better than a randomly pre-trained network [89]. The size of the images in MS-COCO varies but is generally within 640x640 pixels.

Figure 2.11: Example of an image in MS-COCO with superimposed ground truth. Each instance of an object class is labelled separately. For example, each person in this image is segmented using a different colour. [https://cocodataset.org/#home]

Another popular dataset in the semantic segmentation literature is the Cityscapes dataset [158], which focuses on urban scenes. The dataset consists of 5,000 fully-labelled images and 20,000 coarsely-labelled images (see Fig. 2.12 for an example) extracted from videos shot in 50 different cities during daytime. The labels are divided into 30 classes, including 'person' and 'rider' to distinguish pedestrians from people on vehicles. The images in Cityscapes are considerably larger than those in other popular datasets, with a mean resolution of 2040 x 1016 pixels.

Finally, some semantic segmentation algorithms (like MSCI) are tested on scene-labelling datasets like NYUDv2 [159], PASCAL-Context [160], and SUN-RGBD [161]; MS-COCO also has a scene labelling challenge. Unlike the object-centric PASCAL VOC, these datasets focus

on the segmentation of scenery and 'stuff' like grass, sky, and wall. Because this application does not match the one on which this review focuses, these datasets will not be discussed here.

Figure 2.12: Top: example of a finely-labelled image in Cityscapes. Bottom: examples of a coarsely-labelled image in Cityscapes. [https://www.cityscapes-dataset.com/]

2.4.2.4 Metrics

The concepts of *recall* and *precision* were introduced in Section 2.4.1.4. In background subtraction, these metrics are often combined into a single metric, the *F-measure*; in semantic segmentation, they are used to calculate a metric called *average precision* (AP). Let us assume that a segmentation model has a confidence threshold T on its predictions: the model gives a certain label to a pixel if its confidence in the label exceeds T. To this model score threshold correspond a value of *recall* and a value of *precision*. The *Average Precision* summarises the shape of the *precision-recall* curve obtained by varying the threshold of the model so that eleven

equally-spaced *recall* levels are obtained ($r = [0, 0.1, 0.2, ..., 1]$). In other words, the AP is the mean *precision* at a set of eleven equally spaced *recall* levels:

$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, ..., 1\}} p_{interpolated}(r) \tag{2.11}$$

The *precision* at each *recall* level r is interpolated by taking the maximum *precision* measured for a method for which the corresponding *recall* exceeds r:

$$p_{interpolated}(r) = \max_{\widetilde{r}:\widetilde{r} \geq r} p(\widetilde{r}) \tag{2.12}$$

where $p(\widetilde{r})$ is the measured *precision* at *recall* $\widetilde{r}$. This metric penalises methods which detect only a fraction of examples with high *precision*, since it forces the algorithm to have *precision* at all levels of *recall* [109]. But how is the threshold T set, and what does it represent? In the case of PASCAL VOC 2012, T corresponds to an *Intersection over Union* (IoU) value greater than 0.5 [138]. The IoU metric measures how well the ground truth object overlaps the object predicted by the model:

$$IoU = \frac{Area\ of\ intersection}{Area\ of\ union} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.13}$$

In the PASCAL VOC 2012 dataset, the metric reported is AP at $IoU > 0.5$; in other words, a prediction is considered positive if $IoU > 0.5$, and this threshold allows to calculate eleven values of *recall* with corresponding values of *precision*, which in turn allow to calculate the AP. In the MS-COCO dataset, the metric used is the *meanAP* (mAP): the AP is averaged over 10 values of IoU, providing a much stronger metric that rewards methods that are better at precise localisation [162].

The Cityscapes dataset does not use AP as its main metric and focuses on IoU instead. However, since the IoU metric is known to be biased toward object instances that cover a large image area [158], Cityscapes also introduces a metric called *instance-level IoU* ($iIoU$).

$iIoU$ is computed exactly as in equation 18, but TP and FN are computed by weighting the contribution of each pixel by the ratio of the class' average instance size over the size of the respective ground truth instance. Because this metric only pertains to cases where multiple instances of the same object class are present in an image (a condition that will never occur in human motion capture scenarios), the $iIoU$ metric is not be considered further in this review.

2.4.2.5 Evaluation of Methods

Table 2.2 compares the accuracy of the algorithms presented in Section 2.4.2.2, in terms of the metrics discussed in Section 2.4.2.4, on the PASCAL VOC 2012 dataset. Results on the MS-COCO datasets are not reported in this section because MS-COCO deals with instance segmentation, which is not pertinent to the focus of this section. Similarly, because the Cityscapes dataset focuses on urban traffic scenes, results of methods tested on this dataset are not reported in this section.

The results reported in Table 2.2 correspond to the AP values of each method at $IoU > 0.5$. The rightmost column reports the AP values for the 'person' category, while the centre column reports the AP values averaged across all twenty categories of PASCAL VOC 2012. Table 2.2 shows how the top ten algorithms for semantic segmentation evaluated on PASCAL VOC 2012 are very close to each other in terms of AP in the 'person' category. Nevertheless, the more recent versions of DeepLab, in particular DeepLabv3+_JFT [143] and DeepLabv3+_AASPP (still unpublished), are the most accurate semantic segmentation methods available to date.

Out of all background subtraction and semantic segmentation algorithms, FgSegNet_v2, DeepLabv3+, MSCI, and ExFuse are the most promising candidates for use within a markerless motion capture pipeline. However, FgSegNet requires to manually label 200 frames for each new videos to be analysed, making it unsuitable for frequent use within a markerless motion capture system.

Table 2.2: Results of Semantic Segmentation Algorithms Tested on PASCAL VOC 2012

	AP (%)	
Algorithms	**mean**	**person**
DeepLabv3	85.7	92.1
DeepLabv3-JFT	86.9	92.3
DeepLabv3+	87.8	92.8
DeepLabv3+_JFT	89.0	93.8
DeepLabv3+_AASPP (unpublished)	88.5	93
MSCI	88.0	92.8
ExFuse	87.9	92.3
DPC	87.9	92.5
SRC-B-MachineLearningLab (unpublished)	88.5	92.9
DFN (unpublished)	86.2	91.7

2.5 2D Pose Detection

2.5.1 Overview

As Bulat et al. [163] remark, two-dimensional articulated human pose estimation (commonly referred to as '2D pose detection' or '2D pose estimation') is a computer vision problem 'of extraordinary difficulty'; as this section will show, when the images on which 2D pose detection is to be done are of swimmers, this 'extraordinary difficulty' is alleviated in some ways and exacerbated in others.

Let us repeat here the definition of 2D pose detection, so that we may understand how it differs from the process of manual digitisation presented in Section 2.2.2. The term '2D pose detection' refers to the process of (manually or automatically) identifying in image space the two-dimensional coordinates of a certain number of joint centres of one or more people. The difference between manual digitisation and 2D pose detection should therefore be clear: when manually digitising an image, a person (or machine) has to locate the position of *markers* that are attached to the skin of the person; conversely, 2D pose detection is concerned with digitising the actual *joint centres*. Given that the markers used for manual digitisation are placed on landmarks close to the joints they are related to, in some cases 2D pose detection and manual digitisation would involve digitising the same points. For example, in a sagittal view of a person, the marker of the left knee, which would be placed on the lateral epicondyle,

would be in line with the knee joint centre and the two points (the marker and the joint centre) would coincide. For any camera angle that is not exactly sagittal, the points digitised by 2D pose detection and manual digitisation are not the same (though they likely will be close to one another).

Compared to manual digitisation, 2D pose detection is harder: whereas in manual digitisation the operator has to identify a marker placed on the surface of a person, in 2D pose detection the operator has to use prior knowledge of human anatomy to, given landmarks that may not be fully visible, infer the position of the joint centre. In other words, manual digitisation is a matter of localising two-dimensional points (the markers) in a two-dimensional plane (the image), whereas 2D pose detection is a matter of inferring the two-dimensional projections of hidden three-dimensional points (the joint centres) given the operator's knowledge of human anatomy. Once the 2D joints have been digitised via 2D pose detection or manual digitisation, they can be used for two purposes: by themselves, as parameters to perform a 2D kinematic analysis; or as inputs for a 2D-to-3D pipeline. For this second application, the joints digitised via 2D pose detection have a clear advantage: they can be used directly as inputs to the 2D-to-3D pipeline. The same is not true of joints digitised using manual digitisation, for which an intermediate step—converting the digitised *marker* coordinates into *joint* coordinates using an ad hoc anatomical model—has to be taken[38].

From the definition of 2D pose estimation the main challenge of this task becomes evident: inferring from two-dimensional cues the projection of a three-dimensional point that is not visible. Though not discussed in the literature, it is intuitive to think that, unlike in silhouette extraction, the difficulty of this task also depends on the angle (α) at which the person is relative to the camera. For an exactly sagittal view ($\alpha = \pm 90°$), 2D pose detection almost coincides with manual digitisation and can be performed accurately by identifying easily-recognisable bony landmarks; the only challenge (which can be more or less significant depending on the pose of the person) is that, intuitively, sagittal views are more prone to occlusions (see Figure 2.13). For an exactly frontal view ($\alpha = 0°$), 2D pose detection is greatly simplified: not only are

[38]The way this intermediate step is performed is at the core of the difference between manual digitisation methods (for which the conversion has to be modelled by the user) and OSSs (for which the conversion is hard-coded into the software).

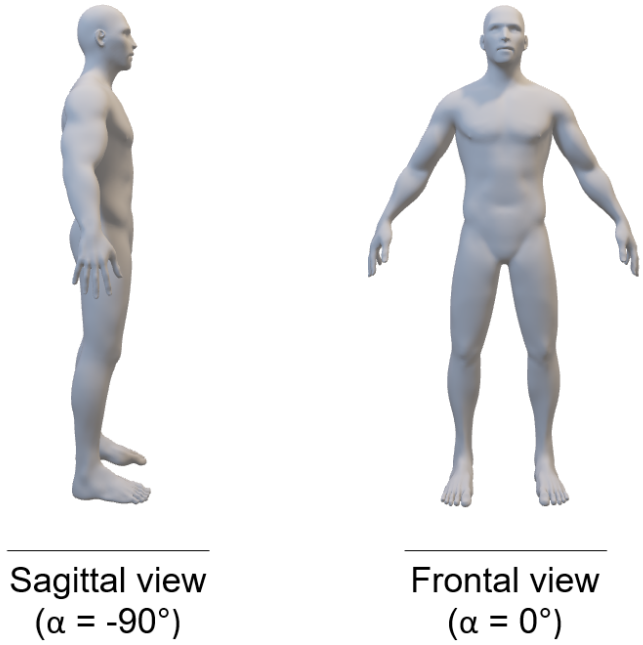


Figure 2.13: The same 3D model viewed from a sagittal and a frontal view. In the sagittal view, most of the joints on the left side of the model are occluded. It is easy to imagine that, had we extended the left arm and leg of the model in the sagittal view, some of the joints would no longer be occluded. This indicates how, in the sagittal view, occlusions are a function of the pose adopted by the person. (Image created by the author)

occlusions rarer in a frontal view, but there also are more cues available to estimate the joint centres' positions. Going back to the example of the knee, identifying its joint centre is even simpler in a frontal view because we have access to twice as many cues: not only the lateral, but also the medial epicondyle; the knee joint centre can then be estimated as the mid point of the imaginary line that connects these two points. Due to perspective, this imaginary line gets progressively more difficult to mentally visualise for $\alpha \to \pm 45°$, especially if only one landmark is visible. The second greatest challenge of 2D pose detection is the fact that, depending on the pose of the person, some joints may be entirely occluded. Occlusions are particularly difficult to deal with for learning algorithms, which to learn have to rely on labels provided by humans. If the labels are highly variable (because occluded joints are difficult to estimate reliably even for humans), the feedback given to the learning model is unreliable and potentially harmful. As we will see, some authors have designed their models to explicitly limit the harm caused by occlusions.

Bulat et al. [163] identify other sources of difficulty for 2D pose detection: the considerable variability of human appearance (i.e differences in body shape, but potentially also the presence and variability of clothing); the large number of feasible human poses; and, potentially, the presence of multiple people in close proximity to one another. This last point does not apply to the intended application of this PhD, since markerless motion capture is performed on a single person at a time. Similarly, by focusing on the underwater fly kick and breaststroke, the space of possible poses is significantly reduced. Furthermore, swimmers wear minimal, tight-fitting clothing. This last aspect makes 2D pose detection easier when done on images of swimmers, for two reasons: the standardised clothing one can expect swimmers to wear reduces the variability of human appearance, meaning that a learning algorithm will require less data to be trained; and, since the clothing swimmers do wear is skin-tight, it makes bony landmarks more visible, thus making it simpler for human operators to label the ground truth joints, providing the learning model with much more reliable feedback. However, dealing with images of swimmers also introduces unique challenges. As discussed in Section 2.4, the unstable background and lighting and the presence bubbles are sources of variability that are hard to model. Furthermore, though this PhD circumscribes the space of all possible human poses to just those adopted during the underwater fly kick and breaststroke, it can be argued that these poses are more prone than average to occlusions—in particular for the underwater fly kick, throughout which the limbs are kept in contact with one another and easily may cause occlusions (see Figure 2.14).

The challenges of 2D pose detection discussed above make manual 2D pose detection a difficult, time consuming task. Furthermore, it arguably is even more subjective a task than manual digitisation, since different labellers may have different levels of knowledge of human anatomy and a different skill in estimating perspective. It is therefore impractical, in the optics of developing a markerless motion capture system, to perform 2D pose detection manually as an intermediary step: it would be so time consuming as to defeat the purpose. Section 2.5.2 will describe the datasets (and metrics) on which 2D pose detection algorithms are benchmarked, as this will help understand the design choices of some of the algorithms that will be discussed. Next, Section 2.5.2 will give a concise overview of the characteristics of the most effective

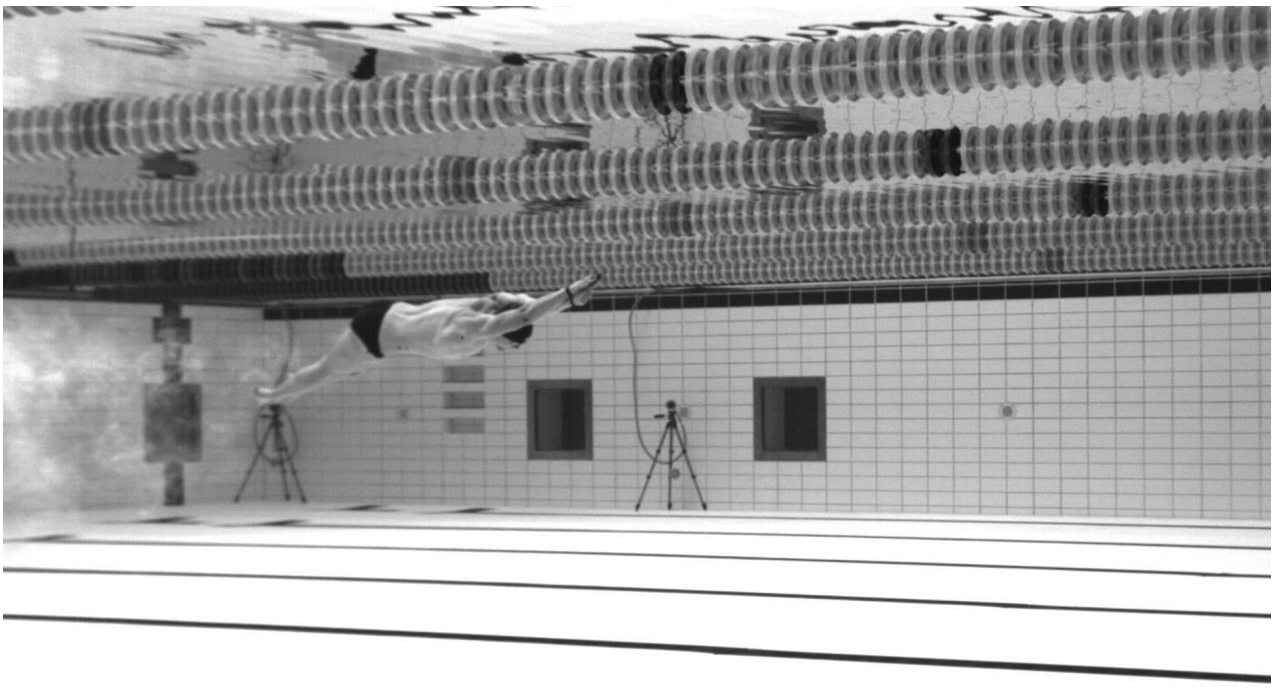


Figure 2.14: Example of how the poses assumed by swimmers may be prone to occlusions. As we said, occlusions are more common in views that are close to sagittal, such as the one shown here. This is especially true if opposing limbs (e.g. left and right leg) are kept in line with each other sagittally, which is the prevalent pose of underwater swimming. (Image created by the author)

algorithms. As in Section 2.4, only algorithms that currently define the state of the art will be discussed[39].

2.5.2 Datasets and Metrics

There are two datasets on which new models for 2D pose detection are usually benchmarked: the MPII Human Pose dataset [165], and the Leeds Sports Pose (LSP) dataset [166][40]. The MPII dataset contains 24,984 images (17,770 used for training, 7,214 for testing) of over 40,000 total people; the resolution of the images ranges between 480 x 272 and 1080 x 1920 pixels. Each image was extracted from a YouTube video and assigned an 'activity label'—such as 'walking', 'pushing car', and the oddly specific 'carpentry, outside house, installing rain gutters'—out of 410 possible categories. Each image comes with a series of annotations for each person in the

[39]For a more thorough review that describes how the field of 2D pose detection has evolved over the past couple of decades, please refer to [164].

[40]Older datasets that are no longer used by researchers (like the Buffy Stickmen [167] and FLIC [168] datasets) and datasets that are only mentioned by a small minority of authors (like the Keypoints section of the MS COCO dataset [139]) will not be presented here.

image:

- the activity label;
- the x and y coordinates of the following 16 joints: left and right ankles, knees, hips, wrists, elbows, and shoulders; pelvis; thorax; upper neck; head top;
- for each joint, a Boolean variable *is_visible*, which is 0 if the joint is in the image but not visible and 1 if it is;
- the coordinates of a bounding box that tightly encloses the head of the person[41];
- a bounding box that 'sufficiently separates' (a term that is not explained nor quantified by the authors of the datasets) individuals.

The images were labelled using Amazon Mechanical Turk (AMT; http://www.mturk.com), a service through which the annotation of large datasets can be outsourced to a collection of separate individuals (who can be screened before the outsourcing). The annotation tool that was given to the people recruited through AMT to label the images in MPII was an extension of the one described in [169], in which the user is simply instructed on how to use the tool, but not *where* to label. In other words, there is no guarantee that the labellers were familiar enough with the anatomy of the human body to accurately label the joints. This potential lack of accuracy pales in comparison to the lack of accuracy caused by the fact that people in almost all of the images are fully clothed. Though the presence of clothes greatly increases the variability of the dataset, it also inevitably introduces errors during the labelling process: an image like the one in Figure 2.15, which is one of the images in MPII, cannot be labelled with the level of accuracy required for algorithms that deal with human motion capture, and therefore is a poor example on which to train a learning algorithm.

[41] The use of this bounding box will become clear when we will discuss the metric used to evaluate algorithms on MPII.

Figure 2.15: Example of an image in the MPII dataset. [http://human-pose.mpi-inf.mpg.de/#download]

The main strength of the the MPII dataset is the high number of images it contains, which allows it to represent many different poses. The Leeds Sports Pose (LSP) dataset has substantially fewer imagees—2,000 in total, split into 1,000 for training and 1,000 for testing (resolution between 525 x 254 and 1024 x 1024 pixels). Such a number of training examples is likely too low for a learning algorithm to reach an acceptable level of generalisability. The only advantage of the LSP dataset over MPII is that all the people in the images of LSP wear sports clothing, which for the most part fits more tightly than average, or at least reveals some of the joints (see Figure 2.16). This means that the labelling of the images in LSP—though not described by the authors—was likely more accurate, as the joints are easier to see.

The joints labelled in the LSP dataset are the left and right ankles, knees, hips, wrists, elbows, and shoulders; the neck; and the top of the head, for a total of 14 joints. Unlike in MPII, the images in LSP do not come with any bounding boxes. This is relevant, because the presence of a bounding box may help a learning algorithm: if the images are cropped around the bounding box before being fed to the algorithm (as is often done [6, 7, 170–173]), the algorithm

Figure 2.16: Example of an image in the LSP dataset. [https://sam.johnson.io/research/lsp.html]

is allowed to ignore a substantial part of the background, thus artificially impeding it from potentially making mistakes by labelling those background pixels as joints. This being said, even though the LSP dataset does not provide bounding boxes, researchers are still free to—as a pre-processing step—use an algorithm to draw their own bounding box and then crop the image around it; the authors of the LSP dataset themselves adopted this strategy when testing their own algorithm on their dataset [166].

The MPII and LSP datasets use two different metrics[42]. The LSP dataset uses the Percentage of Correct Parts (PCP) metric. To calculate this metric, first the joint coordinates must be converted into segment lengths using the formula [166]:

$$Segment\ length = (|x_{proximal\ joint} - x_{distal\ joint}|,\ |y_{proximal\ joint} - y_{distal\ joint}|) \tag{2.14}$$

[42]In computer vision it is standard procedure to make public only the training data of a given dataset, but not the test data. If authors want to validate their algorithms on the dataset, they then need to submit their algorithm to the server on which the test data is hosted, which is where the evaluation is performed. Therefore, the authors cannot freely choose with which metric to validate their algorithms: the metric used will be the one that the creators of the dataset chose and implemented on the evaluation server.

For example, to calculate the 'Left Lower Leg' segment, the proximal joint would be the left knee and the distal joint would be the left ankle. The PCP metric then calculates the percentage of body segments (across the entire test set) that fall within 50% of the length of the ground truth segments. This metric has the advantage that the accuracy of any given joint is not evaluated in a void, but in relation to a joint that is anatomically related to it; it makes sense to evaluate how accurate the localisation of the left knee is with relation to the localisation of the left ankle. However, as Yang et al. [174] point out, the PCP metric is sensitive to the amount of foreshortening of a limb[43], and so can be too loose a measure in some cases and too strict a measure in others. For example, in the bottom image of Figure 2.17, a mistake in the length of the right arm would weigh less than a mistake on the left arm: given that the right arm appears to be longer and that the tolerance to calculate the PCP metric is 50% of the length of the segment, the right arm would have more tolerance than the left. Also, the PCP metric does not penalise false positives. This gives an unfair advantage to approaches that predict a large number of candidates, as shown by Yang et al. [174].

The metric used in the MPII dataset is a modified version of the Percentage of Correct Keypoints (PCK) metric. The PCK metric, introduced by Yang et al. [174] to address the shortcomings of the PCP metric, classifies a predicted joint as correct if it falls within $\alpha * max(h; w)$ pixels of the ground truth joint, where h and w are the height and width of the bounding box that encloses the person, and α is a tunable parameter, usually set to 0.1 or 0.2 [174]; for $\alpha = 0.1$, we would write the metric's name as 'PCK@0.1'. The PCK metric has two advantages over the PCP metric: it is scaled to the size of the bounding box, and therefore it has the same meaning regardless of the size at which the person appears in the image; and it eliminates the problem of foreshortening by calculating the percentage of correct *joints*, not segments. The MPII dataset uses a modification of the PCK metric, called PCKh (where the 'h' stands for 'head'). When calculating the PCKh metric, α is set to 0.5 (and the metric is then written as 'PCKh@0.5') and the bounding box considered is not the one that encloses the full person, but the one that encloses the head. In their paper, the authors of the MPII dataset justify the choice of this bounding box by stating: 'We choose to use head size

[43]In art, 'foreshortening' is the act of drawing or photographing objects or people to make them (or parts of them) look smaller or larger than they are (see Figure 2.17).

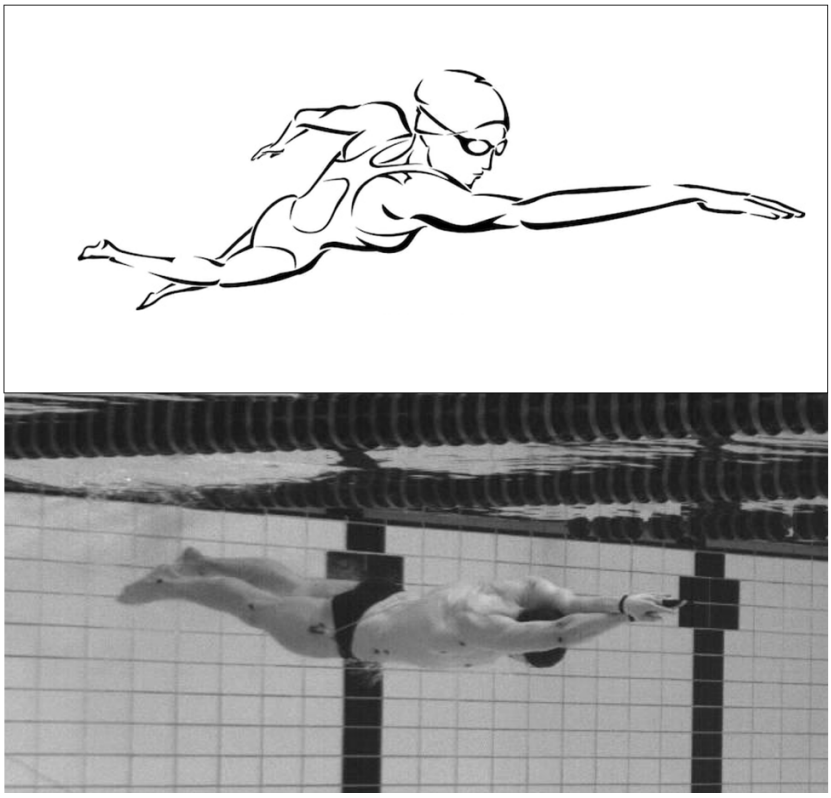


Figure 2.17: Two examples of foreshortening. In the drawing (top image), the right arm of the swimmer is made to appear much longer than the left arm, to convey the angle at which the swimmer is relative to the plane on which she is drawn. The same effect is noticeable—albeit to a lesser extent—in the bottom image. (Image created by the author)

because we would like to make the metric articulation independent' [165]. My interpretation of this statement is that the authors of MPII wanted to choose a bounding box that was not as generic and large as the one that encloses the full person, but that was also not related to a body part that is very prone to foreshortening (like the arms and the legs), given that the shape of the head is similar when viewed from different angles. A more robust (but unmentioned by the authors of MPII) reason for why PCKh@0.5 should be preferred over PCK@0.1 is that it is a stricter metric. To justify this statement, we are going to use the anthropometric measures collected by Plagenhoef et al. [175], who estimated that the head represents $\sim 6\%$ of the total height of a person, the neck $\sim 4.7\%$, and the trunk (from the base of the neck to the pelvis) $\sim 29.5\%$. Consider now the case of an image of a person standing up straight (see Figure 2.18). Since in this case the height of the bounding box is roughly equal to the height of the person, PCK@0.1 is equivalent to considering 10% of the height of the person as the threshold to count the number of correctly classified joints. On the other hand, PCKh@0.5 considers 50% of the

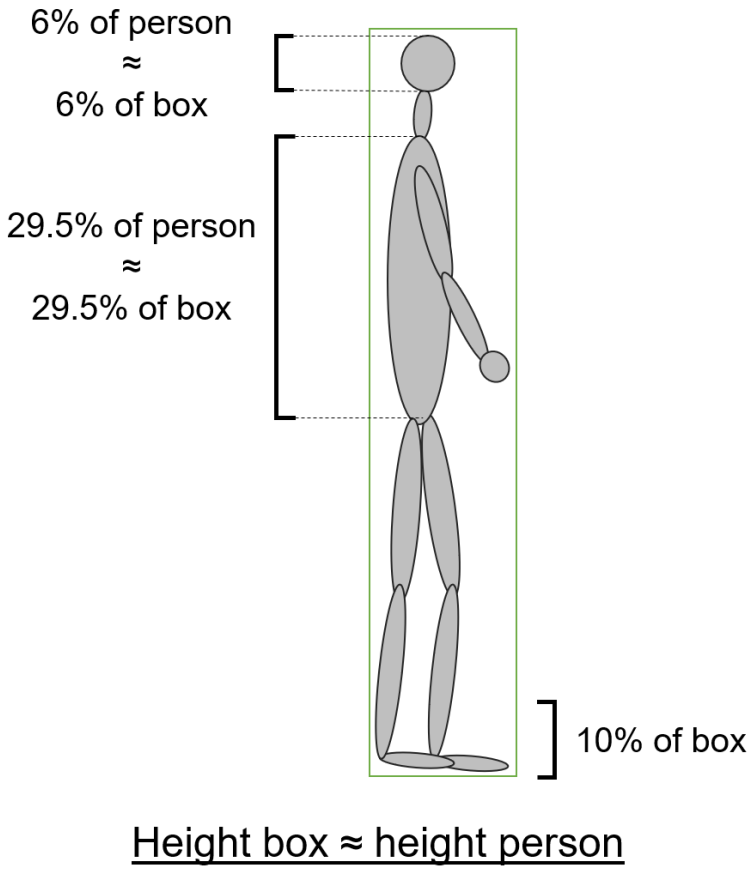


Figure 2.18: In an image of an upright person, the height of the bounding box that tightly surrounds the person is roughly equatable to the height of the person. Therefore, the height of the head represents, in this case, roughly 6% of the height of the bounding box. (Image created by the author)

length of the bounding box of the head; since the head constitutes 6% of the height of the person, PCKh@0.5 is equivalent to considering 3% of the height of the box as the threshold. In other words, in the case of Figure 2.18 PCKh@0.5 is 3.3 times more strict a metric than PCK@0.1. Consider now Figure 2.19. Here, the height of the box is roughly equal to the sum of the lengths of the head, neck, and torso segments. Given that the torso constitutes 29.5% of the height of a person, the head 6%, and the neck 4.7%, we can calculate the height of the head relative to the box as:

$$Head_{\%\ of\ box} = \frac{100}{\frac{trunk_{\%\ of\ person}}{head_{\%\ of\ person}} + \frac{head_{\%\ of\ person}}{head_{\%\ of\ person}} + \frac{neck_{\%\ of\ person}}{head_{\%\ of\ person}}} = \frac{100}{\frac{29.5}{6} + 1 + \frac{4.7}{6}} \simeq 14.9\% \quad (2.15)$$

Therefore, in this case PCKh@0.5 is equivalent to considering 7.47% of the height of the box as the threshold; in this case, PCKh@0.5 is 1.4 times more strict a metric than PCK@0.1. From

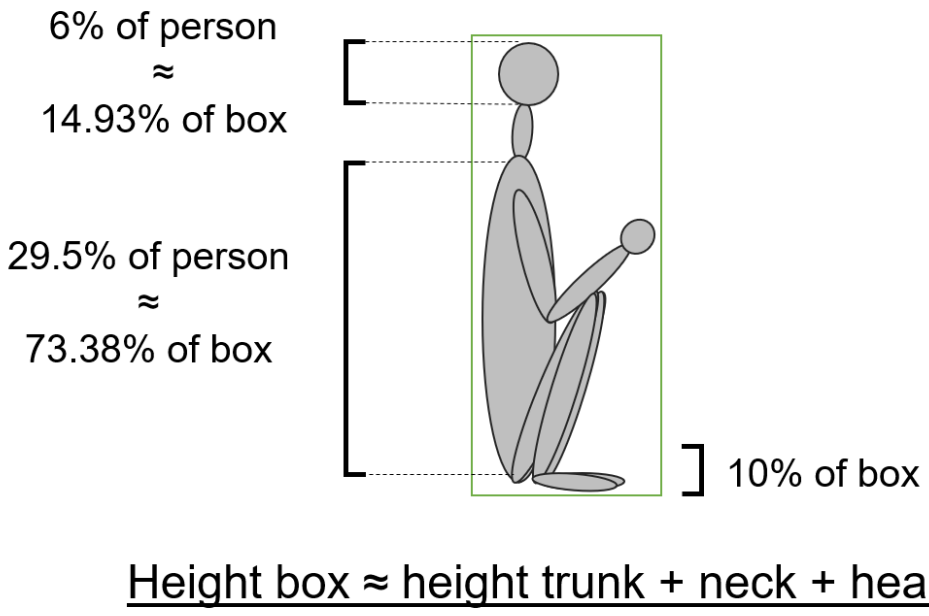


Figure 2.19: In this image, the height of the bounding box coincides not with the height of the person, but with the sum of the length of the trunk, neck, and head segments. In this case, as calculated in Equation 2.15, the head represents roughly 14.9% of the height of the bounding box. (Image created by the author)

these two examples we can infer that the taller the bounding box surrounding the person, the stricter (and therefore better) PCKh@0.5 is than PCK@0.1. Therefore, Figure 2.20 represents the upper bound for the advantage of PCKh@0.5 over PCK@0.1 (since there are no bounding boxes that can be taller than a person standing upright with arms overhead, assuming that the bounding box always tightly surround the person). To quantify this upper bound, we first need to calculate the height of the bounding box (normalised to the height of the person). If we consider that the length of the arm-forearm-hand segment is 38.85% of the height of a person [175], and if we assume that it starts at the same height as the base of the neck, we can estimate the height of the box (normalised to the height of the person) to be:

$$Height_{box} = height_{person} - height_{neck+head} + length_{arm} = 1 - 0.1075 + 0.3885 = 1.28 \quad (2.16)$$

Since we normalised this value to the height of the person, we can now estimate that the height of the head (which is 6% of the height of the person) is $6/1.28 = 4.69\%$ of the height of the bounding box. In this case, then, PCKh@0.5 is equivalent to considering 2.35% of the height of the bounding box. Therefore, PCKh@0.5 is, at most, about four times more strict a metric

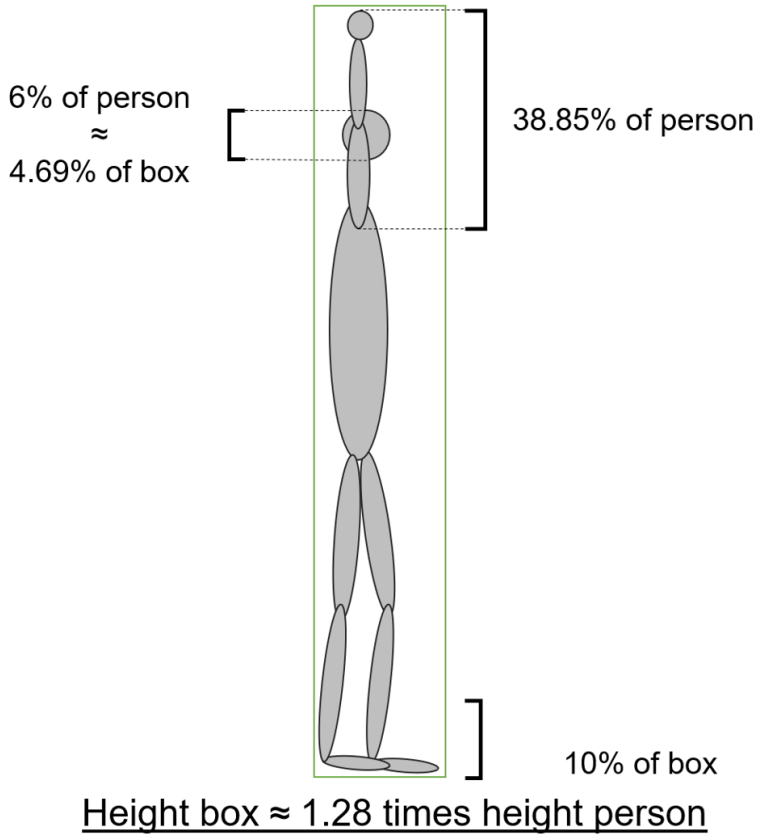


Figure 2.20: In this case, as calculated in Equation 2.16, the box is 1.28 times the height of the person, making the head represent 4.69% of the height of the box. (Image created by the author)

than PCK@0.1.

The lower bound of the advantage of PCKh@0.5 over PCK@0.1 is harder to estimate. For example, if the spine of the stick-man in Figure 2.19 were rounded or inclined, the contribution of the height of the segments to the height of the box would be harder to calculate. Similarly, for images that do not include a full person (such as Figure 2.15), the proportions may change significantly. Nevertheless, we can safely conclude that, at least for images where the entire person is visible, PCKh@0.5 will always be a stricter metric than PCK@0.1—up to about four times stricter, for images in which the person is perfectly upright.

The following sections will describe how the state-of-the-art algorithms for 2D pose detection work. The goal of these sections is not to recount in detail all the advancements that have been made in the field of 2D pose detection over the past few decades; such a survey can already be found here: [164]. Rather, the discussion will focus on three aspects: the type of

input that algorithms use; the type of labels they use and the type of output they produce; and the architectural design choices that seem to be most effective for 2D pose detection. This last point means that the following sections will not discuss every algorithm available for 2D pose detection. The goal of these sections is to disentangle an algorithm's position on a leaderboard (which may be affected by the amount of hardware used or extra training data it had access to) from the effectiveness of its architecture. For example, many papers have been published that make slight changes to the Stacked Hourglass architecture (described in Section 2.5.5.1), achieving slightly better results than it. Although relevant, these slight improvements only indicate that the Stacked Hourglass model can be optimised even better, but they do not offer new, standalone design ideas, and therefore will not be covered in the next sections.

2.5.3 Inputs

The MPII dataset comes with bounding boxes that tightly enclose each person in an image. Most algorithms rely on the presence of said bounding boxes, around which they crop the image. This leads to every cropped image having a different resolution, since most images would have bounding boxes of different sizes. Because deep learning models expect all inputs to have the same dimensions, the cropped images are then resized to the same desired resolution—usually 256 x 256 pixels [5, 6, 163, 170–173, 176, 177]. If the desired resolution's ratio is not the same as, or a multiple/factor of, that of the cropped image, the process of resizing will distort the image (see Figure 2.21).

Most authors employ this 'cropping-resizing' strategy without discussing whether it may be harmful to distort the input images in such a way. Whereas cropping around the bounding box does not alter the spatial characteristics of the subject, resizing the image to a different ratio does. In Figure 2.21, the relationships between the position of the joints—and consequently the lengths and proportions of the limbs—is not the same for the cropped image and for the resized image. Of what consequence this might be, is not clear. However, these images are the examples from which a model is expected to learn the relationship between joints well enough to be able to detect them even in case of occlusions. Therefore, distorting the input images

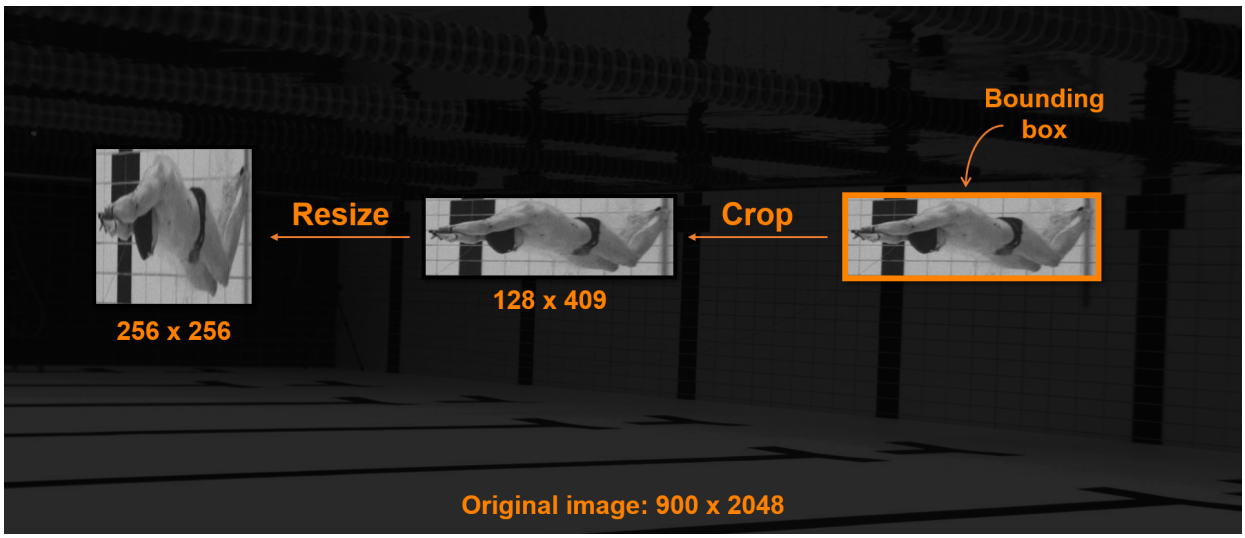


Figure 2.21: Most 2D pose detection algorithms first crop an input image around a given bounding box, then resize the cropped image to fit a desired resolution. Unless the ratio of the desired resolution is a multiple (or a factor) of the bounding box, this process will cause the image to be distorted. (Image created by the author)

might make it harder for the model to learn realistic representations of human pose.

However, if the images are to be cropped around the bounding boxes, resizing them to a common resolution is a mandatory step, as deep learning models need batches of data of uniform shape. The question, therefore, is whether the benefits of cropping around the bounding box outweigh the potential harm of distorting the image by resizing it. The most obvious benefit of the cropping-resizing strategy is that, in most cases, it drastically reduces the size of the inputs to the network. For the images of MPII, which have a mean resolution of 771 x 1330 pixels, cropping-resizing to 256 x 256 would correspond to a 15-fold decrease in size, which is very significant for the purposes of training a deep learning model. For HD images like the one in Figure 2.21, which is of a resolution not uncommon for biomechanics research [178], cropping-resizing to 256 x 256 pixels would correspond to a 28-fold decrease in size. Let us consider the case of an image with resolution 900 x 2048 pixels and a bounding box of resolution 256 x 256—our desired resolution. By cropping around the bounding box, we would be giving our model the 3.56% of the image that actually contains useful information, and discarding the 96.44% that is just background. This means that cropping-resizing not only makes training faster by giving the network smaller inputs; it also makes training more efficient, as a substantial amount of useless information is eliminated. Let us now imagine a second 900 x 2048 pixels

image, with a bounding box of 512 x 512 pixels. By cropping-resizing to 256 x 256 pixels we would eliminate the 85.78% of the image that was useless background, but also 50% of the informative pixels contained in the bounding box. An obvious solution would be to crop-resize such HD images to a resolution higher than 256 x 256—for instance, 512 x 512. Though the computational cost of processing the cropped-resized inputs would double, it is likely that more information would be retained, since it would be less likely that the resolution of the bounding box was much higher than the target resolution of 512 x 512. Ultimately, therefore, the choice of target resolution should be made with two things in mind: the hardware available (the more memory on the GPU(s), the higher the target resolution should be); and the average size of the bounding boxes in the dataset (the larger the bounding boxes, the higher the target resolution should be—ideally, as large as the mean bounding box resolution). However, choosing the target resolution based on the mean resolution of the bounding boxes in the dataset implies that the dataset is static (i.e. the number of images in it does not change over time) and that we can measure the mean of its bounding boxes before choosing the target resolution. This condition would always be met during training and testing, since during these phases the data that our model sees are static and have been labelled beforehand. However, if we want to apply our model to new, unlabelled data, we would not know a priori the average size of the new bounding boxes. Therefore, the best course of action might be as follows:

1. Find (or build) a dataset that is representative of the desired task, both in terms of image resolution and contents of the images. For example, for the purposes of markerless motion capture of swimmers one would need a dataset that features HD images of swimmers underwater;

2. If the dataset does not come with bounding boxes, label them (manually or using an off-the-shelf tool);

3. Calculate the mean resolution of the bounding boxes in the dataset (let us call this variable *mean_box_res*);

4. For images that have bounding box resolutions $>$ *mean_box_res*, resize the image (maintaining the aspect ratio) so that their bounding box resolution becomes $<$ *mean_box_res*);

5. Crop-resize all images using *mean_box_res* as the target resolution;

6. Train the model;

7. When new data are fed to the model for inference, use an off-the-shelf algorithm to identify bounding boxes, then crop-resize all images using the *mean_box_res* that was estimated during training; if enough data were used for training, this value should still be representative of bounding boxes found in new, similar data.

Following these steps would reduce the computational cost of training, while reducing the amount of information lost due to cropping and the amount of distortion due to resizing.

2.5.4 Labels and Outputs

The joint annotations for the images in the MPII and LSP datasets come in the form of $(x_j,\ y_j)$ pairs, where x_j and y_j are the coordinates of joint j in a given image. Most authors convert these annotations into heatmaps: separately for each joint, they create a 2D Gaussian distribution (described by the equation below) centered around the point $(x_j,\ y_j)$:

$$g(x, y) = \frac{1}{2\pi\sigma^2} \exp(-\frac{x^2 + y^2}{2\sigma^2}) \tag{2.17}$$

where σ is the standard deviation (in pixels; this value is usually set to 1 [6, 172]) of the distribution. If N joints were labelled, this would lead to the generation of N heatmaps, one per joint, each with the same resolution as the input image. In each heatmap, the value of each point represents the probability that it is the original label $(x_j,\ y_j)$. By centring the Gaussian on $(x_j,\ y_j)$, the value of the point $(x_j,\ y_j)$ is set to 1, whereas the points around it assume values p $\in$ [0, 1) that decrease according to Equation 2.17; specifically, points that are 2σ pixels away from $(x_j,\ y_j)$ will have a value of approximately 0.05. These N heatmaps can then be put together in the same image for better visualisation, as in Figure 2.22. A neural network trained with heatmaps as labels will learn to produce heatmaps as its output. These outputs are then

converted back into the format (x_j, y_j) either by taking the highest value inside each heatmap, or by using a second neural network to learn the mapping from heatmaps to (x_j, y_j) pairs [163].

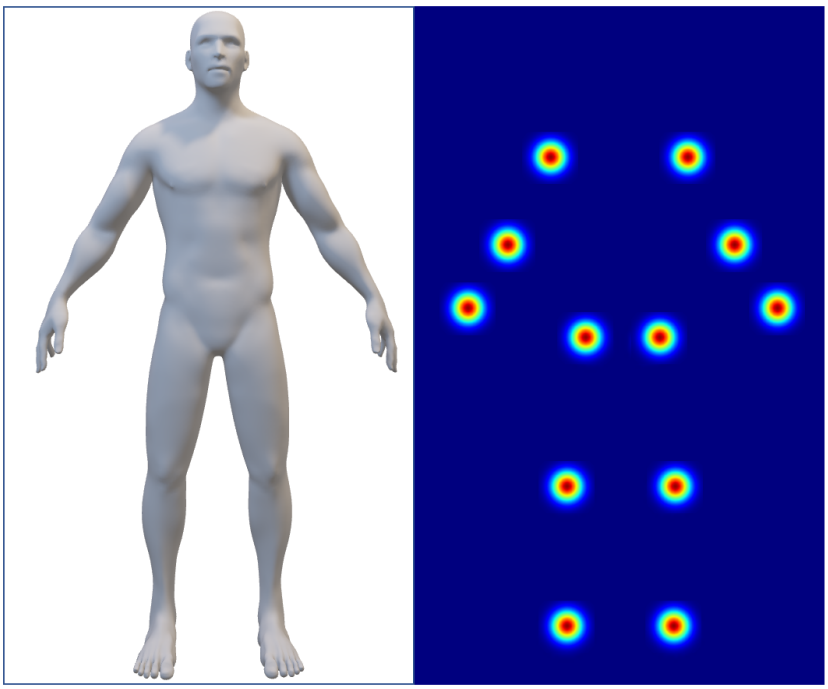


Figure 2.22: Given an input image (left) and a set of joint annotations (x_j, y_j), one heatmap per joint can be obtained by creating a 2D Gaussian centred at (x_j, y_j). In the image on the right, all heatmaps appear in the same image to make their visualisation easier; normally, however, the N heatmaps would be processed separately by a neural network. (Image created by the author)

Tompson et al. [179] are often accredited as the first authors to use heatmaps as labels instead of pure (x, y) pairs [5, 170, 172]. In their paper, Tompson et al. justify their use of heatmaps with the following statement: 'the mapping from input RGB image to XY location adds unnecessary learning complexity which weakens generalization. [...] Since the network is forced to produce a single output for a given regression input [*i.e. an* (x_j, y_j) *pair*], the network does not have enough degrees of freedom in the output representation to afford small errors which we believe leads to over-training (since small outliers will contribute to a large error in XY)'. As I will endeavour to demonstrate in the following paragraphs, this explanation is incomplete. After Tompson et al.'s seminal paper, most authors adopted heatmaps as the default labels with which to train their models, but never pursued any explanation of why heatmaps should be used—nor, indeed, how. This lack of a strong theoretical understanding of the benefits of heatmaps has led to their being used inefficiently.

What Tompson et al. failed to state explicitly is that heatmaps work particularly well as ground truths for 2D pose detection because they encode in the label a prior belief that the label may be inaccurate. If we were absolutely certain that all of our labels were accurate, we *would* want the model to be penalised even for 'small errors'. But because the labels in most 2D pose detection datasets are inherently unreliable (since the process of labelling is complicated by the presence of clothing, occlusions, and foreshortening), modelling this uncertainty by using heatmaps allows the neural network to not be punished for incorrect predictions that it might have learned from noisy labels. In other words, using heatmaps relaxes the strictness of supervision[44], allowing the model to learn a more general representation of 2D pose instead of overfitting labels that may be noisy to begin with. This concept can be rephrased using a terminology more akin to the one used by Tompson et al.: using heatmaps is a special form of data augmentation that lets low-confidence labels be used, labels that allow the model to circumvent our uncertainty in the original binary labels.

With this new rationale for the use of heatmaps, we are better equipped to discuss what is the most appropriate use of heatmaps as labels for 2D pose detection algorithms. Let us begin by considering the case of Rafi et al. [170], who, instead of generating heatmaps by using 2D Gaussians, set to one the values of all pixels in a circle with a radius of eight pixels and centred on $(x_j,\ y_j)$; all the pixels outside the circle were set to zero. Although this approach still succeeds at augmenting the labels—thus alleviating the strictness of supervision—using a flat-valued circle centred at $(x_j,\ y_j)$ is likely too extreme a measure: it gives freedom to the model to be incorrect by eight pixels, with no penalty. The elegance behind the use of a heatmap is that if the model's guess is within 2σ from $(x_j,\ y_j)$, the algorithm is penalised less than normal but it still receives feedback that allows it to improve (i.e., its loss function is greater than zero). Bulat et al. [163] propose a modification of Rafi et al.'s use of heatmaps. First, for each joint they set to one the values of all the pixels inside a circle of radius $\mathrm{r} = 10$ and centred on $(x_j,\ y_j)$; these heatmaps are then used as inputs to train a neural network, which learns to output similar heatmaps. The outputs of this neural network are then used as inputs for a second neural network, whose labels are again heatmaps, but this time generated using

[44]Machine learning models are said to be 'supervised' when the examples they are trained on have associated labels.

a Gaussian with $\sigma = 5$. This approach is strictly superior to the one adopted by Rafi et al., because the high degree of uncertainty encoded in the first stage of the model is mitigated by the second stage, which refines the uncertainty by modelling it as a Gaussian distribution. This brings us to an important consideration: how to set the value σ of the Gaussian distribution. In Tompson et al.'s paper, σ was set to 1.5 pixels; in Rafi et al.'s, it was set to 5; in several other papers it is set to 1 [5,6,172]; in many others, a value for σ is not specified [171,173,180–183]. Figure 2.23 offers a visualisation of the effect of different values of σ. Essentially, the greater the value of σ, the less confident we are about the original labels (x_j, y_j) and the less strict the supervision will be.

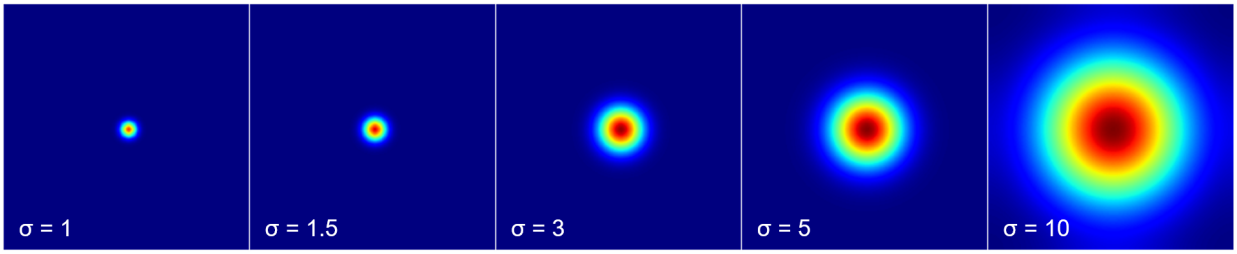


Figure 2.23: As we increase σ, the area of non-zero pixels increases. The images in this figure have a resolution of 50 x 50 pixels but were smoothed using a filter to make them easier to interpret. (Image created by the author)

What motivates authors to choose any given value of σ? Bulat et al. [163] justify their choice of $\sigma = 5$ by stating that that value was found empirically; the same justification is given by Belagiannis et al. [184], who, however, reached a value of 1.3 instead. Empirical fine-tuning of parameters is at the core of deep learning, and should not be discouraged. However, something is missing from this approach. As stated earlier, the greater the value of σ, the less confident we are about the original labels. By fixing σ to one value for all joints of all images, we are stating that we have the same amount of uncertainty for all joints of all images. This is clearly not true. Crucially, it is not true because of the presence in most images of occluded joints, the labelling of which must inevitably have been less reliable than that of visible joints. I would argue, then, that the heatmaps for occluded joints should be created by using a value of σ greater than that used for creating heatmaps of visible joints. The two values could then be fine-tuned empirically, perhaps starting with $\sigma_{occluded} = 5$ and $\sigma_{visible} = 1$. This approach is only viable when training on datasets that provide a *is_visible* label for each joint, like MPII.

2.5.5 Effective Architectural Designs

2.5.5.1 The 'Stacked Hourglass' Architecture

The publication of the Stacked Hourglass algorithm [5] was as important for the field of 2D pose detection as that of the U-Net algorithm [111] was for the field of biomedical image segmentation[45]. Indeed, the Stacked Hourglass algorithm is the de facto baseline model for 2D pose detection: it is often used as an intermediary step in complex pipelines that require automatic 2D pose detection (such as 2D-to-3D algorithms [21,54,57]), and many newer algorithms for 2D pose detection are essentially modified Stacked Hourglass networks [6–8, 185, 186]. The basic module of the Stacked Hourglass network is an encoder-decoder pair (see Figure 2.24): in the encoder, a series of convolutional and pooling layers encode the input image into progressively higher-level features, which have progressively lower resolution and higher semantic content; then, the features of the deepest layer of the encoder are upsampled (or 'decoded') some number of times, so as to restore the original shape of the image[46]. The features of the decoder are, therefore, the highest-level features of the encoder, upsampled once at each layer of the decoder. Because in the Stacked Hourglass network the encoder reaches very low resolutions and the decoder is a perfectly symmetrical copy of the encoder, an encoder-decoder pair takes the name of 'hourglass module'. Several hourglass modules can then be stacked together in series, feeding the output of the decoder of hourglass H_{i-1} to the encoder of hourglass H_i. The highest-resolution features of each hourglass are subjected to intermediate supervision (i.e. they produce heatmaps to which a loss function can be applied) and are connected to one another via skip connections.

Why is the Stacked Hourglass design effective? In particular, it is interesting to consider why it is more effective to stack hourglass modules than to use a single one, as in a U-Net network. The most obvious answer is that by stacking multiple hourglasses we increase the capacity of the model. A model's capacity is a measure of the complexity of the functions that

[45] Interestingly, the two algorithms were published within 12 months of one another and share striking similarity in their architecture. In fact, the Stacked Hourglass model could effectively be renamed to 'Stacked U-Net', as the 'hourglass module' has almost the same architecture as a U-Net with ResNet [98] as its backbone.

[46] The process of first encoding and then decoding information within a network is sometimes referred to as 'top-down, bottom-up inference' [5, 8].

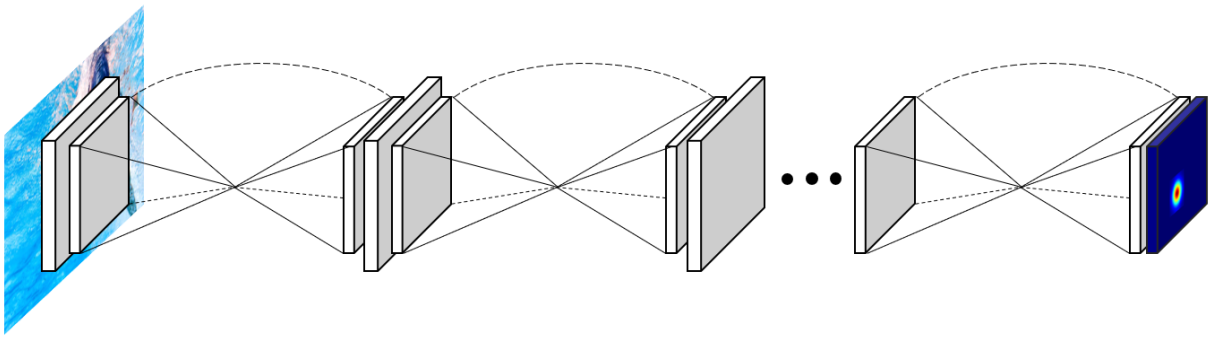


Figure 2.24: Architecture of the Stacked Hourglass network. (Image adapted from [5]) Note that a single encoder-decoder couple would form a structure almost identical to U-Net.

the model can learn [187], and it can be roughly estimated using the number of parameters of the model: larger models with more parameters tend to have higher capacity[47]. If the effectiveness of the Stacked Hourglass model was entirely attributable to an increase in the number of parameters with respect to a model like U-Net, it would mean that a simple way to generate a better model would be to stack more and more hourglass modules, as many as the hardware allowed. But the same could be said about almost any model: more computational power (in the sense of number of parameters) would almost always lead to better results, as per Sutton's Bitter Lesson [189]. In the original Stacked Hourglass paper, Newell et al. address this concern by comparing three Stacked Hourglass models with the same number of parameters but different numbers of hourglass modules: two, four, or eight (see Figure 2.25). The results of their experiment show a modest but relevant improvement as more hourglass modules are used: on the MPII dataset, the two-stack, four-stack, and eight-stack models reach PCKh of, respectively, 87.4%, 87.8%, and 88.1%. Furthermore, halfway through each network the corresponding PCKh values of the intermediate predictions were 84.6%, 86.5%, and 87.1%. These results demonstrate that the accuracy of the Stacked Hourglass network is not entirely attributable to increased capacity, and that the repetition of several hourglass modules also plays an important role. In particular, it demonstrates that stacking several hourglass modules and forcing them to produce intermediate heatmaps allows the model to develop a high-level understanding of the output early on in the network. This is particularly

[47]This measure of capacity is quite informal and not always accurate, as the capacity of a model depends on factors other than the number of parameters—for example, it depends on how these parameters are connected. A more mathematically rigorous measure of capacity is the Vapnik—Chervonenkis (VC) dimension [188], which however can only provide a loose lower bound on the generalization error of the model [187].

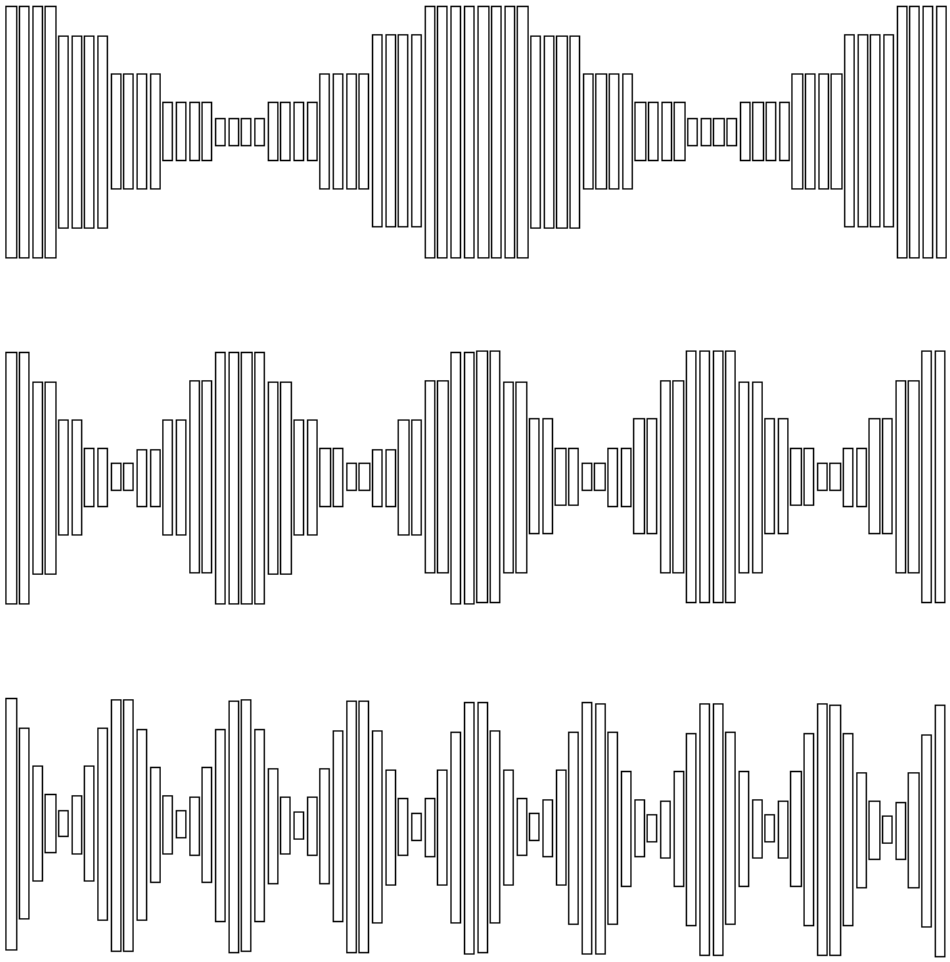


Figure 2.25: From top to bottom: a two-, four-, and eight-stack Hourglass network. The number of parameters is the same for each network. This is achieved by doubling the number of layers in each hourglass module every time the number of stacks is halved. For example, the four-stack model has twice as many layers per module as the eight-stack model. (Image adapted from [5])

important for 2D pose detection, in which the output is structured: the relationship between the position of the joints is deterministic, not casual (except, perhaps, for Cirque du Soleil contortionists). In other words, having information about where a joint is (the left elbow, for example) can help us restrict the space of possible locations for a related joint (like the left wrist) to ones that are anatomically feasible. Therefore, the ability of the Stacked Hourglass model to learn a high-level representation of human pose early on in the network allows it to reuse this high-level information in all subsequent modules, which progressively refine it using the constraints learned from the previous modules. This is achieved by two key design choices found in the architecture of the Stacked Hourglass: the presence of several stages of deep supervision, which allow the generation of intermediate heatmaps; and the repetition of

identical blocks (the hourglass modules), which ensures that the intermediate heatmaps all have the same resolution, which in turn means that subsequent modules have to learn more refined representations while maintaining the same resolution.

2.5.5.2 Structured Output Priors

By progressively refining representations learned early on, the Stacked Hourglass model is able to learn some form of structure in its output. However, the structure of the output is not learned explicitly and cannot be easily defined. For example, we do not know what kind of structured output is learned after four stacks of hourglass modules, or five, nor how it differs from that learned after seven stacks. Instead, the network is left to learn its own function from the data. A different approach consists in explicitly guiding the network towards a structured output, in which there is a relationship between the position of connected joints; in statistical terms, we would say that we are forcing on the model a prior belief that the output function should be structured in a specific way. In early works on 2D pose detection [135, 167, 168, 174, 190–193], the strength of this prior was excessive: it forced the output to follow a rigid structure, in which the relationship between joints followed strict anatomical or statistically learned mathematical functions. In modern computer vision approaches, forcing such a strong prior on a model is seen as bad practice [189]: since our understanding of a computer vision problem could be flawed or incomplete, we want to avoid forcing it on a model, and instead allow it to explore all possibilities just by looking at the data. However, this 'dogma' of modern computer vision approaches is more pertinent to some applications than to others. For example, in object detection, it is exceptionally difficult to manually engineer features that encode our prior belief of what a dog or a cat or a boat ought to look like, and it is quite possible that we would not be able to capture all the nuances of those categories and may thus hamper the learning of our model. In the case of 2D pose detection, however, there are ways of enforcing very weak priors that can nonetheless restrict the space of possible poses. For example, the model of Gkioxari et al. [182] predicts joint locations sequentially: the prediction of each joint is dependent on all previously predicted joints. Mathematically, let $Y = \{Y_j\}_{j=0}^{N-1}$ be the the N joints to be detected, given an input image X and the labels $(x_j,\ y_j)$ for each joint j. The

model of Gkioxari et al. tries to learn the following probability distribution:

$$P(Y = y \mid X) = P(Y_0 = y_0 \mid X)\ \Pi_{j=0}^{N-1}\ P(Y_j = y_j \mid X,\ y_0,\ y_1,\ ...,\ y_{j-1}) \tag{2.18}$$

This type of prior is 'weak' because it only tells the model that there should be a relationship between the joints and it tells it the direction in which the relationship should be expressed (i.e. joint Y_j has information useful for predicting joint Y_{j+1}, but not necessarily for predicting joint Y_{j-1}); it does not tell the model *what kind* of relationship it should learn, something the model can still learn freely from the data. The model of Gkioxari et al. achieved 85.0% PCKh on MPII, which is substantially lower than the 90.9% achieved by Stacked Hourglass. However, the comparison is not fair: the novelty of Gkioxari et al.'s work is in the way they represent the output, but the architecture they used is much smaller than that used in Stacked Hourglass. In fact, Gkioxari et al.'s model has fewer parameters than a single hourglass module.

A better way to understand if explicitly enforcing a structured output is beneficial is by analysing the work of Tang et al. [6]. In their paper, Tang et al. propose to use an eight-stack hourglass network with a branching output. The number of branches corresponds to the number of groups into which joints can be divided based on their relationship with one another. For example, using our understanding of human anatomy we could split the 16 joints labelled in MPII into six groups: (1) head top, upper neck, thorax; (2) left wrist, left elbow, left shoulder; (3) right wrist, right elbow, right shoulder; (4) left knee, left ankle; (5) right knee, right ankle; (6) left hip, right hip, and pelvis. With this grouping, the network would have six separate branches, each predicting the three joints associated with that group. Another way to group the joints is by calculating their mutual information, which is expressed by the equation:

$$I(l_m, l_n) = \sum_{l_m \in L} \sum_{l_m \in L} p(l_m, l_n) log \left(\frac{p(l_m, l_n)}{p(l_m), p(l_n)} \right) \tag{2.19}$$

The mutual information of two random variables l_m and l_n quantifies the amount of infor-

mation that is obtained about one variable by observing the other[48]. Using the joints labelled in the MPII dataset, Tang et al. computed the table in Figure 2.26. They then performed spectral clustering [194] on the data in the table to split the joints into arbitrarily large clusters of associated joints. For example, by forcing the spectral clustering to divide the joints into six clusters (the elements of which showed higher mutual information with one another than with the elements of any other group), Tang et al. found the following grouping: (1) head top, upper neck; (2) thorax, left shoulder, right shoulder; (3) left wrist, left elbow; (4) right wrist, right elbow; (5) left ankle, right ankle, left knee, right knee; (6) left hip, pelvis, and right hip. The fact that the grouping found using mutual information is different from the one found using our intuition is a reminder of the fact that enforcing hand-crafted priors onto deep learning models can be dangerous: we might force the model to learn patterns that are not those naturally found in the data. The advantage of using mutual information and spectral clustering is that it allows to enforce the same strength of prior (the model is still forced to predict outputs divided into six groups), but instead of using a prior derived from our intuition, it allows us to derive one from the data, which means that it will be much more objective. Indeed, Tang et al. found that the grouping strategy based on mutual information always outperformed the hand-crafted grouping strategy under different numbers of clusters.

The comparison with the Stacked Hourglass network is much more direct this time, since the main difference between the two is in how their output is structured. Since Tang et al.'s model outperforms Stacked Hourglass by 1.8% PCKh, we can safely conclude that enforcing a weak prior on the model in the form of a structured output is beneficial for 2D pose detection, as it confines the model to exploring solutions that are compatible with the enforced prior.

2.5.5.3 Parallel Stacking

In the Stacked Hourglass network, the hourglass modules are stacked in series (see Figure 2.24). This means that in each hourglass module the information is first encoded down to very low

[48] Unlike Pearson's correlation coefficient, mutual information can capture non-linear associations between variables, making it more suitable for the purposes of training a neural network [6].

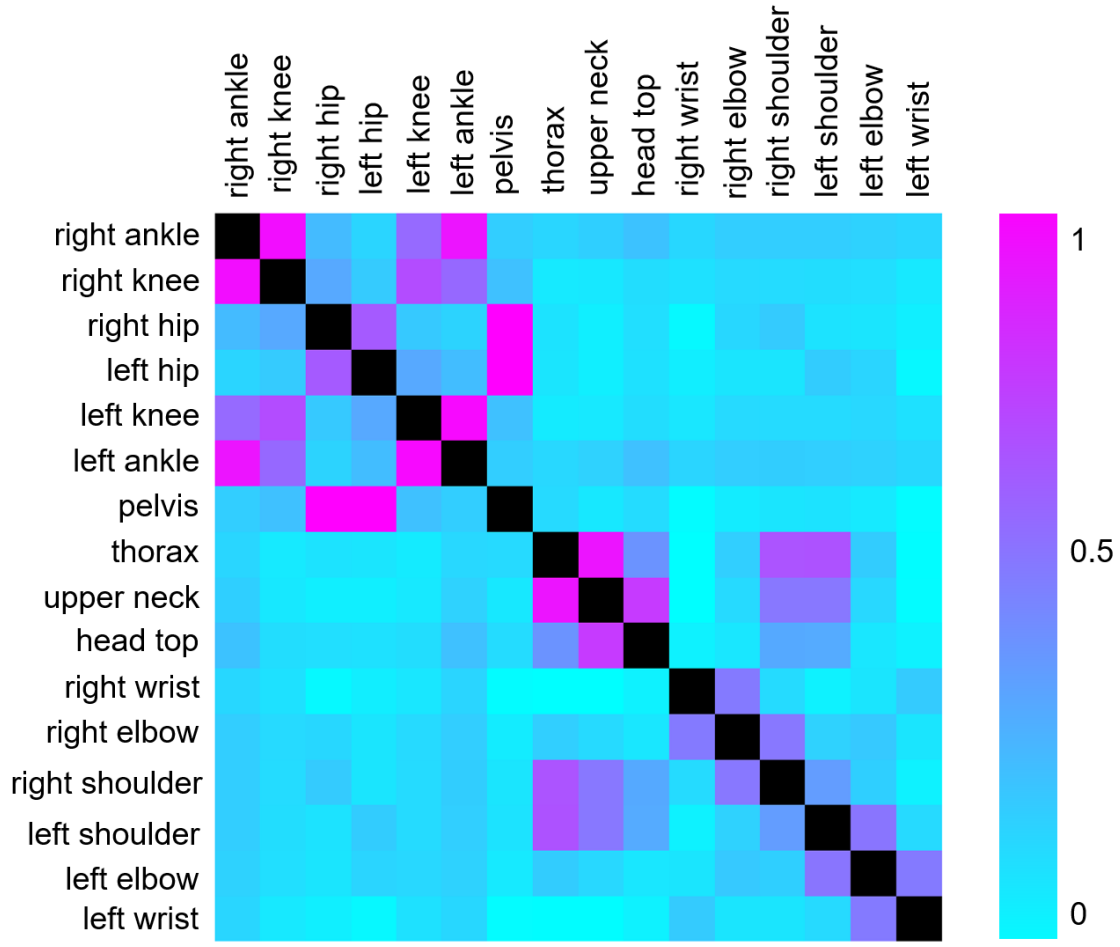


Figure 2.26: Mutual information between each pair of joints, as calculated by Tang et al. [6] using the labels in the MPII dataset. All values are normalised to the range [0, 1]. (Image adapted from [6])

resolutions, and then must be restored to the original input resolution. An alternative approach, proposed by Sun et al. [7], is to stack modules in parallel rather than in series (see Figure 2.27). The blocks in Figure 2.27 are residual blocks, the same ones used to build the encoder of an hourglass module. Given their arrangement in parallel, however, a decoder is not necessary: the information is processed simultaneously across different levels of resolution, and fused together through multi-level skip connections. This parallelisation strategy is reminiscent of the success of Convolutional Networks over Fully Connected Networks, success which showed that parallel processing of information is much more efficient than in-series processing. In this case, too, parallelisation proved to be effective: Sun et al.'s model, which has fewer parameters than an eight-stack Hourglass network, outperforms it by 1.4% PCKh on MPII.

An even more remarkable improvement over Stacked Hourglass was achieved by Su et al. [8], who also use parallelisation in their architecture, though in a simpler way. Instead of

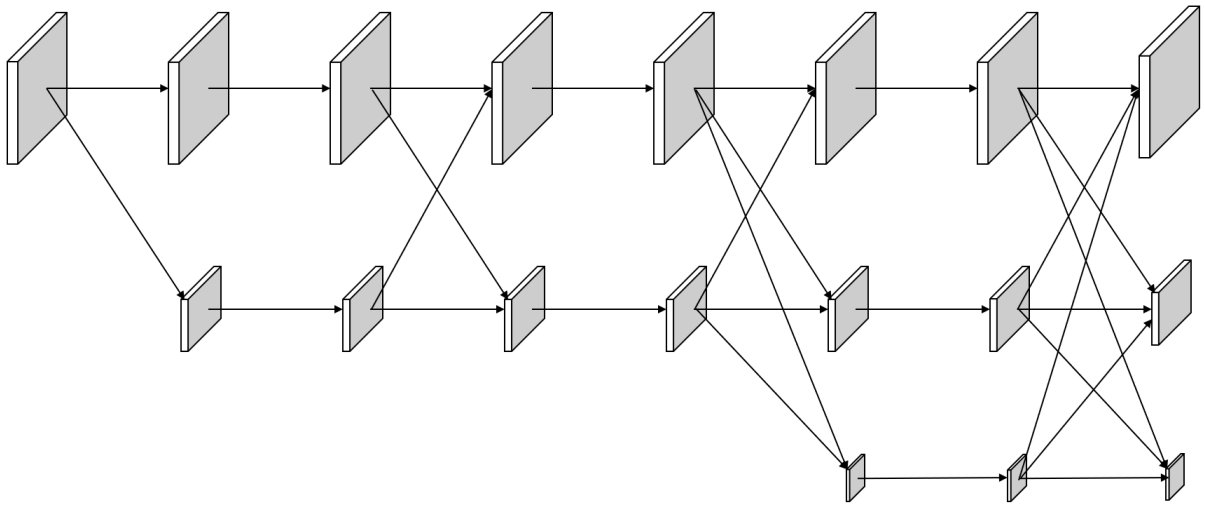


Figure 2.27: Architecture of the model proposed by Sun et al. [7]. At each parallel branching, the resolution of the feature maps is halved and the number of their filters doubled, as would happen in the encoder of an hourglass module. (Image adapted from [7])

completely re-structuring Stacked Hourglass, they stack in parallel—rather than in series—several hourglass modules (see Figure 2.28). The key difference between Su et al.'s architecture and that of a Stacked Hourglass is that in Su et al.'s network each module has access to the output of the previous module *as well as* to the original input. Therefore, this architecture is not entirely in parallel and not entirely in series, but a combination of the two: the connection in series is given by the fact that the output of a module is given as input to the next; the connection in parallel is given by the fact that all modules have access to the original input. Su et al.'s algorithm achieved a remarkable 93.9% PCKh on MPII, 3% higher than Stacked Hourglass. However, at least 1.5% PCKh (as visible from the graphs found in Su et al.'s paper) is attributable to the fact that, before testing on MPII, they trained the model on the MPII training set but also on the HSSK dataset. Since the results published by most other researchers are obtained by training models exclusively on MPII, it is not correct to compare Su et al.'s algorithm to others. Nevertheless, given that Su et al. attribute roughly 1.5% PCKh to their training on the HSSK dataset, their adjusted result (92.4%) with training only on MPII would still be a marked improvement over Stacked Hourglass.

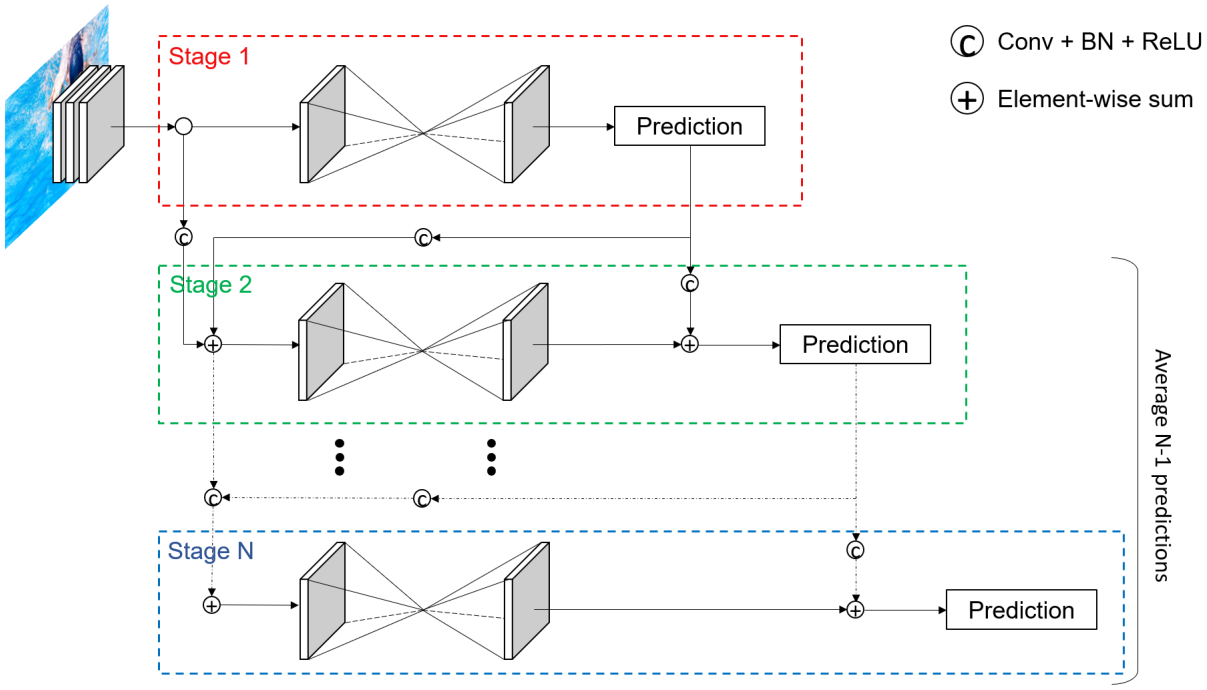


Figure 2.28: Architecture of the model proposed by Su et al. [8]. Here, N hourglass modules are stacked in parallel rather than in series. (Image adapted from [8])

2.5.5.4 Adversarial Augmentation

Data augmentation is one of the most commonly used strategies to reduce the generalisation error of a neural network [187]. It consists in performing certain types of transformations on all the images in a dataset, thus generating new data samples. For example, every image could be flipped vertically and horizontally, thus creating a dataset with three times as many images as it originally had. In 2D pose detection, the transformations typically performed are left-right flipping, colour jittering, random rotation, and random scaling—which essentially means that the image is zoomed in or out by a certain amount, and then resized to the desired resolution [5, 7, 170, 176, 185, 195]. For transformations that require randomisation (rotation, colour jittering, and scaling), an upper and lower bound are given (for rotations, these might be -30°and +30°), and samples are then drawn from a Gaussian distribution thusly bounded. Rafi et al. [170] found that using narrower bounds (for example, -5°and +5°of rotation instead of -30°and +30°) led to a very significant drop in performance when they tested their algorithm on the FLIC dataset [168], thus illustrating the importance of data augmentation for 2D pose detection algorithms.

Nevertheless, traditional data augmentation schemes are fundamentally flawed, for two reasons. First, they are disentangled from the optimisation of the algorithm: they are used as a pre-processing step, they do not change dynamically during the training of the network. Since we cannot use the entire Gaussian distribution of augmentations (the computational cost of doing so would be unmanageable), we have to select random samples from it, but the random samples we select might not be the ones that would have helped the training of the algorithm the most. Because there is no communication between the data augmentation phase and the training phase, we do not have any guide for sampling the data augmentation better than random selection. Second, by sampling the transformations from Gaussian distributions, traditional data augmentation schemes tend to include very few (if any) examples from the long-tailed portion of the Gaussian distribution, which may be a useful region to explore—for instance, it would make the algorithm more robust against outliers in the data. In the specific case of 2D pose detection, traditional data augmentation schemes also have a third limitation: they do not address the problem of occlusions, which is one of the key challenges of this task. Ideally, we would want a data augmentation scheme that made 2D pose detection algorithms more robust against occlusions. An attempt at such an augmentation scheme was offered by Ke et al. [172], who, as a data augmentation technique, cropped parts of the background of images and placed them—in the same image—on top of visible joints, thus artificially occluding them (see Figure 2.29). The purpose of this was to make the algorithm more robust to occlusions by artificially creating more in the data. Furthermore, they cropped patches in which joints appear, and pasted them in the background of the same image, labelling the pasted pixels as background. The purpose of this second patch was to make the algorithm ignore portions of the background that may look like people. This approach has two limitations. First, the patches used come from the same image, meaning the added variance is limited. Second, the patch containing a joint and placed in the background is not realistic: only a few of its pixels contain information relative to the joint, and a model that would have learned a structured output would not find such a patch difficult to classify as background, as all the pixels around it are disconnected from it. Nevertheless, Ke et al. trained a modified version of the Stacked Hourglass network using this data augmentation scheme and achieved 92.1% PCKh on MPII—a

1.2% improvement over Stacked Hourglass.

Figure 2.29: In this image, the right joint was artificially occluded by using a patch of pixels coming from the background. Furthermore, a patch from a different image containing the pixels around a joint (in this case, an elbow) was added to the background and labelled as such. The purpose of the first patch is to make the algorithm more robust to occlusions by artificially creating more in the data; the purpose of the second patch is to make the algorithm ignore portions of the background that may look like people. (Image created by the author)

The idea of addressing occlusions via data augmentation was taken further by Bin et al. [195], who structured it as an adversarial learning scheme. First, they used an algorithm to segment body parts from images in the LIP dataset [196]. This allowed them to collect a large database of realistic body parts. Their goal was then to paste these segmented body parts onto the images of MPII and LSP to create more occlusions and more difficult training examples (see Figure 2.30). This approach would already be an improvement over the method of Ke et al., since the patches added to the images come from different images (thus increasing the variability added by the augmentation) and are realistic, full body segments (and thus are harder for the network to classify as background). However, the true innovation of Bin et al.'s augmentation scheme lies in the way the augmentation is done. Instead of randomly sampling from the database of segmented body parts and randomly assigning the samples to input images, they devise an adversarial learning scheme. First, they take a pre-trained 2D pose detection

Figure 2.30: An example of the type of occlusions generated by the data augmentation scheme proposed by Bin et al. (Image created by the author)

network [7]; let us call this network the discriminator. Then, they build a simple network that, given an image, can paste in a random position a segmented body part sampled randomly from the database; let us call this network the generator. The output of the generator is then fed into the discriminator, which predicts the joint locations for the augmented image. If the predictions are correct, it means that the augmentation sampled by the generator was not effective, and the generator is discouraged from producing similar augmentations. If the predictions are incorrect, it means that the augmentation sampled by the generator was effective, and the generator is encouraged to produce more outputs of that kind; at the same time, the discriminator, which failed on this example, can learn from it and improve its performance. Using this adversarial scheme allowed Bin et al. to sample types of augmentation that directly target occlusions and that directly target the weaknesses of the baseline network. In other words, the sampling is not at random from a distribution that may or may not be useful for the generalisation of the network; it dynamically changes as the discriminator improves, addressing exactly the areas that it is weak in. This data augmentation scheme allowed Bin et al. to reach 94.1% PCKh on MPII, which as of October 2020 grants them the top position on the MPII leaderboard (http://human-pose.mpi-inf.mpg.de/#results). This result is particularly significant because it

represents a 1.8% improvement over the baseline model they used as a discriminator—which, coincidentally, was Su et al.'s algorithm (described in Section 2.5.5.3), which already was a state-of-the-art algorithm without Bin et al.'s augmentation scheme.

2.6 Conclusions

This chapter started by introducing the three criteria by which the validity of a motion capture system can be assessed: speed (the systems must be quick to set up and results should be outputted within a few hours of the recording session); non-invasiveness (athletes must not have their movement restricted in any way); accuracy (the system must accurately identify the 3D location of the joint centres). From a brief review on the literature on motion capture systems (which is presented in more detail in Appendices A to D), it emerged that all 'traditional' methods have at least one flaw that makes them sub-optimal for use in swimming research and as a monitoring and feedback tool:

- Marker-based systems (also known as OSSs) like Vicon or Qualisys arguably respect the first criterion (though setup times may be high), but violate the second (the presence of markers substantially increases drag and may restrict movement);
- Sensor-based systems respect the first criterion (though setup times may be high if several sensors need to be used), but violate the second (the presence of sensors substantially increases drag and may restrict movement);
- Manual digitisation respects the second criterion (it is a completely non-invasive technique) but does not respect the first (the computation of parameters is too slow to be practical);
- Depth-based systems are the least accurate systems, and the setup they require would be impractical for swimming motion capture.

It was then concluded that the best solution for swimming motion capture would be an image-based markerless motion capture system—specifically, a generative image-based system,

since discriminative methods require large amounts of labelled 3D data, which is not readily available. Evidence shows that for generative methods to perform optimally, two types of 'intermediate information' need to be extracted from images: the silhouette of the person, and the locations of their joints in camera coordinates (i.e. the '2D joints' or '2D pose'). Due to the No Free Lunch Theorem, existing algorithms for silhouette extraction and 2D pose detection perform poorly on images of swimmers, because during training they were never shown such images. Therefore, to develop an image-based markerless system for swimming motion capture, it is necessary to first develop algorithms well-suited to perform silhouette extraction and 2D pose detection on images of swimmers.

Consequently, the fields of silhouette extraction and 2D pose detection were reviewed next, with two goals: to describe the datasets on which these algorithms are usually trained; and to describe the most effective features of the top-performing algorithms in these two fields. From the description of datasets, the main observation that can be made is that even the largest and most popular datasets available (PASCAL VOC for silhouette extraction and MPII for 2D pose detection) do not contain images of swimmers. This means that swimming-specific datasets had to be developed during this PhD to train the algorithms that were to be developed. From the description of the state-of-the-art algorithms, it emerged that Convolutional Neural Networks (CNNs) are at the core of all modern algorithms for silhouette extraction and 2D pose detection, indicating that the algorithms to be developed in this PhD likely would need to also be based on CNNs. In particular, encoder-decoder architectures are at the top of the leaderboards of both fields of research (ExFuse and DeepLab for semantic segmentation; Stacked Hourglass and its derivatives for 2D pose detection).

However, in Section 2.3.2 was described a generative method for image-based markerless motion capture that only requires silhouettes as inputs: the visual hull. The visual hull has two key advantages over other generative systems: it is relatively easy to implement, and it does not require 2D joints as inputs. If such a system could be used for swimming motion capture, then, the only requirements for it would be a dataset and an algorithm for silhouette extraction (and not also ones for 2D pose detection). Therefore, the next chapter discusses the visual hull in more detail, to understand why it was ultimately deemed unsuitable for swimming

motion capture and discarded in favour of generative methods that also require 2D joints to be labelled.

Chapter 3

Preliminary Research: The Visual Hull

3.1 Introduction

The concept of the visual hull was introduced by Laurentini in 1994 [70]. The visual hull is an approximation of the volume of an object: its volume will at least equal that of the object, but in most cases it will be greater. To compute the visual hull of an object, multiple calibrated cameras pointed towards the object are needed. On each camera plane, the object appears as a two-dimensional surface bounded by a silhouette. Let us consider the case when only one camera is used. If we know the position of the camera in a global coordinate system (information which we obtain by calibrating the cameras) and the position of the object's silhouette in camera coordinates, we can imagine to connect the camera's projection centre with all the points along the silhouette. If then we extend these rays towards infinity, we define a cone within which the object must lie (see Figure 3.1). If the process is repeated for multiple calibrated cameras, the intersection of the cones—all of which represent simultaneous constraints for where the object can lie—yields an approximation of the volume of the object; this approximation is known as the visual hull of the object. In biomechanics, the visual hull has been used to study the tennis serve [71], [72]; to perform gait analysis [69]; to analyse the movement pattern of gymnasts [22]; and by Ceseracciu et al. to measure arm movements during front crawl swimming [23]. The results of Ceseracciu et al. were unsatisfactory: the least accurate joint of their model was the

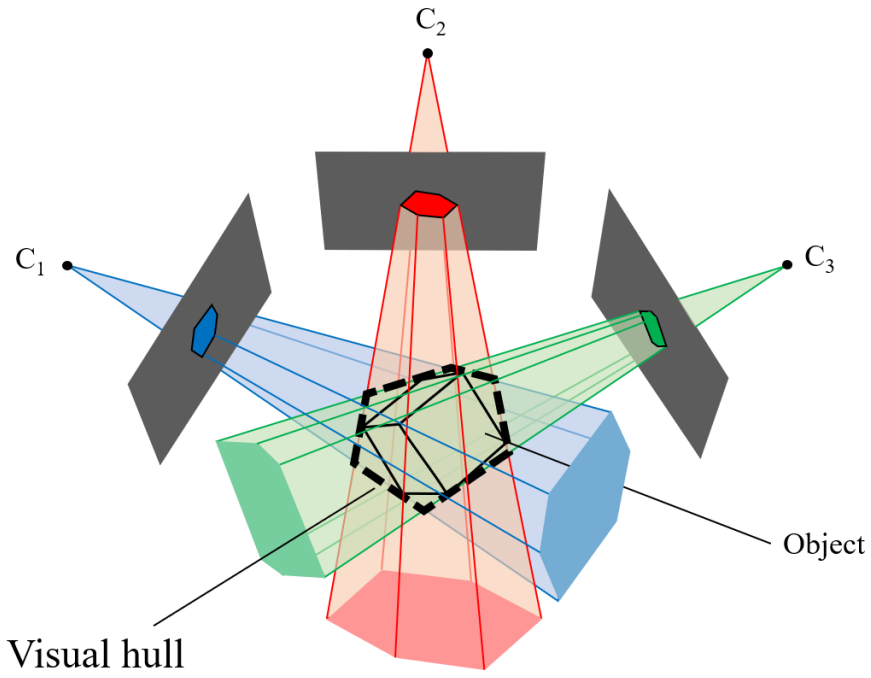


Figure 3.1: Illustration of how the visual hull of an object is obtained. Suppose there are three cameras (C1, C2, and C3) evenly distributed, at a fixed distance, around the object (a cube, in this case) to be reconstructed. What each camera sees is the two-dimensional shape (here colour-coded in blue, red, and green) of the object from a particular angle, separated from the background (here colour-coded in dark grey) by its contour (here colour-coded in black). If, for each camera, imaginary rays were to connect the origin of the camera (C1, C2, or C3) with each point along the silhouette of the object, a three-dimensional cone would be formed, within which the original object lies. By intersecting the cones defined by each camera view, the visual hull is obtained. (Image created by the author)

shoulder joint, which deviated from the ground truth by a mean of 155 mm, while the most accurate joint was the wrist, which deviated from the ground truth by a mean of about 56 mm. One of the sources of error for Ceseracciu et al.'s model was certainly the fact that it used a mixture of Gaussian method [197] for silhouette extraction. As discussed in Section 2.4.1.1, such an algorithm cannot deal well with the highly complex and ever-changing background of a swimming pool, and is likely to give poor, noisy silhouettes.

The extent to which the accuracy of the extracted silhouettes influences that of the reconstructed visual hull has not been quantified. Grauman et al. [198] and Gall et al. [199] have suggested that a small segmentation error in even one camera view could have a significant effect on the reconstructed visual hull. Though the authors did not quantify this 'significant effect', it seems intuitive that higher-quality silhouettes would give higher-quality visual hulls.

When constructing a visual hull, a 3D point is determined to be part of the visual hull if and only if its projection lies within the silhouette on all camera views; therefore, one noisy silhouette could spoil the quality of the entire visual hull [200]. In the example in Figure 3.2, however, even if S_1 had been perfect the accuracy of the reconstructed visual hull would have been poor. As will be discussed in the following paragraphs, the accuracy of a visual hull depends not only on the accuracy of the silhouettes, but also on the number of cameras and their placement relative to the object. Most research on the visual hull is conducted in controlled laboratories,

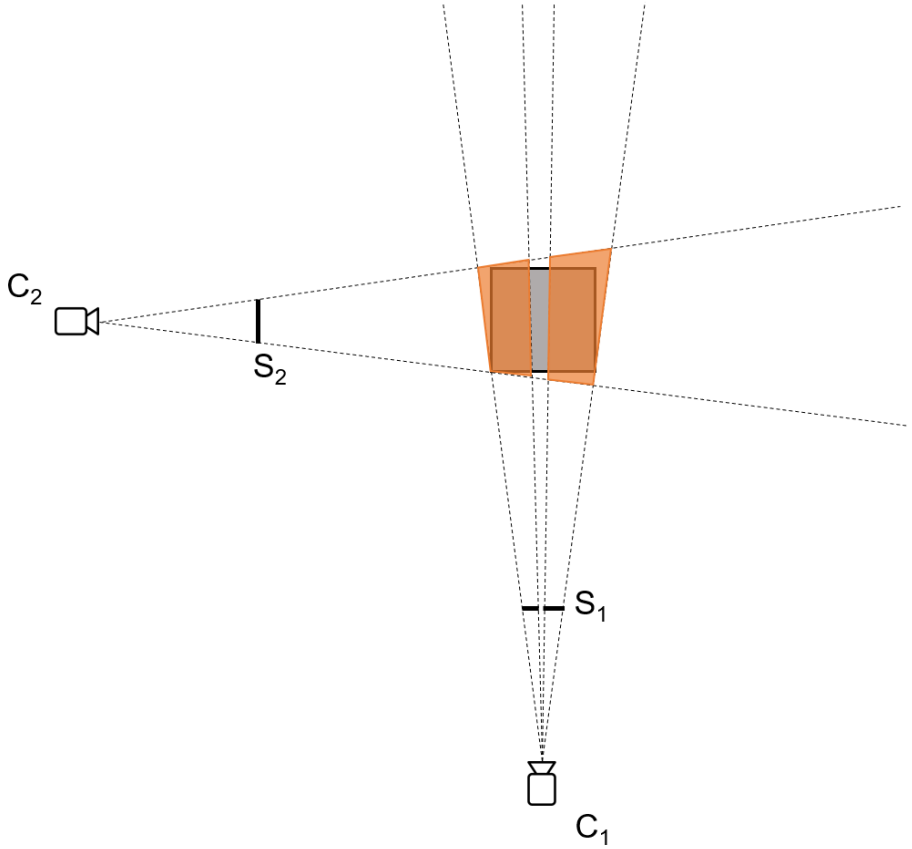


Figure 3.2: In this image, the silhouette (S_1) extracted by camera C_1 is inaccurate: it contains a hole in the middle. This inaccuracy is translated into an inaccuracy in the reconstructed visual hull (in orange). (Image created by the author)

allowing researchers to place a uniform, monochromatic, stable background behind the object to be reconstructed. This kind of background makes it possible to use basic background subtraction methods (see Section 2.4.1.1) to extract the silhouettes of the object [201,202]. Indeed, silhouette extraction is so trivial under such controlled conditions that several authors who have published papers on the visual hull did not even mention the silhouette extraction method used in their studies [203–205]. However, the background found in images of underwater swimmers

cannot be controlled; instead, it is unstable, making basic background subtraction algorithms perform poorly (see Figure 3.3). This suggests that if a visual hull (or any other generative method that relies on silhouettes) of swimmers has to be reconstructed, a sophisticated method for silhouette extraction must be used.

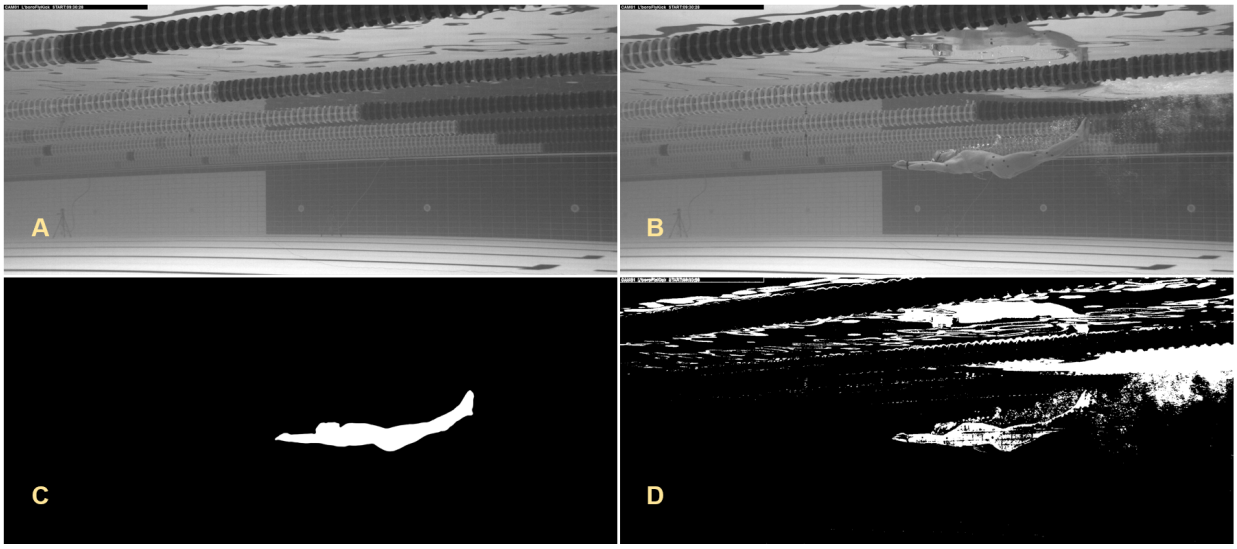


Figure 3.3: A: reference image used to initialise background subtraction (see Section 2.4.1.1). B: image containing a swimmer whose silhouette is to be extracted. C: silhouette extracted by hand. D: silhouette extracted using background subtraction. (Image created by the author)

3.2 Experiments on the Impact of Silhouette Accuracy on Visual Hull Accuracy

To provide a qualitative answer to the question of how silhouette accuracy impacts visual hull accuracy, I performed an experiment on the TempleRing dataset[1], which contains 47 images of a plaster reproduction of the 'Tempio dei Dioscuri' temple. Each image in the dataset was taken by the same camera, which was placed away from the object and then rotated around the object's vertical axis, for a total of 47 unique camera positions (each separately calibrated) that defined a circle around the object. I chose to perform experiments on the TempleRing dataset for two reasons: it contains many camera views, which, as will be discussed later in this section, is important when reconstructing visual hulls; and the background of the images is almost

[1] https://vision.middlebury.edu/mview/data

uniformly black, which makes silhouette extraction easier. This simple type of background allowed the reconstruction of a 'baseline' visual hull (using the algorithm described in [206]), one for which it is guaranteed that the silhouettes used were as accurate as possible. I then performed tests in which I deviated from the baseline by artificially altering the silhouettes so that they included a varying amount of random error. To achieve this, I applied to each silhouette (separately) a certain number of masks of size 10 x 10 pixels. Each mask was placed over a random patch of the silhouette, making sure that no two masks overlapped and that the distribution of masks was not the same for all silhouettes, so as to model only random—not systematic—silhouette inaccuracies. Each mask changed the values of the pixels of the silhouette it covered: if a pixel covered by a mask was originally labelled as background, the mask would label it as belonging to the silhouette; if it was originally labelled as belonging to the silhouette, the mask would label it as background (see Figure 3.4). Therefore, each

Figure 3.4: Left: original image. Middle: silhouette (considered to be as accurate as possible). Right: 20 binary masks (10 x 10 pixels large) were applied randomly to the silhouette and to the background, changing the accuracy of the silhouette's segmentation by about 0.1%. (Image created by the author)

mask represents a randomly generated error in the segmentation of the silhouette. The size of the masks was chosen so that they represented errors that, while small enough to reasonably occur in reality, were not so small that they might be viewed as Gaussian noise. The focus of the experiment was to understand how visual hull accuracy varied as the quality of the silhouettes was lowered. For example, given that the images in the TempleRing dataset have a resolution of 480 x 640 pixels, using twenty masks of size 10 x 10 would result in about a

0.1% decrease in silhouette accuracy. Would the visual hull reconstructed with these slightly worse silhouettes also be slightly worse? What if the decrease in silhouette accuracy was 1%, or 10%? The qualitative results of this experiment are reported in Figure 3.5, which shows that

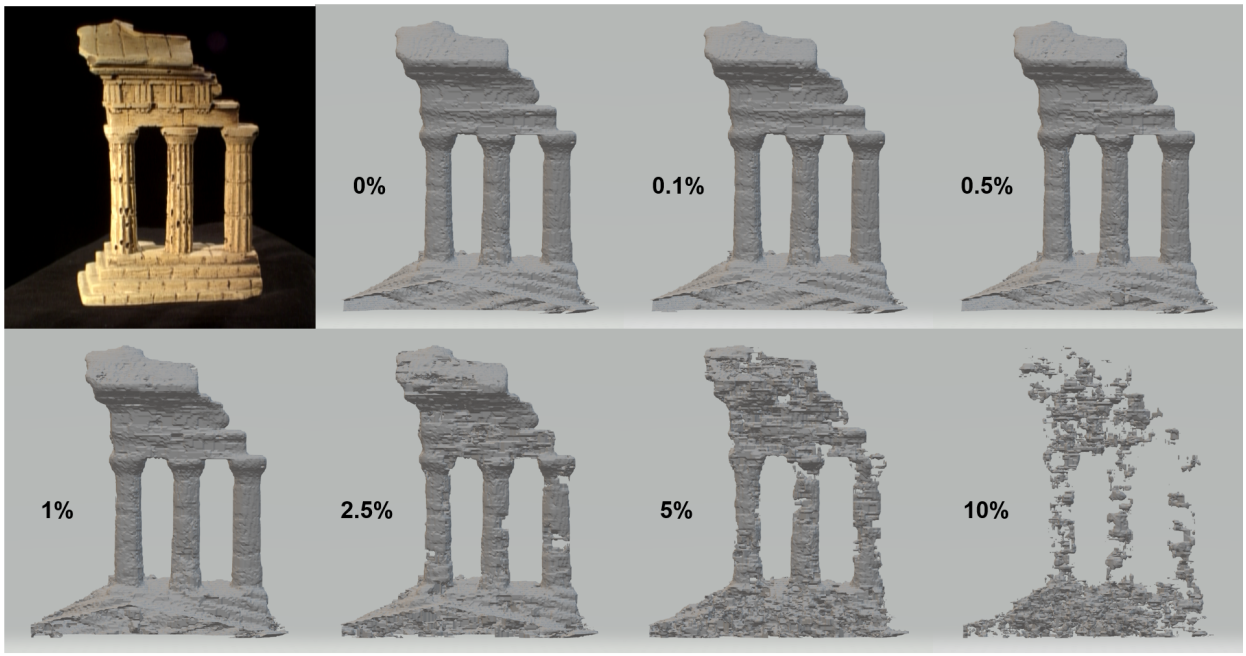


Figure 3.5: Qualitative interpretation of the effect of silhouette accuracy on visual hull accuracy. The top left image is one of the images in the TempleRing dataset. Every other image was obtained by reconstructing a visual hull from the TempleRing dataset using silhouettes that were altered by a percentage indicated by the number reported next to each image. (Image created by the author)

for silhouette errors up to 1% there is no substantial change in visual hull accuracy. However, for silhouette errors between 2.5% and 5% the reconstructed visual hull is noticeably worse, and for silhouette errors of 10% the reconstructed visual hull is poor. In Figure 3.3, images C and D differ by 75% (meaning 75% of the their pixels have opposite labels); even when cropped around the swimmer (which removes most of the noise in image D while not altering the quality of image C), the two images differ by 54%. This experiment thus provides qualitative results that strongly suggest that: 1) silhouette accuracy plays an important role in determining the accuracy of the reconstructed visual hull; in particular, silhouettes containing errors above 2.5% might give visual hulls that are too inaccurate; 2) background subtraction performed on images of swimmers can cause errors in the silhouettes above 2.5%, meaning that this algorithm cannot be used to obtain the silhouettes for the reconstruction of a visual hull.

3.3 Other Factors Impacting Visual Hull Accuracy

Silhouette accuracy is not the only factor that contributes to the accuracy of a visual hull. It is easy to see from Figures 3.1 and 3.2 how the number of cameras available, as well as their position relative to the object, greatly influences the accuracy of the reconstruction. Mündermann et al. are the only authors to date to have investigated the effect of camera number and position on the accuracy of the reconstruction of visual hulls of humans, and the results of their study are often cited as guidelines for determining how many cameras to use and where to place them when reconstructing visual hulls in biomechanics [22, 23]. In particular, Mündermann et al.'s study provided a lower bound (eight) for the number of cameras needed to obtain acceptable results, and an upper bound (16) above which using more cameras does not result in a meaningful increase in visual hull accuracy. The lower bound is of particular importance, because it defines the bare minimum equipment that researchers must use if they wish to reconstruct visual hulls. As an example of the consequences of using fewer than eight cameras, Mündermann et al. estimated that the volume of the visual hull of a person reconstructed using only four cameras is about 1.2 times larger than the true volume of the person (even if using highly accurate silhouettes). Equipped with these guidelines, we can now identify a second source of error in Ceseracciu et al.'s study on the reconstruction of visual hulls of swimmers: they only used six cameras, thus likely overestimating the volume of the swimmers' arms.

Having to use 8-16 cameras infringes criterion 1—speed—from Section 2.2.1. It takes four to six hours to set up eight cameras underwater, between placing the cameras in waterproof housings, mounting them on tripods, finding the best location for each camera (potentially depending on the length of the cables), carefully putting them underwater at the desired depth, focusing them, and calibrating them. Furthermore, unlike for land-based laboratories in which cameras might only need to set up once and re-calibrated before each use, it is not possible to leave cameras permanently in a public swimming pool, meaning the system would need to be set up and dismantled each time it was to be used. Furthermore, a 16-camera system (which ultimately is the recommendation given by Mündermann et al.) would be costly: given that

during the course of this PhD only eight cameras were available and that each camera + waterproof housing costs roughly £2,500[2], buying enough equipment to satisfy the recommendations of Mündermann et al. would have cost over £20,000.

Another aspect analysed by Mündermann et al. was how to arrange the cameras around a person to obtain the most accurate visual hull possible. Their results showed that the best camera arrangements are a hemisphere or circle centered around the person (with a hemisphere leading to slightly better results for # cameras > 16). However, all their tests were conducted with the participants standing upright. Therefore, when they refer to a 'circular' arrangement of cameras they mean that the cameras are to be placed in a circle that lies on a plane parallel to the transverse plane of the person's body. If such an arrangement was to be used to reconstruct the visual hulls of swimmers (who are not upright, but horizontal), the cameras would need to be arranged in a 'Ferris wheel' fashion, which would be impossible due to the limited depth of swimming pools (most pools in the UK are no deeper than 2 metres). A hemisphere arrangement of cameras would similarly be impossible to set up in a swimming pool, leading to the conclusion that the only camera arrangements possible in a swimming pool would be sub-optimal for the reconstruction of visual hulls of swimmers.

One aspect related to the most optimal arrangement of cameras that Mündermann et al. did not investigate is the relationship between visual hull accuracy and distance between the cameras and the object. Intuitively, the further the cameras are from the object, the more the rays connecting the camera to the object's silhouette become tangential to the silhouette; the more tangential the rays are to the silhouette, the less 'exogenous' volume is included in the cone defined by the rays, which is the constraint used to define the visual hull. As was discussed when describing the flaws of depth-based markerless systems, a direct relationship between the accuracy of the reconstruction and the distance between the camera and the object can be problematic for sports motion capture, and in particular for swimming. Since the cameras would be static and the swimmer would swim through the capture volume, during the movement the distance between the swimmer and each camera would vary, and in particular it would vary

[2]The price refers to the high-resolution cameras used during this PhD. Cheaper cameras could be bought, lowering the overall cost of such a system by an amount that would depend on the type of cameras bought.

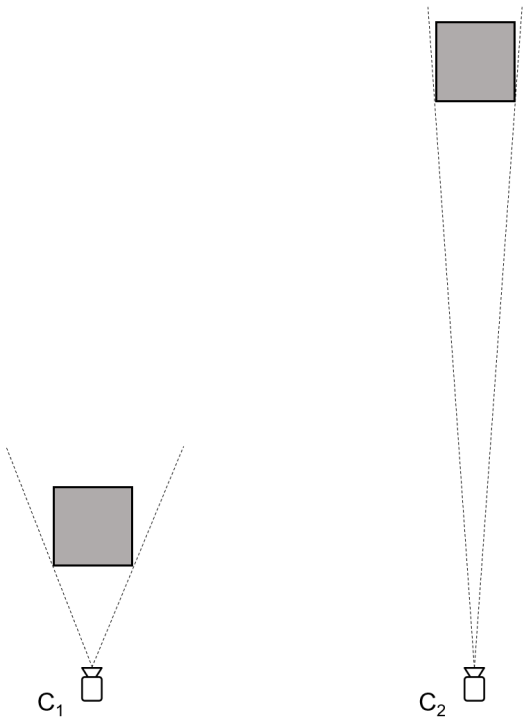


Figure 3.6: This example seems to—at least qualitatively—confirm my theory that the distance between the cameras and the object influences the accuracy of the visual hull. (Image created by the author)

in a different way for each camera. This means that the accuracy of the reconstructed visual hull would not be stable over the duration of the motion of the swimmer.

Another weakness of the visual hull is that it struggles to reconstruct concave parts of objects, as shown in Figure 3.7. This weakness is likely to have been another source of error for Ceseracciu et al.'s model, which focused on the arm. Indeed, the shoulder (the arm joint most prone to creating concave areas) was the most inaccurate joint in Ceseracciu et al.'s model. This weakness in dealing with concave areas is greatly mitigated by having access to more viewpoints, explaining why having more cameras leads to more accurate visual hulls. It also explains why Mündermann et al. found that arranging cameras in a hemisphere is slightly better than arranging them in a circle, since having cameras on multiple planes increases the chances that at least one of them would be perpendicular to a concave area that was invisible to all others.

So far we have assumed that the visual hull itself is used as an estimation of the shape and pose of an object. Indeed, for most inanimate objects this is usually the case [206–209].

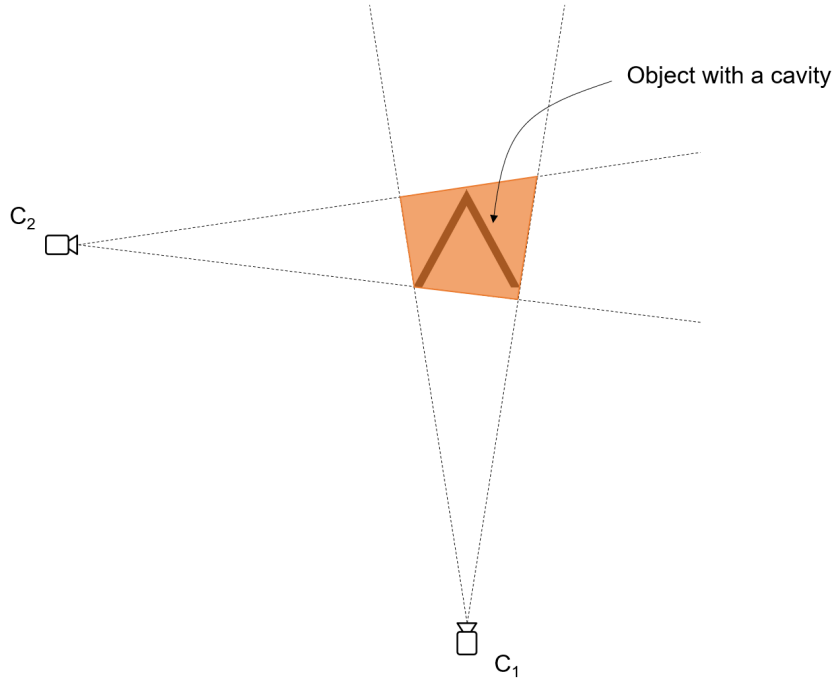


Figure 3.7: In this 2D example, the cameras are co-planar with the object (a simple V shape) and are therefore incapable of reconstructing the cavity within it; the resulting visual hull is a gross overestimation of the true volume of the object. No matter where on the 2D plane outside of the object new cameras are put, the cavity will not be detected. However, if we imagine placing a camera perpendicular to the 2D plane on which the object lies, the cavity would be detected perfectly. (Image created by the author)

However, as mentioned in Section 2.3.2, when visual hulls are used to reconstruct the shape and pose of a person they can be used in conjunction with a parametric model. In fact, it is only in conjunction with a parametric model that the visual hull becomes a truly generative method. Recall that in Section 2.3.2 generative methods were defined as fitting a generic, articulated model to certain constraints. The visual hull can be used as one such constraint: we can iteratively scale a parametric model and change its pose so that it fits tightly inside a reconstructed visual hull. This was the pipeline used by Corazza et al. [22], whose algorithm is, to date, one of the most accurate for markerless motion capture. However, for swimming motion capture the visual hull would need to be reconstructed with fewer than eight cameras (Corazza et al. used eight) placed in a sub-optimal arrangement (Corazza et al. used a circle arrangement). This means that the reconstructed visual hull would likely be less accurate than the ones reconstructed by Corazza et al. (who also had a perfectly monochromatic and stable background), and therefore it would be a looser constraint. This in turn means that

the optimisation algorithm needed to fit a parametric model to it might not be able to reach a global optimum.

3.4 Conclusions and Departure From the Visual Hull

Given the limitations of the visual hull and the difficulty in using it as a constraint for a generative method, it is not surprising that the most recent literature on 2D-to-3D models has shifted from using visual hulls as constraints to using the location of 2D joints. Indeed, this type of constraint allows for a more streamlined optimisation: the 3D joints of the parametric model can be projected onto the plane of each camera and 'anchored' to the 2D joints, quickly defining the correct pose of the person; then, the silhouette from each camera can be used to fine-tune the model and give it the correct shape. This type of generative model does not rely on the number of cameras available as strongly as does a generative model that uses the visual hull as a constraint: having access to the ground truth location of each joint in each camera plane is a much stronger constraint than a visual hull, and therefore it is possible to solve the optimisation algorithm with fewer data points (i.e. cameras). For this reason, the best option for swimming motion capture is an image-based generative method that uses silhouettes and 2D joint labels as constraints to optimise a parametric model. Such an algorithm consists of four parts: a silhouette extraction algorithm; a 2D pose detection algorithm; a parametric model; and an optimisation algorithm that is able to adapt the parametric model to the silhouettes and 2D joints. When developing such a generative algorithm for use in swimming motion capture, the greatest challenge lies in the development of the silhouette extraction and 2D pose detection algorithms. Parametric models like SMPL [64] can be downloaded for free[3], and optimisation algorithms, while complex, are not influenced by the nature of the input images: as long as the silhouettes and 2D joints are accurate, an optimisation algorithm performs equally well on any kind of image, regardless of background, lighting, pose, and any other factor. The same is not true for silhouette extraction and 2D pose detection algorithms: since they would need to be based on neural networks (which vastly outperform non-learning algorithms), these algorithms

[3] https://smpl.is.tue.mpg.de/

would need to be trained on large datasets of images of swimmers, and they would need to be optimised to this domain. For instance, most silhouette extraction and 2D pose detection algorithms expect RGB images as inputs, but the cameras that were available for this PhD project were greyscale. Therefore, I decided to make the development of a silhouette extraction algorithm and of a 2D pose detection algorithm that would work well on images of swimmers the main focus of this PhD, leaving to future research the implementation of an optimisation algorithm that fits a parametric model to these constraints.

Chapter 4

Construction of Scylla, a Dataset of Images of Swimmers for Silhouette Extraction

4.1 Preliminary Considerations

There are three factors to consider when constructing a dataset for training deep learning models: 1) exactly what the contents of the images should be; 2) how many images are needed; and 3) how the images are going to be labelled; this last factor will be discussed in Section 4.3. Factor 1 may seem trivial (for example, one might address it simply by saying 'the Scylla dataset needs to contain images of underwater swimmers'), but care needs to be taken that the images gathered fully describe the data-generating process[1] that algorithms are supposed to model. In particular, it is important that the dataset contains examples of all the types of variation in the data that an algorithm is expected to learn to distinguish (which is a re-formulation of the NFL theorem discussed in Section 2.3.2). For example, if a dataset was built using images of swimmers recorded only at night, algorithms trained on the dataset would struggle to perform

[1] In the case of the Scylla dataset, the data-generating process is the act of extracting the silhouette of a swimmer given an image as input.

well when shown images taken during the day. This is an example of dataset bias[2] [211], which is a phenomenon that may happen without the authors of the dataset realising it (as illustrated by the 'Neural Network Tank' urban legend[3]). Section 4.2 will describe what steps I took to ensure that the Scylla dataset contained minimal bias.

Factor 2 is also difficult to address: there is no way to know a priori how much data a neural network will need to reach a given target for a given metric—not only because neural networks use different metrics depending on their intended application, but also because power analysis tools [212] used for simpler models do not apply to the complex (and often 'black-box') functions that neural networks learn [187]. Some authors [9,10] have proposed that researchers should perform an 'inverse power analysis' (IPA) by training their algorithms on increasing amounts of data and testing them on a separate, previously acquired test set. If the results of these tests are plotted (with the amount of training data on the x axis and the performance of the algorithm on the test set on the y axis, as in Figure 4.1) and a plateau is observed, the x value corresponding to the onset of the plateau represents the amount of data that are *necessary* for the algorithm to perform adequately; any data in excess of that amount will not provide substantial benefits. If a plateau is not observed, it means that the algorithm needs to be trained on more data to reach optimal performance. Researchers can then continue to perform IPA until a plateau is reached. Unlike power analysis, therefore, IPA does not provide an a priori estimate of the sample size required to achieve a certain performance, but it does provide a way to estimate if the amount of data available is sufficient for the algorithm to perform optimally, and, more importantly, to estimate whether gathering more training data is likely to increase the performance of the algorithm substantially. In the example of Figure 4.1, if only 4,096 training examples had been gathered and used to perform IPA, and the performance of the model was sill unsatisfactory, the IPA graph would show that gathering more training data would likely not be as beneficial as changing the design of the algorithm.

[2] An example—very pertinent to current events—of the danger of dataset bias is the recent scandal concerning facial recognition algorithms (some of which were used by US police departments) which failed to recognise the faces of Black, Middle Eastern, and Latino people more often than those of Caucasian people [210]. This does not mean that those algorithms are inherently discriminatory against people of those races: it likely means that the dataset was biased towards Caucasian people—for example, because it had more examples of Caucasian people than of any other race.

[3] https://www.gwern.net/Tanks

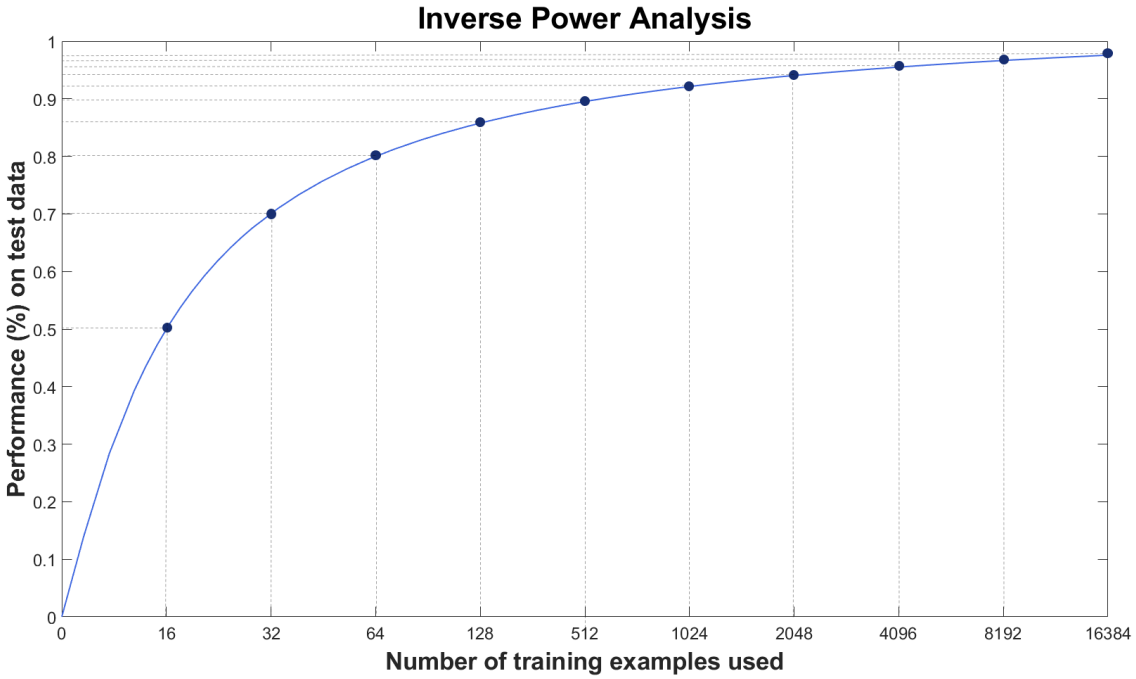


Figure 4.1: To perform the 'Inverse Power Analysis' method proposed by [9] and [10], an algorithm needs to be tested on increasingly large amounts of data, and its performance plotted—for each test—against the amount of data used. Note that the x axis is in powers of two (as indicated by [9]), so the plateau is much flatter than it appears to be. (Image created by the author)

However, IPA has two main limitations. First, it relies on the assumption that a test set is available prior to the analysis, which means that IPA cannot provide an answer to the question of how large the test set should be. A naïve answer could be 'as large as the available resources allow it to be', but how can we decide how many resources we can allocate to collecting and labelling the test set if we have no idea how large a training set we will need? Therefore, IPA only solves half the problem: *after* it has been established how much test data is needed (using some other heuristic), and all the test data and at least some training data have been collected and labelled, IPA can give an estimate of how much more training data will be needed. The second limitation of IPA is that it needs to be repeated every time the algorithm is changed, since different algorithms have different requirements in terms of the amount of training data needed to reach optimal performance. This creates a bottleneck in the development of the algorithm: each time a new version of the algorithm is developed, further development needs to be halted until IPA has been repeated. This bottleneck can be crippling if it takes a long time

to perform IPA—which, especially for large (in terms of the number of parameters they have) models trained on large amounts of data, is often the case. For example, training the FISHnet algorithm (which is not particularly large, at 28 million parameters) described in Chapter 5 on the full training set of the Scylla dataset (which is relatively small) takes about 24 hours; since IPA involves training an algorithm multiple times using increasing amounts of data, performing IPA on FISHnet may take several days.

Nevertheless, IPA should not be discarded altogether: if an algorithm has undergone several stages of development and is still not performing as desired, IPA can inform researchers of whether this is due to a lack of training data, information which could save time that otherwise might have been spent fruitlessly debugging or fine-tuning the model. However, IPA cannot be used as a tool to decide how much data will need to be gathered before any of them are available, and how to split them between training and test sets (though the general guidelines for deep learning models are that test sets should comprise 10-20% of the total data available [187]). The best heuristic available is to consider the size of existing datasets in the same field, as well as the general opinion of the research community on whether those datasets are appropriate for the task, or if they are used just because better options are not available.

For silhouette extraction, the most established datasets are PASCAL VOC, which contains almost 10,000 images with labels for segmentation, and MS COCO, which contains 200,000 images with labels for segmentation (see Section 2.4.2.3). Given that PASCAL VOC is used as much as (if not more than) MS COCO, it is reasonable to assume that about 10,000 images are sufficient to train semantic segmentation algorithms that generalise well. In other words, PASCAL VOC likely describes the true data-generating process well enough for researchers to assume that algorithms that perform well on its test set should perform well on never-before-seen images. It might seem fair to say, then, that the Scylla dataset needs to contain about 10,000 images to make algorithms trained on it generalise as well as those trained on PASCAL VOC. However, the data-generating process that PASCAL VOC attempts to describe is the extraction of silhouettes of 20 types of objects (many of which could be present in the image at the same time) from RGB images, while Scylla needs to describe a simpler data-generating process: the extraction of silhouettes of one class of objects (only one of whom would be in any

given image) from greyscale images. Therefore, the number of images needed to capture the variability of the data-generating process that PASCAL attempts to model is greater than that needed to capture the variability of the data-generating process that Scylla needs to model. This means that Scylla likely needs fewer than 10,000 images to fully describe its target data-generating process. Therefore, I estimated that roughly 3,000 images (between the training set and the test set) would be sufficient for algorithms trained on Scylla to generalise well. This estimate was reached also considering the time and financial constraints of this project, since, as will be discussed in Section 4.3, labelling the silhouettes of large images of swimmers is time-consuming.

4.2 How the Data were Gathered

To construct the Scylla dataset I had at my disposal a collection of about 400 videos recorded before this PhD for a different project; these videos will be referred to as 'The Collection'. The videos in The Collection feature 14 highly trained (5+ years experience) adult swimmers (five female, nine male). The videos were recorded on three separate days in the Olympic-length swimming pool in Loughborough. To record the videos, either a one-, four-, or eight-camera setup was used; all cameras were greyscale. All recording sessions lasted several hours, meaning the lighting conditions changed during the session. Swimmers were recorded (one at a time) performing either underwater butterfly kicking or the breaststroke pull-out; in most cases, the same swimmer performed both motions several times.

Since from each video several hundred frames could be extracted, there is no doubt that The Collection could provide the 3,000 images that I estimated should be in the Scylla dataset. However, factor 1 described in Section 4.1 remains to be addressed: are the videos in The Collection descriptive enough of underwater swimming to avoid dataset bias? To answer this question, we need to identify the main sources of variability against which we would want algorithms to be invariant, and determine to what extent they are represented in The Collection. As humans, we would group the sources of variability of the videos in The Collection in two

categories:

- Demographic variability; this relates to factors such as the age, height, or weight of participants.
- Recording variability; this relates to the conditions in which the images were recorded, such as the position of the camera relative to the swimmer, the shutter speed of the camera, its exposure, etc.

However, from the perspective of a neural network, sources of variability are classified into these two categories:

- If the pixel values of two labels (i.e. silhouettes) are not identical, the corresponding two images are treated as two separate, unique training examples, *regardless of whether they feature the same person*; I call this type of variability 'Different Image, Different Label' (DIDL) variability;
- If the pixel values of two labels are identical, but the pixel values of the corresponding two images are not, the two images are treated as two separate, unique training examples, *regardless of whether they feature the same person*; I call this type of variability 'Different Image, Same Label' (DISL) variability.

The only aspect of demographic variability that could meaningfully impact DISL variability is skin tone: there is no other demographic factor that could cause two images to have the same labels but different pixel values. In this regard, The Collection can be said to be biased towards Caucasian swimmers, since all 14 swimmers featured in its videos are Caucasian. Given that there currently are no elite swimmers in the UK who are non-Caucasian, addressing this type of dataset bias of The Collection by recording videos of people with dark skin tones was not feasible within the scope of this PhD. This source of bias should be addressed by future work, by recording (and labelling) videos of swimmers with dark skin tones.

Recording variability clearly has a greater impact on DISL variability: depending on how, where, and when the cameras were set up (as well as what cameras were used), the images

might be more or less in focus, bright, or distorted by motion blur—all conditions that would affect the pixels of the image but not those of the label. Recording variability also explains a greater portion of DIDL variability than does demographic variability, especially in the context of elite swimming. If we used the same camera (with the same settings) to capture an image of two swimmers standing in the same position at the same distance from the camera, the two silhouettes extracted from the images would differ due to the differences in body shape of the two swimmers. The main source of DIDL variability would be a difference in the height of the two swimmers: though we would not expect two elite, able-bodied swimmers to have wildly different body shapes, their different heights would lead to silhouettes of different sizes. However, a similar size-related variability could be achieved by changing the distance of the swimmer from the camera. Indeed, recording variability enables greater variations of the shape and size of silhouettes than demographic variability: from the same swimmer we could get images whose silhouettes vary a lot, simply by placing the camera at different angles and distances, or by making the swimmer assume different poses.

Therefore, we can conclude that recording variability, more than demographic variability, is what needs to maximised in the Scylla dataset. Because the videos in The Collection were recorded under varied conditions, one would expect the recording variability, and consequently both DIDL and DISL variabilities, to be high. However, DIDL variability might be lowered by the process of extracting still frames (of which only 3,000 were needed for the Scylla dataset) from the videos of The Collection (which, together, contain tens of thousands of frames): if N consecutive frames were extracted from a video, they would share too much information with one another and reduce the total amount of DIDL variability in the dataset. Likewise, if one frame every N was extracted, bias might inadvertently be introduced into the dataset: since swimming is a cyclic motion, it is possible that extracting one frame every N could accidentally 'synchronise' the extraction of the frames with a movement cycle, thus extracting frames of swimmers who are always in similar positions. Therefore, I decided to extract from each video a number of frames that varied semi-randomly: at least ten frames per video were extracted, and a gap of at least X frames was kept between any two consecutive frames (where X was randomly assigned a new value between 10 and 30 whenever a frame was extracted).

This scheme led to the extraction of over 5,000 frames from The Collection. Of these, several hundreds had to be discarded due to sub-par image quality (in most cases due to blurring, or excessive turbulence of the water) or to the presence of multiple swimmers in the same image (for example, because a swimmer would stay at one end of the pool while another performed a trial). After removing all such frames, 3,100 HD greyscale images of underwater swimmers remained. Of these 3,100 images, 2,793 had a resolution of 2048 x 900 pixels, and 307 had a resolution of 1920 x 1080 pixels. To make all images have the same resolution (which is a requirement of deep learning models), instead of resizing the images I decided to pad them with black pixels until they had a resolution of 2048 x 1088 pixels. Resizing images would cause distortions which would also be present in the silhouettes. Since this type of distortion is absent in nature, algorithms should not be invariant to it. Padding images, on the other hand, does not alter the label in any way, and is therefore likely to be a better option. Finally, the reason I padded the images to the specific resolution of 2048 x 1088 is that this resolution does not require extensive padding of the images, but still makes both axes divisible by two at least six times. This is a property that makes training neural networks easier[4], since most of them have components that halve the dimensions of the input up to about six times. Finally, the images were split into two sets: 2,635 for training, and 465 for testing. Swimmers who appeared in the test images did not also appear in the training images.

4.3 How the Data were Labelled

To label the images in the Scyllla dataset, an expert PhotoShop user[5] was employed to manually trace all 3,100 images using PhotoShop. The guidelines given to the labeller were as follows:

- Pay close attention to the face, which often is masked by bubbles;
- Do not pay extra attention to finer details like the nose and the fingers, since they contribute very little to the overall shape and size of the silhouette; this is not to say that

[4]This is a property mainly of convolutional neural networks, which are the types that are used for computer vision problems.

[5]This person, who is an architect by profession, has eight years of experience using PhotoShop.

details should be overlooked;

- When in doubt regarding where the contour of the silhouette lies, it is preferable to overestimate it than to underestimate it[6];
- After having labelled an image, superimpose the silhouette to the image to verify that there are no gross mistakes.

The labelling process lasted four months, with a typical time per image of about six minutes. All silhouettes were then visually inspected, and images whose silhouettes were insufficiently accurate were sent back to be labelled again. Figure 4.2 shows an example of an image and segmented silhouette from the Scylla dataset.

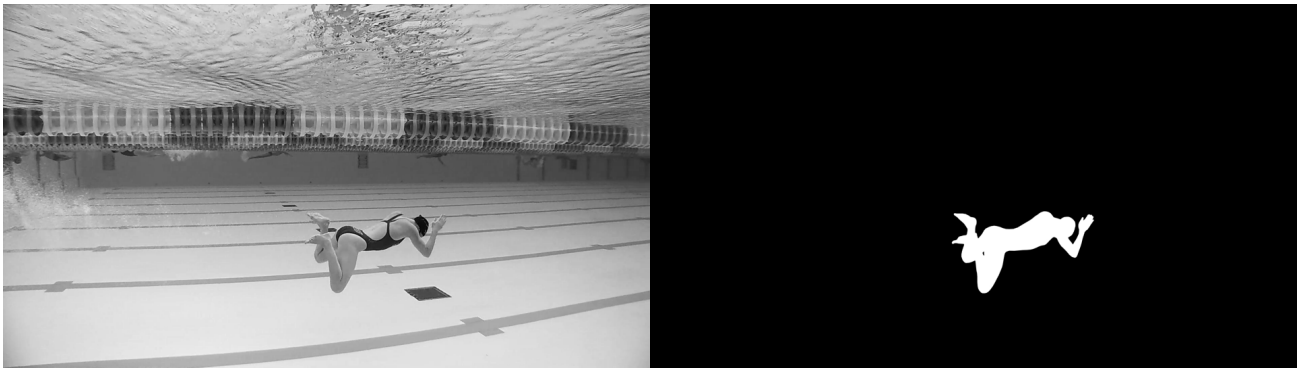


Figure 4.2: On the left: an image from the Scylla dataset. On the right: the corresponding silhouette, traced by hand by the labeller. (Image created by the author)

To quantify the inter-operator reliability of humans (against which to compare the performance of future algorithms) in this labelling task, two more expert PhotoShop users were employed to label ten of the images that had been labelled by the first operator. The performance of the labellers (in pairs) were then compared by calculating the mean Dice score (which is another name for the F-score described in Section 2.4.1.4) of the silhouettes they traced. The Dice score was chosen because it is easy to interpret: it is a harmonic mean of precision and recall, and it takes values from 0 to 1, with 1 indicating that two silhouettes overlap completely and 0 indicating that two silhouettes do not overlap. The results of this analysis are reported in Table 4.1.

[6]Since the silhouettes are going to be used as constraints for a generative model, it is preferable for the constraint to be looser than stricter, since too strict a constraint might make it impossible for the optimisation algorithm to converge.

Table 4.1: Inter-operator Reliability of Silhouette Extraction by Humans

Labeller ID	Dice score
1 and 2	0.9422
1 and 3	0.9372
2 and 3	0.9509

These results justify the choice to depart from the visual hull: since even humans incur at least a 5% error when labelling the silhouettes of underwater swimmers, the resulting visual hull would almost certainly be quite poor (see Figure 3.5).

4.4 Conclusions

It is difficult to estimate a priori how many images are needed to construct a dataset for deep learning algorithms. Though some authors suggest the use of Inverted Power Analysis to estimate the number of training images needed for a specific algorithm to perform adequately, this tool is time consuming and only addresses half the issue, since it cannot give any estimate of how large the test set should be. The best heuristic available for determining the required size of a dataset, then, is to look at the size of established datasets that address a similar data-generating process. For the case of the Scylla dataset, this means looking at the most commonly used datasets for silhouette extraction, MS COCO and PASCAL VOC. Given that these datasets attempt to describe a data-generating process that contains much more variability than would need to be described by Scylla, I estimated that about 3,000 images would be sufficient to train a deep learning algorithm to segment the silhouettes of underwater swimmers.

From the videos of The Collection, I extracted 3,100 frames, ensuring that the interval at which successive frames were extracted from a video did not introduce bias. All 3,100 images were then labelled by hand by an expert PhotoShop user. The results of the experiment on human reliability, reported in Table 4.1 testify to the difficulty of the task. The following section will describe how the Scylla dataset was used to train a novel silhouette extraction algorithm, FISHnet.

Chapter 5

Development of FISHnet, a Silhouette Extraction Algorithm for Images of Swimmers

(*Sections 5.3 and 5.4 were published in revised form in the following* IEEE Access *paper:* [213].)

5.1 Initial Development on the Carvana Dataset

To be able to start developing an algorithm while the images of Scylla were being labelled, I decided to adopt the Kaggle Carvana dataset as a proxy for Scylla. The Kaggle Carvana dataset consists of 105,152 images (split into 5,088 for training and 100,064 for testing) of cars, and was built as part of a public competition (hosted on Kaggle, where the dataset can be downloaded[1]). Though its images feature cars and not swimmers, the Kaggle Carvana dataset has characteristics that make it a suitable proxy for Scylla:

- Like Scylla, it is focused entirely on the segmentation of one class of object. Therefore, algorithms do not need to spend capacity learning to distinguish different types of objects;

[1] https://www.kaggle.com/c/carvana-image-masking-challenge

- In any given image, only one car appears, and the car is always fully visible. This simplifies the function that neural networks need to learn, since they will not need to spend capacity learning partial segmentations (i.e. segmentations of objects that are only partially in the image) or segmentations of overlapping objects;
- The resolution of the images is 1280 x 1918. Cityscapes [158] is the only other public dataset for silhouette extraction that has images with resolution over 640 x 640.
- The ground truth images were labelled more accurately than those in either PASCAL VOC or MS COCO; an example of this is shown in Figure 5.1;
- The test set is large enough to guarantee protection against overfitting;
- Since it was part of a Kaggle competition with a cash prize of $25,000, the dataset comes with a leaderboard that was likely quite competitive, making it an excellent reference against which to compare the performance of the algorithms developed.

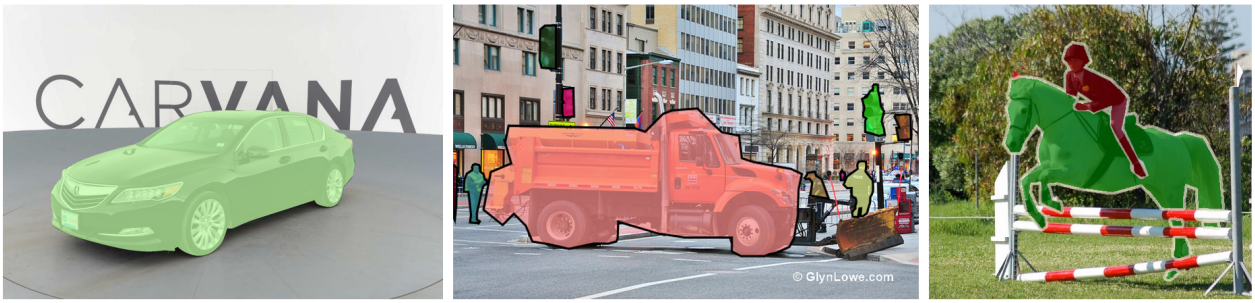


Figure 5.1: Example of the quality of annotation in the Kaggle Carvana, MS COCO, and PASCAL VOC datasets.

To establish a baseline from which to develop FISHnet, I decided to re-implement U-Net [111] and train it on the Kaggle Carvana dataset. U-Net is easy to implement by hand, since its architecture is so simple. Originally developed for biomedical image segmentation, U-Net (Figure 5.2 consists of an encoder composed of a series of blocks made of convolutional, batch normalisation, and max pooling layers, followed by a symmetrical decoder in which the max pooling layers are replaced by bilinear upsampling layers. Each block in the encoder is then connected to the corresponding block in the decoder via skip connections. The symmetry of

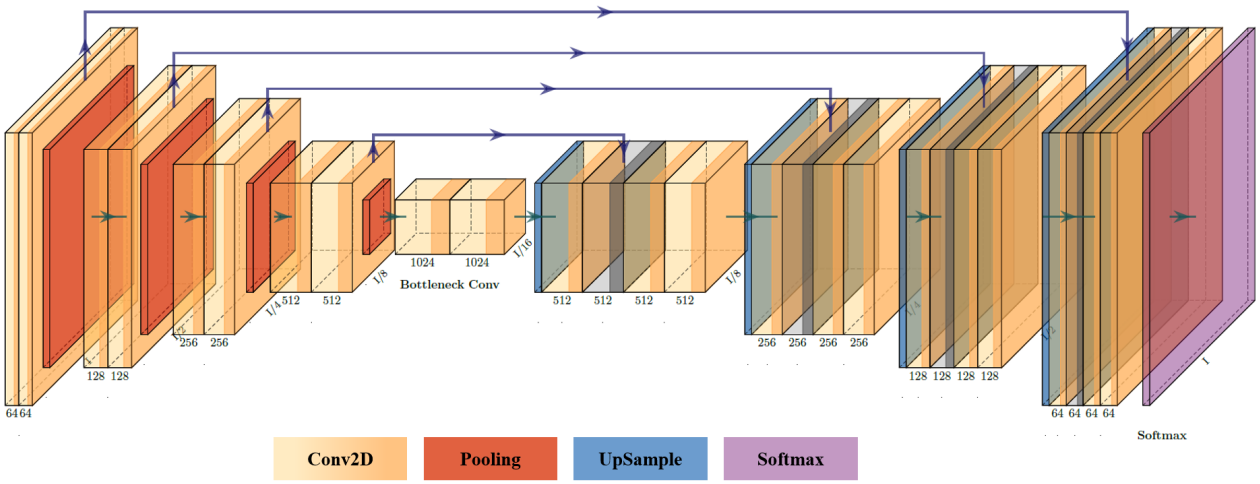


Figure 5.2: Architecture of U-Net

U-Net allows it to output predictions at the same resolution as the inputs, a feature which many silhouette extraction algorithms do not have. For example, DeepLabv3+ outputs predictions that are 16 times smaller than the inputs[2]. Since the final evaluation is performed on the full-sized test images, having the network output predictions of the same size as the input is likely to be an advantage. A second advantage of U-Net is that it can easily be scaled to be deeper or shallower by modifying the input resolution and adding more blocks to the encoder and to the decoder. For example, U-Net 128 is a U-Net network with 128 x 128 inputs and four blocks in the encoder and decoder. To make the network deeper, the input can easily be changed to be 256 x 256 and use five blocks in the encoder and decoder; this network would take the name of 'U-Net 256'.

To establish a baseline, then, U-Net 128, 256, 512, and 1024 were trained and tested on the Kaggle Carvana dataset. Since the largest batch size with which U-Net 1024 could be trained on the GPU available was two, all U-Nets were trained with this batch size, to standardise the results. For a loss function, the weighted Dice loss (proposed by [214], who won the Kaggle Carvana competition) was used. The weighted Dice loss is calculated as:

$$Weighted\ Dice\ Loss = BCE + 1 - Weighted\ Dice \tag{5.1}$$

[2]Very deep models like DeepLabv3+ cannot be trained with full-sized outputs because either they would not fit in most GPUs or they would take too long to train.

where the Weighted Dice score is obtained by giving pixels along the contour of the silhouette a weight three times larger than is given to pixels deep within the silhouette; and BCE is the per-pixel binary cross-entropy loss function [187], which is calculated as:

$$Binary\ \ Cross\text{-}Entropy = \frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(\hat{y}_i) + (1 - y_i) \cdot log(1 - \hat{y}_i) \tag{5.2}$$

where N is the number of samples, y_i the ground truth for the i^{th} sample, and $\hat{y}_i$ the prediction for the i^{th} sample.

Table 5.1: Dice Score of Various U-Nets on the Kaggle Carvana Dataset

Model	Dice score
U-Net 128	0.9895
U-Net 256	0.9927
U-Net 512	0.9951
U-Net 1024	0.9960

Table 5.1 shows that even U-Net 128 achieved a Dice score over 0.98[3] (even though it would place 502$^{\text{nd}}$ on the Kaggle Carvana leaderboard). This indicates that the number of images in the training set was adequate, or a model as small as U-Net 128 would have overfit. It also indicates that the complexity of the data-generating process that the dataset describes is low, or a model as small as U-Net 128 would have had a lower Dice score on both the training and the testing set. Nevertheless, U-Net 1024, the deepest baseline model tested, would place 337$^{\text{th}}$ on the Kaggle Carvana leaderboard, indicating that there is room for improvement.

Taking U-Net 1024 as a baseline, modules from state-of-the-art semantic segmentation algorithms were added to it—starting with the Atrous Spatial Pooling Pyramid (ASPP) module, which is one of the defining modules of DeepLab [140, 142]. DeepLab's ASPP module, which is placed towards the end of DeepLab's encoder, consists of a variable number of convolutional layers (stacked in parallel) with increasing dilation rates (d). By stacking these layers in

[3]Could this result already be considered sufficient for the application? Potentially, but there is no obvious answer this question, as the literature suggests no thresholds of acceptable accuracy. Nevertheless, given that in Chapter 3 it was shown how 2D-to-3D rely on the silhouettes being extremely accurate, it is fair to suggest that silhouette extraction accuracy should be as high as possible with the hardware available.

parallel, the ASPP module grants the encoder a variable receptive field and enables it to identify objects at different resolutions. For example, adding a convolutional layer with $d = 2$ doubles the receptive field of the encoder, while using $d = 4$ grants it four times as large a receptive field; by using both in parallel, the encoder can decide which one to use depending on the situation. I implemented the ASPP module within U-Net 1024 by placing six convolutional layers with increasing dilation rate (2, 4, 8, 16, 32, 64) after the bottom block of the encoder, and obtained a Dice score of 0.9970—a 0.1% improvement over the baseline. To investigate whether the structure of the ASPP module—rather than the additional capacity it introduces—was responsible for this improvement, a version of U-Net 1024 with an ASPP module in which all the convolutional layers had the same dilation rate (which was set to 16) was also tested. This modified ASPP module still allows the bottom layers of the encoder to recover some spatial resolution, but it does not allow them to recover different resolutions simultaneously, which is the main purpose of the ASPP module. On Kaggle Carvana's test data, U-Net 1024 with a modified ASPP module achieved a Dice score of 0.9968—0.02% worse than the score obtained by using the original ASPP module. This result indicates that the main contribution of the original ASPP module to U-Net 1024's performance was not the ability to recover different resolutions simultaneously, but the increased capacity that comes with adding the dilated convolutional layers. This is not a surprising result. In DeepLab, in which the output resolution is 16 times smaller than that of the input, the function of the ASPP module is critical: it allows DeepLab to 'pretend' to have a larger output resolution than it actually has. In U-Net, this feature is not needed, since the output resolution is already the same as the input resolution.

Improving the connectivity between different parts of convolutional neural networks has been found to be beneficial on most computer vision tasks, and is the reason for the success of ResNet [98] and DenseNet [12], two of the most popular architectures in deep learning[4]. The rationale behind improving intra-network connectivity is that the more interconnected the network, the more freely information can be shared between different parts of the network. In encoder-decoder networks like U-Net, this feature is essential. The max pooling layers of the encoder encode the image into progressively higher-level features, which have progressively lower

[4]As of the time of writing, the two papers that introduced ResNet and DenseNet have been cited, combined, in 76,162 published papers.

resolution and higher semantic content; then, the features of the deepest layer of the encoder are upsampled (or 'decoded') until their resolution matches the resolution of the input. The features of the decoder are, therefore, the highest-level features of the encoder, upsampled once at each layer of the decoder. As the upsampling operation does not restore spatial resolution, the features of the decoder lack spatial resolution at all depths. The purpose of the skip connections in encoder-decoder networks, then, is to pass some of the spatial resolution information present in low-level features of the encoder to the high-level features of the decoder. For example, the features at depth $D = 0$ of the encoder (maximum spatial resolution, minimum semantic meaning) are connected with the features at depth $D = 0$ of the decoder (minimum spatial resolution, maximum semantic meaning), which is the layer most responsible for the final segmentation and which, therefore, is most in need of fine-grained spatial resolution.

The authors of ExFuse found that if more semantic information is embedded in the low-level features and more spatial resolution is embedded in the high-level features, the effectiveness of the skip connection is magnified. The rationale is that if the two blocks contain incompatible information (both in terms of semantic content and spatial resolution), passing information between them is difficult and the weights of the skip connection might decay [11]. As ExFuse is, along with DeepLab, one of the most innovative and high-performing algorithms for semantic segmentation available, I decided to implement some of its intuition into my own network. To embed more semantic information into the low-level features of the encoder, the authors of ExFuse use three strategies:

- They re-arrange the layers of their encoder—which is either ResNet50 or ResNeXt101, both of which have blocks with a varying number of convolutional layers; this solution is not applicable to U-Net 1024, since U-Net 1024 uses blocks with a constant distribution of convolutional layers;
- They introduce intermediate losses after each block of the encoder. This strategy has been shown to be beneficial for very deep models (> 100 layers) such as ResNet and DenseNet, but some authors have shown that they are ineffective for neural networks with under 100 layers (U-net 1024 has 96) [152, 215, 216];

- They introduce a new module, called Semantic Embedding Branch (SEB), shown in Figure 5.3. The SEB module adds to each skip connection an additional connection from each block of the encoder below the current block. For example, if a skip connection is being performed between blocks of the encoder and decoder at depth D, the outputs of the blocks of the encoder at depth $D-1$, $D-2$, $D-3$, etc., are fused to the output of the block of the encoder at depth D before being skip-connected with the block at depth D of the decoder.

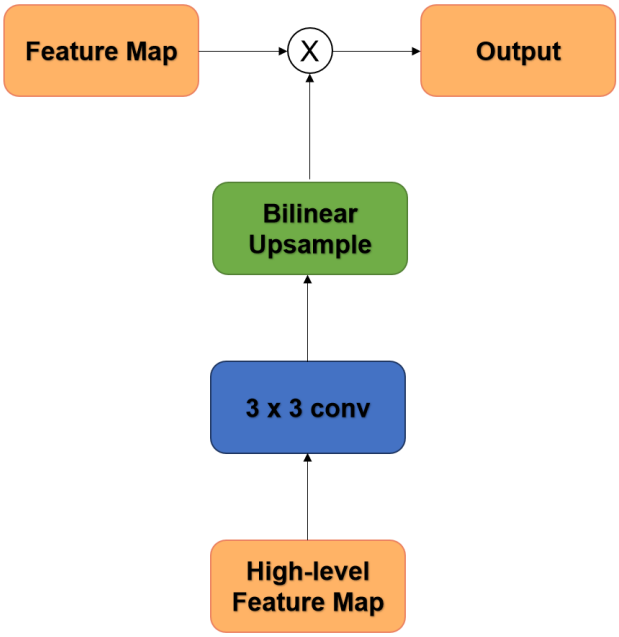


Figure 5.3: The SEB module from ExFuse. The '×' sign represents element-wise multiplication. If multiple high-level feature maps are present, they are all multiplied together (element-wise). (Image created by the author)

To embed more spatial resolution into the high-level features of the decoder, they use two strategies:

- They upsample the deepest block on the encoder until it has the same resolution as the inputs, and then apply an intermediate loss to it. According to the ablation study they performed, this additional loss increased the IoU (which is a metric related to the Dice score and which also goes from 0 to 1) of their algorithm on PASCAL VOC by 0.5%, which is considered a substantial improvement. However, the exchange of information between the deepest block of the encoder and this heavily upsampled loss is as difficult as

that between the encoder and decoder blocks at depth 0. If the authors of ExFuse argue that this type of exchange of information is not optimal and needs to be modified, I would argue that the same logic applies to this intermediate loss that they propose. Therefore, I believe that the addition of this loss, like the addition of the other intermediate losses that ExFuse uses in the encoder, might have contributed to performance simply because it made the optimisation of their very deep network easier, not because it introduced more spatial resolution in the high-level features of the encoder;

- They modify the way in which the network outputs predictions by adopting a mechanism they call 'Densely Adjacent Prediction' (DAP). Regular decoders output a feature map in which each point assumes a value between 0 and 1, which is the probability that that pixel belongs to the background or to the foreground. Therefore, each point of the feature map is responsible for the prediction of the value of just one pixel of the input image. The DAP mechanism makes each point of the feature map also predict the values of its neighbouring pixels. This leads to every pixel having multiple predicted values, which are then averaged to obtain one value per pixel. The downside of the DAP mechanism is that it multiplies the number of feature channels of the output by a number equal to the size of the 'neighbourhood' that each point of the output is responsible for. In the original implementation of ExFuse, this meant that adding the DAP mechanism increased the size of the output feature map by nine times. Due to hardware restrictions, it was impossible for me to embed the DAP mechanism in U-Net 1024.

The SEB module, then, was the only part of ExFuse that could be implemented in U-Net 1024 that I thought would improve its performance. After re-training the network, U-Net 1024 with the SEB module achieved a Dice score of 0.9963. This indicates that the SEB module is effective, but not as effective as it was in ExFuse.

Next, DeepLab's ASPP module was re-purposed to align it with ExFuse's goal of strengthening the skip connections. In the original implementation of the ASPP module within U-Net 1024, it was placed after the bottom block of the encoder, where the features have the lowest resolution and the highest semantic content. In DeepLab this is a sensible placement for the

ASPP module, but, as indicated by the results reported earlier in this section, it is not in U-Net. What ExFuse shows is that in a U-Net architecture it is insufficient to increase the spatial resolution of the high-level features; the spatial resolution needs to be increased *at the right point* of the network—specifically, at a place that will improve the quality of the skip connections between the encoder and the decoder. This can be achieved by placing the ASPP module not at the bottom of the encoder, but at the top of the decoder, before the skip connection with the top block of the encoder. At this depth ($D = 0$), the features of the encoder have maximal spatial resolution and minimal semantic content, while the features of the decoder have minimal spatial resolution and maximal semantic content. Therefore, adding the ASPP module before the skip connection allows the top block of the decoder to increase its spatial resolution, making it easier for it to communicate with the top block of the encoder via skip connection.

From this idea, I designed the Spatial Resolution Enhancer (SRE) module. The SRE module consists of a variable number of parallel dilated convolutional layers, incapsulated in a 1 x 1 convolutional bottleneck (see Figure 5.4)[5] (see Figure 5.4). How many dilated convolutional layers the SRE should contain—and what their dilation rate should be—depends on the depth at which the skip connection takes place. If the skip connection takes place between the deepest layer of the encoder and that of the decoder (in the case of U-Net 1024, the deepest possible skip connection is performed at $D = 4$), the features of the encoder do not contain much more spatial information than those of the decoder. Therefore, only a little spatial resolution needs to be injected into the features of the decoder for the skip connection to be effective. Therefore, the SRE module at this depth does not need to be as large as the ASPP module of DeepLabv3+. Conversely, at $D = 2$ the features of the encoder have much higher spatial resolution than the features of the decoder, which means that the SRE module at $D = 2$ will need to embed more spatial resolution than the SRE module at $D = 4$. This leads to the formulation of a variable

[5]The main purpose of a 1 x 1 convolutional layer is to adjust the number of filters of a layer. Given that each layer in a CNN has dimensions n x m x num filters, the dimension of a layer (and therefore its computational expense) can be lowered by convolving it with a 1 x 1 convolutional layer that has a lower number of filters. For example, an input layer of shape n x m x 1024 can be convolved with a layer of shape 1 x 1 x 512 to halve its number of filters, making it less computationally expensive. After the 1 x 1 convolution, the heavy processing (i.e. the dilated convolutions) can be performed using fewer parameters. Finally, the dimension of the output is restored by applying a second 1 x 1 convolution. This type of structure, in which a a component of a network that is expensive is preceded and followed by 1 x 1 convolutional layers, is called a "bottleneck".

structure for the SRE module, summarised in Table 5.2.

Table 5.2: Dilation rates and number of layers of the SRE modules at each depth of the decoder

Depth	Number of Layers	Dilation Rates
4	1	4
3	2	4, 8
2	3	4, 8, 16
1	4	4, 8, 16, 32

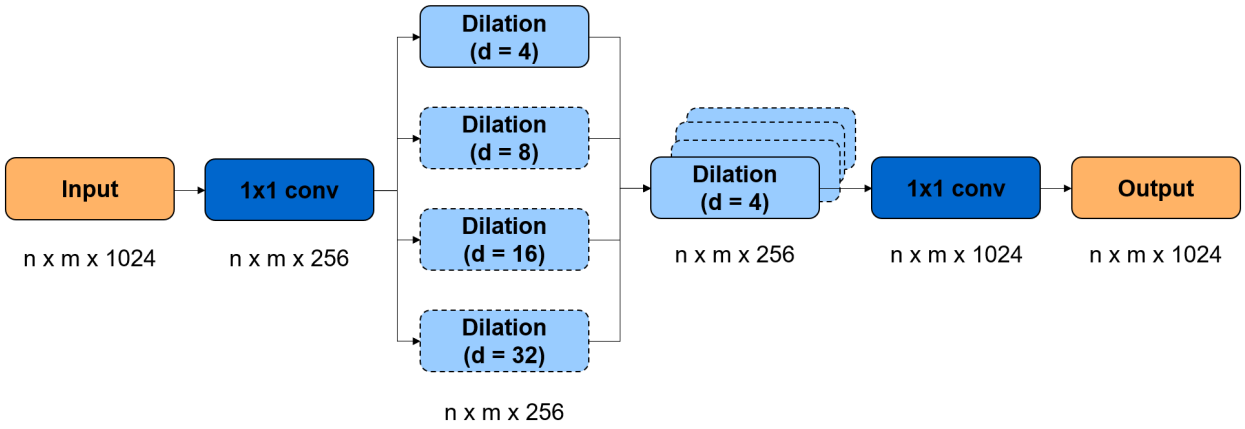


Figure 5.4: The SRE module takes as input a layer y_D of the decoder (after upsampling), reduces its number of filters using a 1 x 1 convolutional layer, and passes it through one to four dilated convolutions in parallel. The outputs of the dilated convolutions are concatenated and fed to a 1 x 1 convolutional layer, which restores the number of channels of the output. In this figure, the outline of most dilated convolutional layers is dashed because those layers will only be active at certain depths (see Table 5.2). (Image created by the author)

The variable complexity of the SRE module allows it to embed an amount of spatial information proportional to the expected spatial information of the features of the block of the encoder to which it will be connected. U-Net 1024 with the SRE module attached (but not ExFuse's SEB module) achieved a Dice score of 0.9972, which would place 17$^{\text{th}}$ on the Kaggle Carvana leaderboard.

It was at this stage of the development of the algorithm that the Scylla dataset became

available. Therefore, instead of trying to gain positions on the Kaggle Carvana leaderboard[6], I decided to continue developing the algorithm on Scylla.

5.2 Moving Development to the Scylla Dataset

To continue on Scylla the development of the algorithm, a baseline had to be re-established by training and testing U-Net 128, 256, 512, and 1024 on Scylla (see Table 5.3). I chose to keep the Dice score as the evaluation metric, to make it easier to compare results on Scylla with the previous results on the Kaggle Carvana dataset.

Table 5.3: Dice Score of Various U-Nets on the Scylla Dataset

Model	Dice score
U-Net 128	0.8727
U-Net 256	0.9374
U-Net 512	0.9540
U-Net 1024	0.9609

Two observations can be made regarding the results in Table 5.3. First, whereas on the Kaggle Carvana dataset the gap between U-Net 128 and U-Net 1024 was 0.65%, on Scylla this gap is 8.82%. This indicates that model depth is more correlated with high performance on Scylla than it is on Kaggle Carvana, which means that there is more complexity in the Scylla dataset than there is in the Kaggle Carvana dataset. The added complexity almost certainly comes from the fact that in the images in the Kaggle Carvana dataset the object of interest (the car) was located in the centre of the image. This simplifies the learning process, because even networks as small as U-Net 128 will quickly learn that the pixels at the centre of the image cannot belong to the background and that those towards the edges cannot belong to

[6]The team that placed first on the Kaggle Carvana leaderboard used over 15 GPUs to train 49 slightly different models, which they then averaged to obtain the final prediction. Furthermore, they heavily pre-processed the training images—for example, by completely re-labelling some of them. Both of these practices (pre-processing training data and model averaging) are discouraged in academic practices [187], but are allowed in public competitions. Indeed, all the teams who placed in the top 15 of the Kaggle Carvana leaderboard and who shared their solution used some kind of pre-processing and much more computational power than was available during this PhD. Therefore, continuing to fine-tune models on the Kaggle Carvana dataset was unlikely to improve my standing on the leaderboard unless more GPUs became available.

the foreground. Second, while on the Kaggle Carvana dataset U-Net 1024 already achieved a near-perfect Dice score, on Scylla there is more room for improvement, likely meaning that the capacity of the model needs to be increased.

To increase the capacity of U-Net 1024, VGG16 [116] was used as an encoder. The structure of VGG16 is almost identical to that of a basic U-Net's encoder, with the exception of added convolutional layers after the first two blocks. Using VGG16 as the encoder for U-Net 1024 has the additional benefit that it is possible to download a version of VGG16 that was pre-trained on ImageNet (a large object classification dataset). This means that the weights of the encoder would be already initialised to better-than-random values, and therefore it is likely that they would converge to better results. Indeed, this was the case: U-Net 1024 with a pre-trained VGG16 as encoder (a model that will be referred to as 'VGG16 U-Net 1024') achieved a Dice score of 0.9701 on Scylla. Encouraged by the apparent benefits of using pre-trained models, I decided to train VGG16 U-Net 1024 on the Kaggle Carvana dataset to initialise its weights for the task of semantic segmentation, and then re-train it on Scylla. This test led to a decrease in performance: VGG16 U-Net 1024 pre-trained on Kaggle Carvana achieved a Dice score of 0.9679 on Scylla. It is possible that the negative impact of the pre-training on the Kaggle Carvana dataset was due to an improperly calibrated transfer learning procedure (i.e. pre-training on one dataset and then re-training on a different one). When a model is trained on a dataset to initialise its weights and then it is re-trained (or 'fine-tuned') on a different dataset, during the fine-tuning stage the learning rate should be set to values much lower than those used during the pre-training. Otherwise, the large gradients that the network receives in the early stages of fine-tuning would wipe away the values to which the weights had been initialised. For the same reason, during the early stages of fine-tuning most of the weights of the network are frozen, and then they are progressively unfrozen the more the network is trained; this means that fine-tuning a network might take multiple iterations of the training process. However, there is no guideline that says to what value the learning rate should be set during fine-tuning, or how many layers should be frozen, or when they should be unfrozen. Therefore, it is possible that the way I implemented transfer learning was not optimal and had to be adjusted. It is also possible, however, that the weights learned on Kaggle were not beneficial

for the purposes of training VGG16 U-Net 1024 on Scylla. After all, the shapes of cars are different from those of humans, even though the task of identifying their contour is essentially the same. The only way to solve this ambiguity was to spend time fine-tuning VGG16 U-Net 1024 from Kaggle Carvana to Scylla. Since this might have led to this pre-training strategy being discarded anyway (since there were no guarantees that it would work), I decided not to pursue this strategy further, and instead concentrated on improving the performance of the algorithm by modifying its architecture.

The next set of experiments involved testing the SEB and SRE modules on Scylla, to confirm the results obtained on the Kaggle Carvana dataset. First, VGG16 U-Net 1024 was trained using just the SEB module, and it achieved a Dice score of 0.9709. Motivated by the results in Table 5.3, which showed that the baseline models were too shallow for the Scylla dataset, I decided to modify the SEB module. The function of the SEB module is to introduce more semantic meaning to the low-level features of the encoder before they are fused with the high-level features of the decoder. However, its original structure (see Figure 5.3) does not add much capacity to the model, given that it only introduces one non-linearity in the form of a 3 x 3 convolutional layer. Therefore, I modified the SEB module (see Figure 5.5) by adding a second 3 x 3 convolutional layer and encapsulating the two 3 x 3 convolutional layers inside a ResNet-like bottleneck (i.e. two 1 x 1 convolutional layers), the purpose of which is to reduce the number of filters before the expensive 3 x 3 convolutions and restore it after; this way, adding a second 3 x 3 convolutional layer does not cause the number of parameters to increase. VGG16 U-Net 1024 with the modified SEB module achieved a Dice score of 0.9711—a marginal improvement over the original SEB module.

Finally, I added the SRE module to VGG16 U-Net 1024, achieving a Dice score on Scylla of 0.9712. The reason for the marginal increase in performance attributable to the SRE module is unclear. In particular, it is unclear why it was more effective on the Kaggle Carvana dataset than on the Scylla dataset. Nevertheless, it did improve performance, and therefore was included into the final version of my algorithm, which I called 'FISHnet'. The following section will summarise the architecture of FISHnet, as well as give the implementation details required to reproduce the experiments.

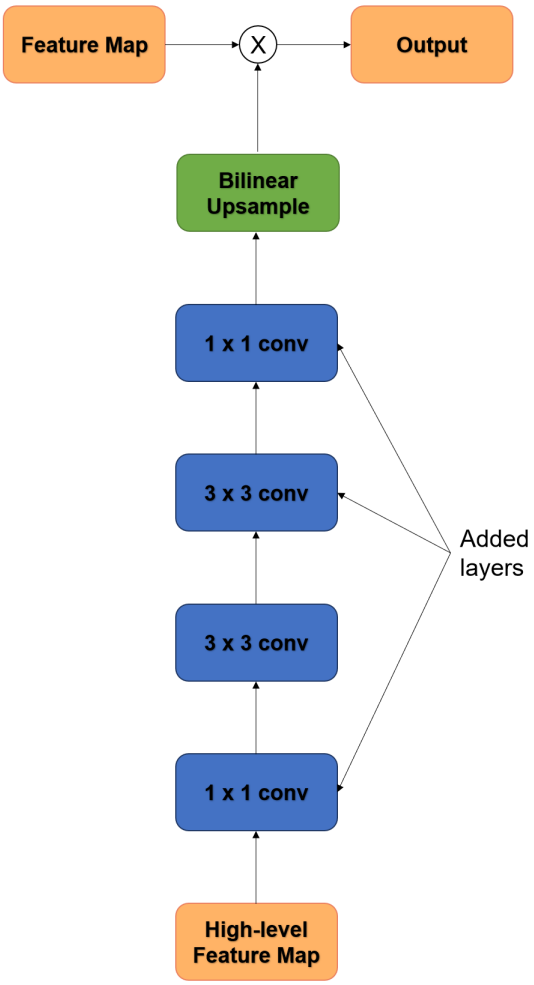


Figure 5.5: The modified SEB module; compared with the SEB module proposed by the authors of ExFuse [11], the modified SEB module adds one more 3 x 3 convolutional layers, and places the two 3 x 3 convolutional layers inside a bottleneck layer made of two 1 x 1 convolutional layers. (Image created by the author)

5.3 Final Architecture of FISHnet, and Implementation Details

FISHnet uses a U-Net-like architecture in which the encoder consists of VGG16 (pre-trained on ImageNet) and the decoder of a simple series of bilinear upsampling and convolutional layers. The main novelty of FISHnet's architecture lies in the structure of the skip connections that link the blocks of the encoder to the corresponding blocks of the decoder. Figure 5.6 shows the difference between the skip connections in a normal U-Net and those in FISHnet. In particular, the skip connections in FISHnet contain two modules: a modified version of ExFuse's Semantic

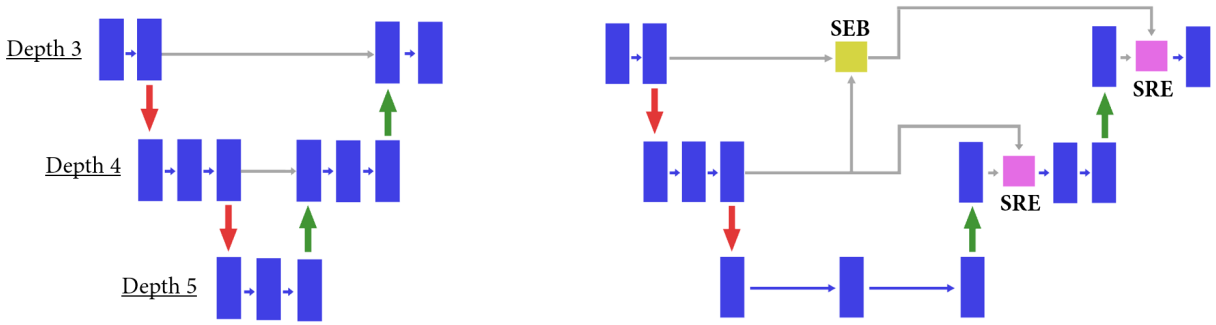


Figure 5.6: Left: skip connections in a normal U-Net; the output of the encoder at depth D is copied and concatenated with the first layer of the decoder at depth D. Right: skip connections in FISHnet; the yellow block represents the modified SEB module, while the pink blocks represent SRE modules. (Image created by the author)

Embedding Branch (SEB) module, and a re-purposed version of DeepLab's Atrous Spatial Pooling Pyramid (ASPP) module. The function of the SEB module is to encode more semantic information in the high-level features of the encoder and to increase the overall capacity of the model without increasing the number of its parameters (which I achieved by using ResNet-like bottleneck layers). The function of the SRE module is to encode more spatial resolution in the high-level (and therefore low-resolution) features of the decoder. The addition of these two modules allows blocks of the encoder and blocks of the decoder to communicate more easily, since they share similar semantic content at a similar resolution.

During training, early stopping was used, meaning training was forced to to stop once the validation loss had not decreased by at least 0.0001 for five consecutive epochs. Due to the limited memory available, during training all input images were resized to 1024 x 1024 (using padding when necessary) and the batch size was set to two. The following data augmentation techniques were used: horizontal flip, random re-scale, random rotation. The loss used was the Weighted Dice loss (see Equation 5.1). The optimiser used was RMSprop with an initial learning rate of 0.001. FISHnet was implemented using Keras with TensorFlow backend. All training and testing was done on an Nvidia Titan X GPU.

Table 5.4: Results on the Scylla Dataset

Network	Dice score
U-Net 128	0.8727
U-Net 256	0.9374
U-Net 512	0.9399
U-Net 1024	0.9609
DeepLabv3+	0.9510
FISHnet	**0.9712**

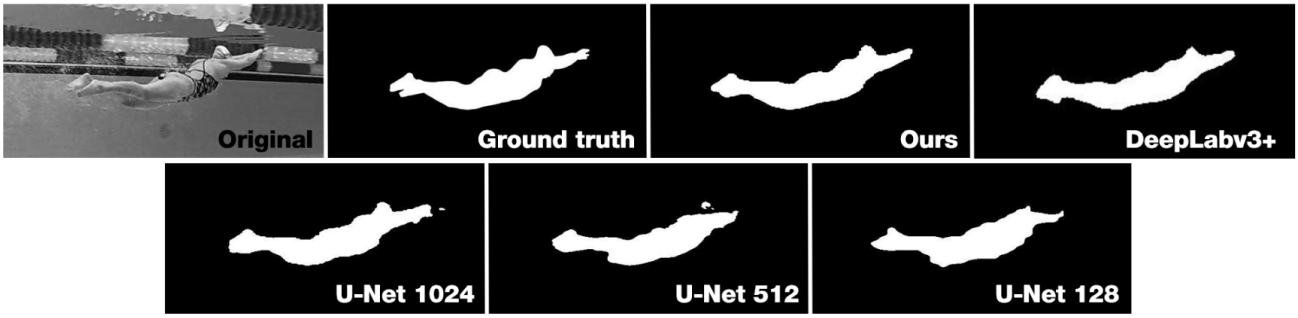


Figure 5.7: Comparison of the algorithms considered in this paper on a sample image from the Scylla dataset. All images were cropped around the figure of the swimmer, to make the fine details of each image more visible.

5.4 Evaluation: FISHnet and State-of-the-Art Algorithms on Scylla

To evaluate FISHnet, DeepLabv3+ was also trained on the Scylla dataset, using DeepLab's official source code[7]; as there is no source code available for ExFuse, it could not be implemented. Table 5.4 reports the test Dice score for DeepLabv3+, FISHnet, and the U-Net architectures that were tested. Figure 5.7 also reports the output of the algorithms reported in Table 5.4 when tested on a sample image from Scylla's test data, so that the differences between the algorithms' performance can be visualised.

The fact that DeepLabv3+ achieved a lower Dice score than both FISHnet and U-Net 1024 goes against my intuition that the reason U-Net 1024 performed more poorly on Scylla than on the Kaggle Carvana dataset was that Scylla requires models with greater capacity. At least, it indicates that additional capacity alone is insufficient to increase performance on Scylla, since

[7]https://github.com/tensorflow/models/tree/master/research/deeplab

DeepLabv3+ is a deeper, more complex model than both U-Net 1024 and FISHnet. This result led me to a different interpretation of why larger U-Nets performed better than smaller ones. Namely, I believe that the larger the input (and therefore output) resolution of a network, the more accurate its predictions will be. Intuitively, this seems like a reasonable assumption: if a network outputs a silhouette with a resolution lower than 2048 x 1088 (which is the resolution of the images in Scylla), this output needs to be upsampled to 2048 x 1088 before it can be tested against the ground truth present in the Scylla dataset. If the output resolution is below 2048 x 1088, details along the contour of the silhouette will be lost during the upsampling process, resulting in lower-resolution silhouettes. Figure 5.7 visually confirms this intuition: the silhouette of U-Net 1024 has more detail than the silhouette of U-Net 128, and a smoother contour. As the output resolution moves closer to 2048 x 1088, more details of the silhouette will be preserved. Since DeepLabv3+ outputs images at a resolution that is 8 to 16 times (depending on the implementation) smaller than the input image, its accuracy is hampered when the test images are as large as those of Scylla. It is unsurprising, then, that DeepLabv3+, which is a deeper and more complex model than FISHnet and U-Net 1024, does not outperform either despite having a more complex and deep architecture.

However, even if increasing capacity is not the most optimal way to increase performance on Scylla, it is still surprising to see that the modified SEB module and the SRE module had only a marginal impact on performance. One interpretation of this result is that the SRE and the modified SEB modules do not actually improve the effectiveness of the skip connections as intended. However, the original SEB module on FISHnet also has a marginal contribution to overall performance (0.08%), whereas in ExFuse it had a contribution almost ten times as large (0.7%) [11]. Therefore, there must be another explanation for why all additional modules that are attached to FISHnet only have marginal effects. I believe the explanation could be found in the amount of training data available. The optimal capacity of a neural network depends on the complexity of the task at hand and on the amount of training data available [187]—which, in the case of the Scylla dataset, is only 2,635 images. This could mean that FISHnet reaches optimal capacity roughly at baseline, and that adding capacity to the model (by introducing the SRE and modified SEB modules) past that point results in overfitting. Since overfitting is mitigated

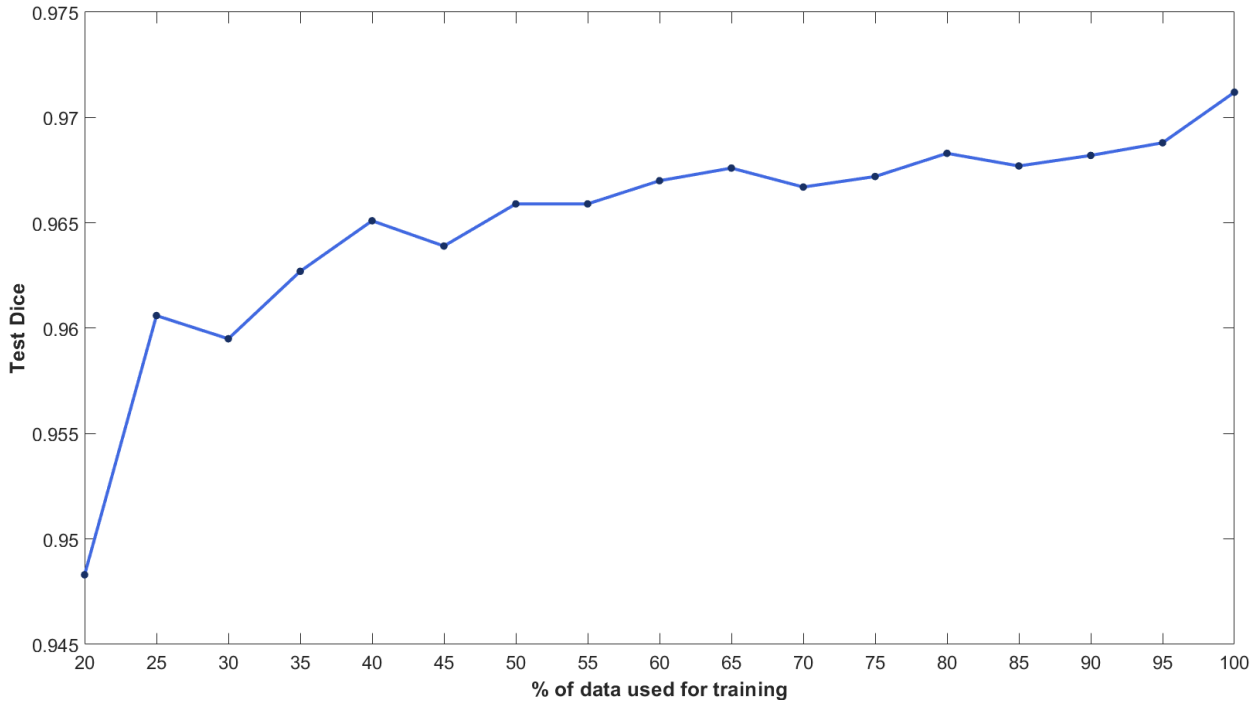


Figure 5.8: Results of the Inverse Power Analysis (IPA) performed on FISHnet, using from 20% to 100% of the training data available in Scylla.

by using early stopping (i.e. stopping training when the validation loss stops improving), all variants of FISHnet converge to similar values because the added capacity of the more complex models is not allowed to be trained. To test this theory, FISHnet was re-trained without early stopping, and observed that convergence took 59 epochs (compared to 26 with early stopping) and reached a train dice of 0.9740 (compared to 0.9715 with early stopping). Furthermore, the test dice without early stopping was 0.9687 (compared to 0.9712 with early stopping). This finding suggests that FISHnet has enough capacity to achieve higher dice scores, but is gated by the limited data available to it for training. In other words, if it expresses its full capacity it suffers from overfitting.

To further assess whether FISHnet incurs into overfitting because Scylla's training set is too small, an Inverse Power Analysis (IPA) was performed by training FISHnet on progressively larger portions of Scylla's training set and recording the test Dice score after each training cycle. What an IPA graph can be used for is estimating if the training data that were used were sufficient, or if they were so few as to cause the algorithm to overfit. The results of this analysis are shown in Figure 5.8. Though the curve does seem to plateau between 65

and 85% training data, it also shows an unexpected upward trend starting from 85% training data, indicating that the flatness of the plateau might have been exaggerated by the sparsity of the data points. However, the same reasoning can be applied also to the points responsible for the late upward trend. Therefore, Figure 5.8 is inconclusive with regards to the amount of overfitting into which FISHnet incurs. To provide a clearer answer, a second IPA was performed, in which on the vertical axis is reported not the test Dice, but the difference between the train Dice and the test Dice, which is the definition of overfitting [187]. Figure 5.9 shows a trend

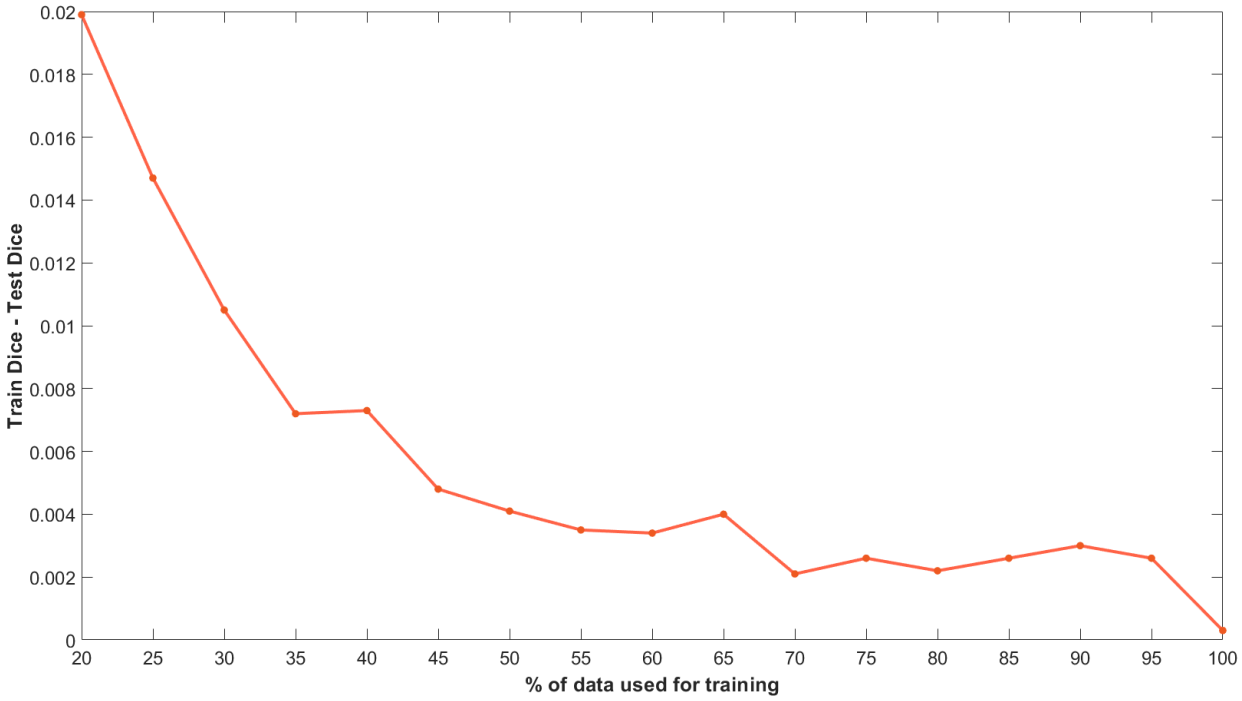


Figure 5.9: Results of the modified IPA performed on FISHnet.

almost identical to the one in Figure 5.8, in terms of the amount of overfitting present: for low amounts of training data FISHnet overfits badly; as more training data are used, the amount of overfitting decreases and eventually plateaus; but then, the downward trend resumes quite sharply. This confirms that Scylla could benefit from having more training images.

Two more observations, related to each other, can be made from the graphs in Figures 5.8 and 5.9. First, while optimal performance requires more training data than were available, even a small fraction of Scylla's training set is enough to train models to almost 95% Dice score. A possible explanation for this is that there is insufficient variability in the data; therefore, a small number of training examples are enough to cover most of the variability present in

the dataset and achieve high Dice scores. However, this explanation is incompatible with the upward trend observed in Figure 5.8. Second, though the graph in Figure 5.9 shows that using all the training data causes ten times less overfitting than using 20%, the scale of the values on the vertical axis is small. A possible explanation for this is that the test set is too similar to the training set. This would explain the small scale of the values on the vertical axis in Figure 5.9, but not that the test Dice of FISHnet trained with only 20% data is already high. Therefore, it is unclear whether the images in Scylla have enough variability. However, it is likely that Scylla would benefit from having more test data, as well as more varied data.

Finally, Figure 5.10 shows three examples of images on which FISHnet performed particularly poorly. The failure on image A of Figure 5.10 can be attributed to the presence of a

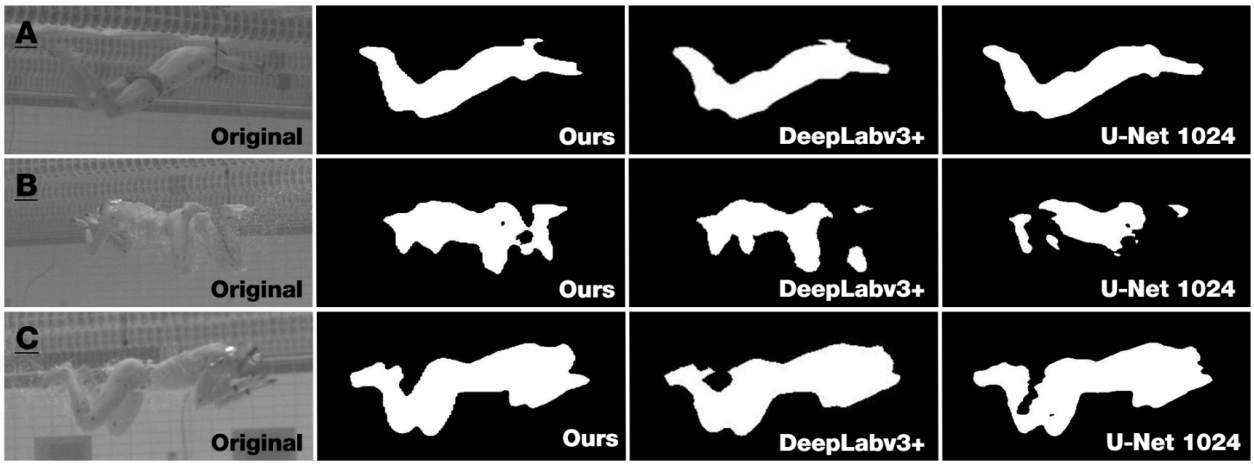


Figure 5.10: Three examples of FISHnet network making noticeable segmentation errors. The outputs of DeepLabv3 and U-Net 1024 are also reported here to understand if the images on which FISHnet failed were inherently difficult.

background object (a floating pole) that, because it was in front of the head of the swimmer, was segmented as belonging to it. The mistake is repeated by DeepLabv3, but, unexplainably, not by U-Net 1024. The pole is also visible in image B of Figure 5.10, but in this case it was not segmented as part of the swimmer's silhouette, likely because in that image the pole colour is inconsistent with any part of the swimmer, whereas in image A the pole was of the same colour as the swimmer's cap, with which it was juxtaposed.

The reason for FISHnet's failure on image B of Figure 5.10 is not as easily diagnosed, though it is clear that this image must have some feature that makes it difficult, as all algo-

rithms performed poorly on it. In this image, the colour camouflaging between the swimmer and the background is pronounced. Indeed, even a human might struggle to detect the outline of the legs. However, many of the images in the Scylla dataset have a similar degree of colour camouflaging—for example, though to a lesser degree, image A in Figure 5.10. Another possible explanation for why this image was challenging is the presence of bubbles around the legs (the area where the grossest segmentation mistakes were made): the algorithms might have learned from the training data to ignore bubbles, and therefore in this image ignore much of the legs. The negative effect of bubbles is somewhat confirmed by image C of Figure 5.10, in which all algorithms fail to correctly segment the left shank and right foot of the swimmer—the areas most enveloped by bubbles. However, the effect of bubbles on this image was less drastic than in image B. It could be concluded, then, that the reason all algorithms performed poorly on image B was the combination of bubbles and colour camouflaging. Since these two conditions overlap rarely in the images of the Scylla dataset, FISHnet commits few gross mistakes. The main source of error for the silhouette extraction of swimmers, then, is not to be found in easily observable mistakes as the ones shown in Figure 5.10, but in small-scale errors that occur along the contour of all silhouettes. Because FISHnet and U-Net 1024 should reconstruct the most accurate outlines (since their output resolution is the closest to the resolution of the images in the Scylla dataset), and since it was just established that accuracy along the contour is likely to be indicative of overall performance on the test set of the Scylla dataset, one would expect in theory that FISHnet and U-Net 1024 should outperform the other algorithms; this result is confirmed empirically by the results reported in Table 5.4. In other words, I attribute the margin of FISHnet over DeepLabv3+, U-Net 512, and U-Net 128 to it outputting smoother silhouettes, which in turn is a consequence of the higher output resolution on which FISHnet and U-Net 1024 operate. What accounts for FISHnet's advantage over U-Net 1024 is the presence of the SEB and SRE modules (to a minor extent) and the fact that FISHnet uses a pre-trained VGG16 as an encoder (to a greater extent).

5.5 Conclusions

The results reported in this chapter indicate that the Scylla dataset would likely benefit from having more (and more varied) training and test data. In particular, the size of Scylla did not allow the unique modules of FISHnet (the SRE and modified SEB modules) to express their full potential without making the network overfit. Nevertheless, these additions to U-Net 1024 did improve performance over the baseline network. However, a much bigger impact on performance is attributable to the choice of using a pre-trained encoder, VGG16. This conclusion agrees with the current theoretical understanding of neural network training [187].

The main advantage of FISHnet over DeepLabv3+ seems to be that FISHnet outputs prediction at higher resolution than DeepLabv3+. This allows FISHnet to predict silhouettes with smoother, more detailed contours, at the cost of not having as much capacity as DeepLabv3+—a cost which, given that FISHnet already overfits on Scylla, is negligible. Therefore, whereas much of the incorrectly classified pixels of DeepLabv3+ and small U-Net models (like U-net 128 and 256) are due to low-resolution contours, the main sources of error for FISHnet are images in which the silhouette is particularly difficult to separate from nearby bubbles or objects, or images in which colour camouflaging is particularly pronounced.

Having developed an algorithm for silhouette extraction that outperforms the state-of-the-art on the Scylla dataset, the next step towards the realisation of the aim of this PhD was to build a dataset (described in Chapter 6) on which to train a 2D pose detection algorithm (the development of which is described in Chapter 7).

Chapter 6

Construction of Charybdis, a Dataset of Images of Swimmers for 2D Pose Detection

6.1 Preliminary Considerations

From the development and evaluation of FISHnet it emerged that the Scylla dataset probably had too few images, especially in the test set. As discussed in Section 4.1, there are no precise heuristics for determining a priori how many training and test examples a dataset will need. Therefore, Scylla being too small does not necessarily mean that Charybdis should have more images than Scylla, because the two datasets model two different data-generating processes, one of which may be more complex (and therefore require more images) than the other. Nevertheless, it is likely that the task of 2D pose detection requires more training images than that of silhouette extraction, as one image labelled for 2D pose detection provides less supervision than one labelled for silhouette extraction[1]. This intuition appears to be confirmed by the fact that

[1]The more pixels of an image are labelled, the more a neural network can learn from that image. In an image labelled for 2D pose detection, usually 14-15 pixels are labelled (though this number can be increased by using heatmaps, as discussed in Section 2.5.4); in an image labelled for silhouette extraction, (potentially hundreds of) thousands of pixels are labelled.

MPII, the most popular dataset for 2D pose detection, has 25,000 images, making it 2.5 times larger than PASCAL VOC, the most popular dataset for silhouette extraction. However, it is unlikely that Charybdis should contain as many images as MPII, since there is more variability in MPII than there needs to be in Charybdis (see Section 2.5.2). This is the same reasoning that was used to justify why Scylla should have fewer images than PASCAL VOC. Given that it was ultimately found that Scylla did not contain enough images, it is possible that this reasoning is not valid, and therefore that the Charybdis dataset should contain about 25,000 images. This conclusion seems unlikely: MPII undoubtedly contains more variability than is needed in Charybdis, and therefore Charybdis certainly needs fewer images than 25,000—but how many fewer? Given that the second most common dataset for 2D pose detection, the Leeds Sports Pose dataset, contains only 2,000 images, and taking into consideration the time available during the PhD to gather and label data, I estimated that about 8,000 images would be sufficient for the Charybdis dataset—provided that they be more variable than the images in Scylla, and that enough of them be allocated to the test set. To increase the variability of the images of The Collection, new videos of underwater swimmers were recorded, at the Manchester Aquatics Centre. This data acquisition session is described in the following section.

6.2 How the Data were Gathered

Seven participants were recruited for the data acquisition session at the Manchester Aquatics centre. Of these, four were highly trained swimmers (5+ years of practice). All swimmers wore tight-fitting swimwear. Unfortunately, all swimmers were males, potentially introducing a source of dataset bias, which will need to be corrected (by recording videos of female swimmers) in future work.

Each swimmer performed two trials of underwater breaststroke; the four experienced swimmers also performed two trials of underwater butterfly kick. The swimmers were recorded with an eight-camera system, set up fully underwater (see Figure 6.1). The cameras used were POI (GigE) Mako G-223B cameras (Allied Vision Technologies GmbH, Stemmer Imaging, Surrey,

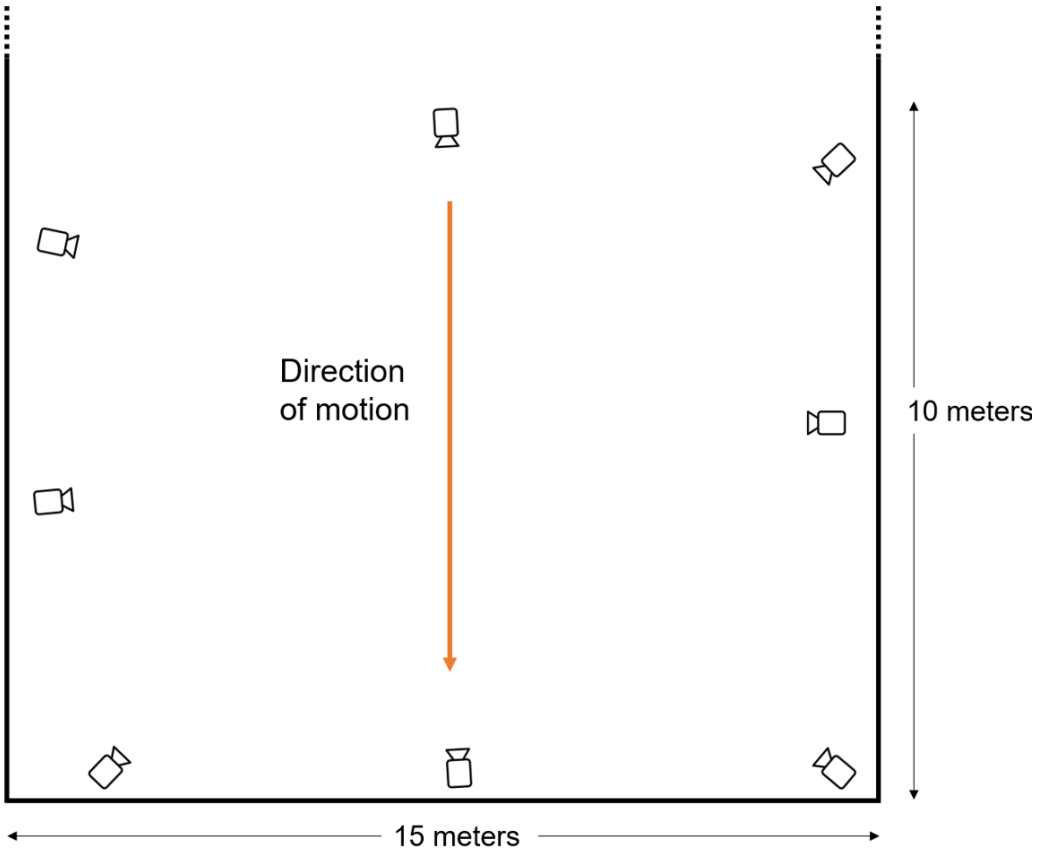


Figure 6.1: Camera setup used to record videos at the Manchester Aquatics Centre. (Image created by the author)

UK), and they were placed in Nautilus IP68-rated waterproof housings (Autovimation GmbH, Baden Württemberg, Germany). The recording software used was Gecko GigE Video Recorder v1.9.4, (Vision Experts Ltd, Surrey, England). The cameras were placed asymmetrically to avoid cameras on opposite sides of the pool recording specular images, which would reduce the benefit of using multiple cameras. For the same purpose, cameras were placed at various depths below the water surface, ranging from 0.5 to 1.5 m. The cameras were calibrated using the custom-made 3D frame shown in Figure 6.2, which has 42 control points. Though for the purposes of building the Charybdis dataset it was unnecessary to calibrate the cameras, these videos might be used in future research to perform 2D-to-3D inference, for which having calibrated cameras is required. For the same reason, the cameras were synchronised by flashing an LED light (visible by all cameras) moments after the cameras started recording. In total, setting up the equipment and calibrating the cameras took about five hours.

From the videos recorded, 2,113 images were extracted, using the methodology described

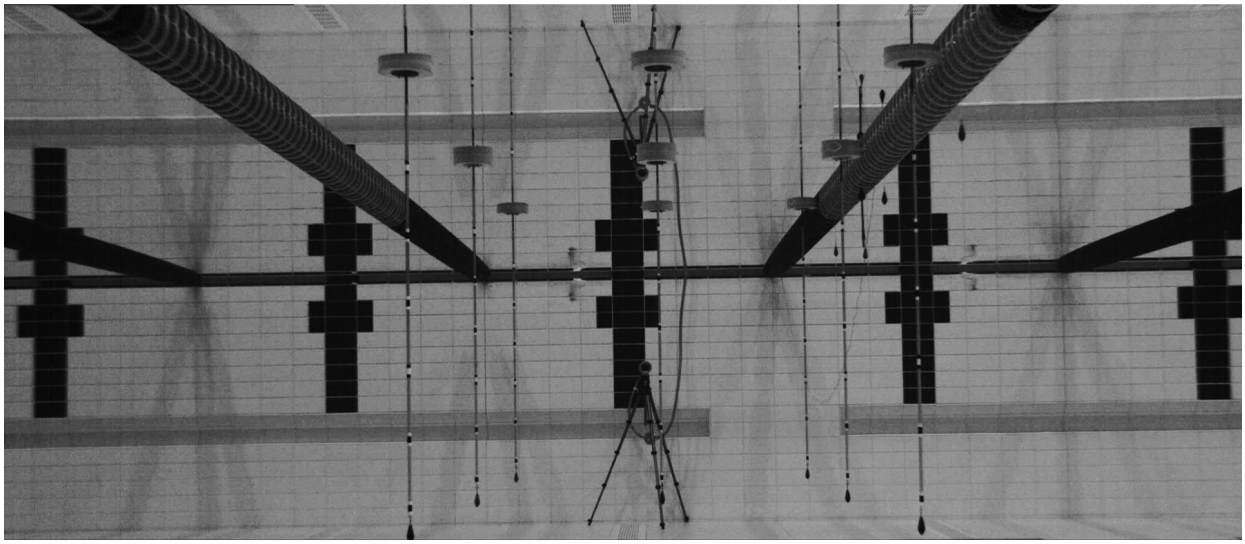


Figure 6.2: Custom-made 3D frame used to calibrate the cameras.

in Section 4.2. To these 2,113 images were added 6,000 images extracted from the videos in The Collection, giving a total of 8,113 images. The images were then split into a training set of 6,485 images (of which 20% came from the Manchester recording session) and a test set of 1,628 images (of which 50% came from the Manchester recording session), for an 80-20 train-test split. Compared to Scylla, therefore, Charybdis has images from one more swimming pool (two instead of one), it has a higher percentage of test data (20% instead of 15%), and it features more swimmers (21 instead of 14).

6.3 How the Data were Labelled

The first thing to decide when labelling images for a 2D pose detection dataset is what joints to label. Within the scope of this PhD[2], the purpose of Charybdis is to train an algorithm to automatically detect a certain number of joints of swimmers, joints that would then be used as constraints on which to fit a parametric model—most likely SMPL, the most popular parametric model available. Therefore, it is reasonable to label in the images of Charybdis the same joints that a SMPL model expects (see Figure 6.3). However, the 'joints' of a SMPL model do not necessarily correspond to anatomical joints. Rather, they are keypoints onto which the

[2]Outside of the scope of this PhD, the Charybdis dataset could be used to train algorithms for 2D pose detection disentangled from 2D-to-3D inference. In this scenario, the joints to be labelled might be different.

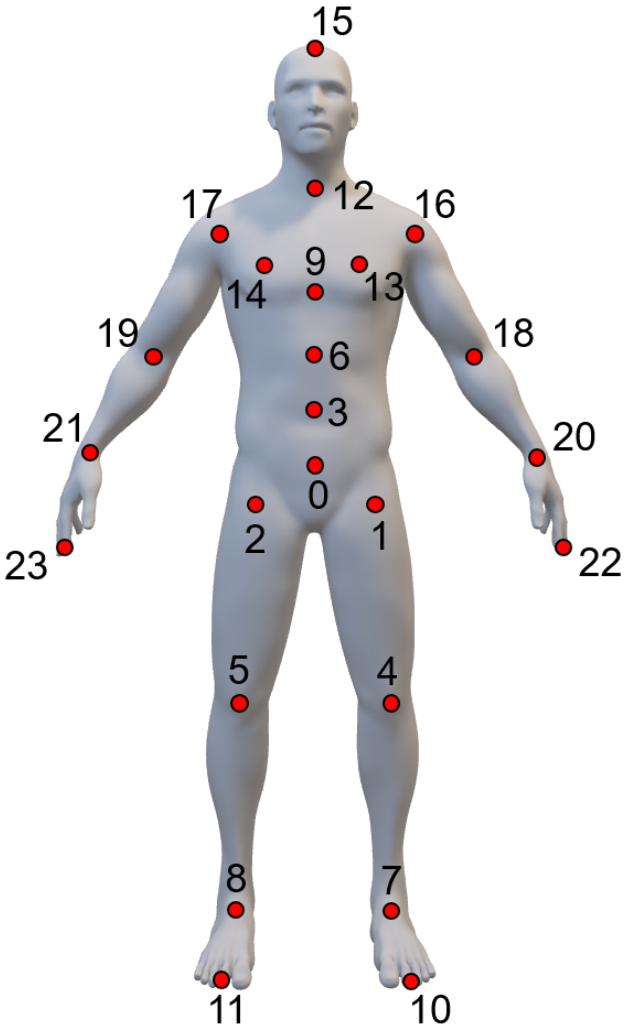


Figure 6.3: Joints of the SMPL parametric model. (Image created by the author)

model—which is a 3D mesh—is anchored and around which the parts of the model are rotated and translated. These 'joints' are derived statistically as the points that produce the most realistic (but not necessarily anatomically correct) model poses. Therefore, 'joints' like number 13 or 14 in Figure 6.3 do not correspond to any particular joint (though they probably are related to the scapulae). Therefore, they cannot be identified by any bony landmarks, making it hard to label them reliably. Similarly, 'joints' 22 and 23 lack a clear definition: do they refer to the tip of a finger? If so, which one? Or do they refer to the outer-most point of the hands, meaning if the hands were curled into fists, they would be labelled on the outer-most knuckles? The same questions apply to 'joints' 10 and 11. However, SMPL does not require all joints to be labelled (though the optimisation algorithm will converge faster if more joints are labelled). For this reason (and to reduce the time spent labelling), I decided to label only

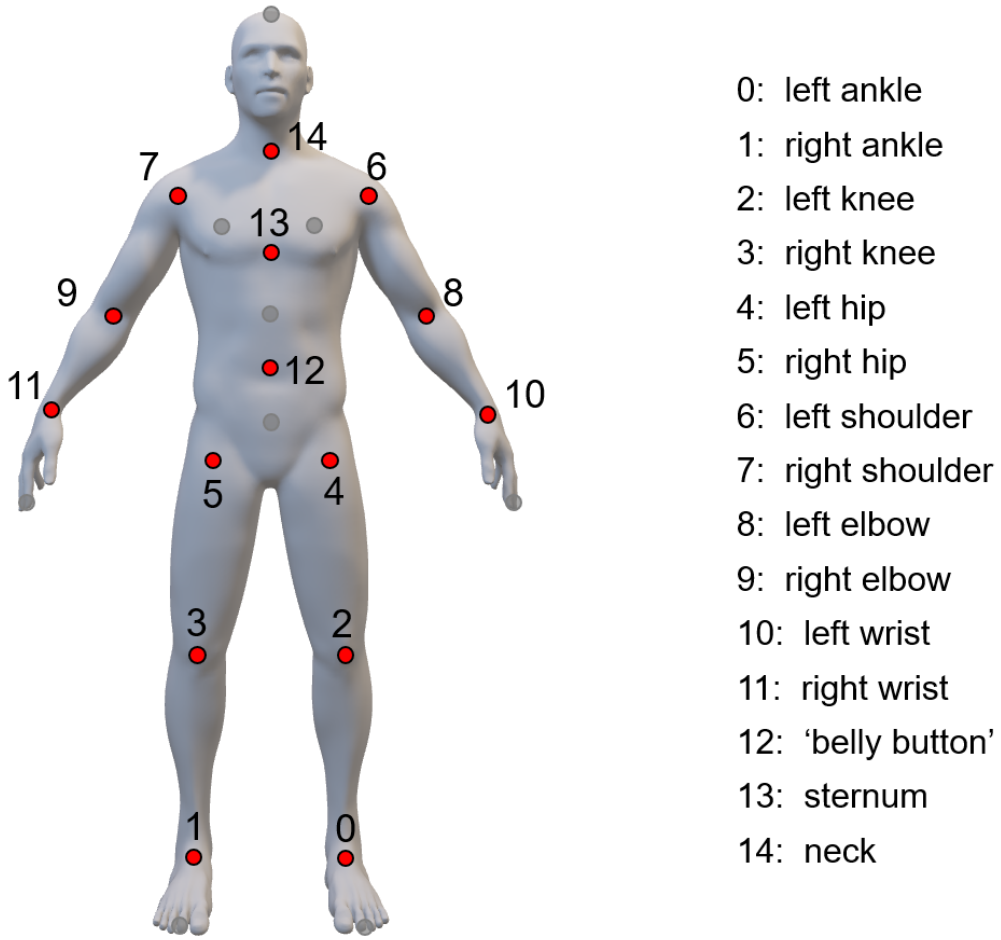


Figure 6.4: The joints labelled in the Charybdis dataset are a subset of the joints of a SMPL model. In this image, the joints of SMPL that are not labelled in Charybdis are greyed out. (Image created by the author)

the joints of SMPL that I consider essential (see Figure 6.4), in terms of the strength of the constraint they would impose on the optimisation of a SMPL model. These joints are the left and right ankles, knees, hips, shoulders, elbows, and wrists; the 'belly button' (defined in the next paragraph and visible in Figure 6.4); the sternum; and the neck, for a total of 15 joints[3]. This list of joints is similar to that of the joints labelled in the MPII and LSP datasets, with the difference that in Charybdis the top of the head is not labelled but the 'belly button' is. This decision was motivated by the fact that having an additional constraint (i.e. labelled joint) along the spine benefits the early stages of the optimisation algorithm that fits a SMPL model to the constraints. Since SMPL models are not constrained to anatomical poses, only

[3]In SMPL, the joints listed have such a strong relationship to anatomical joints as to make unimportant the distinction between anatomical joints and model joints. For example, joint 4 of a SMPL model (see Figure 6.3) can be interpreted as being the actual knee of the model, placed in its anatomical position.

labelling two joints along the spine (neck and sternum) might cause the model to 'fold onto itself'[4]. Therefore, adding more constraints along the spine allows the optimisation algorithm to quickly converge towards configurations of the 3D model that are anatomically sound.

As discussed in Section 2.5.1, the points to be labelled for 2D pose detection are not the superficial bony landmarks that would be labelled for manual digitisation systems. Rather, they are the true centres of rotation of the joints. Through consultation with a physician, the following guidelines were established to locate in an image the joints shown in Figure 6.4:

- Ankles: midpoint between the two malleoli;
- Knees: midpoint between the two epicondyles;
- Hips: if in flexion/extension, at the intersection between the line that vertically bisects the femur and the line that passes through the pelvis and about which the flexion/extension is observed; if neutral, 2 cm proximal to the greater trochanter;
- Shoulders (glenohumeral joints): if the acromion is visible, 2 cm below the acromion; if the acromion is hidden, in the centre of the deltoid muscle;
- Elbows: midpoint between the two epicondyles;
- Wrists: midpoint between the two styloid processes;
- 'Belly button': on the line that vertically bisects the torso, just below the belly button;
- Sternum: on the line that vertically bisects the torso, at the height of the sternum;
- Neck: on the line that vertically bisects the torso, at the base of the neck.

Therefore, while the location of the joints does not coincide with the location of the superficial bony landmarks, it is inferred from their location. In the videos of The Collection, whose original purpose was to be used for manual digitisation, tape markers were applied to most of the bony

[4]This phenomenon can be seen in commercially available marklerless motion capture systems that fit 3D models to 2D joints, such as SIMI Shape: from my experience with the software, I have noticed that if not enough joints are labelled along the spine, the 3D model will often fold onto itself, making further optimisation slow and potentially divergent.

landmarks listed above. The presence of these markers makes it easier for human labellers to estimate the location of 3D joint centres, but learning algorithms could also leverage this 'assistance' and fail on images of swimmers without tape markers on their skin. In the videos recorded in Manchester, no tape markers were used, making the images extracted from those videos harder for neural networks to learn from. Therefore, it is essential that the test set of Charybdis be split into 50% images from The Collection and 50% from the Manchester recording session, so that learning algorithms would be penalised for over-reliance on the presence of tape markers, and instead learn to rely on the actual bony landmarks (on which there may be a tape marker).

To ensure that the joints were labelled in a format that would be convenient as input for a neural network, I developed a labelling tool using PyQt5, Python's binding of the Qt toolkit. The labelling tool consists of a Graphic User Interface (GUI) (shown in Figure 6.5) from which images can be loaded and saved. The tool only functions in full-screen resolution, and requires a monitor with a resolution of at least 1920 x 1080. To select a joint to label, users can either

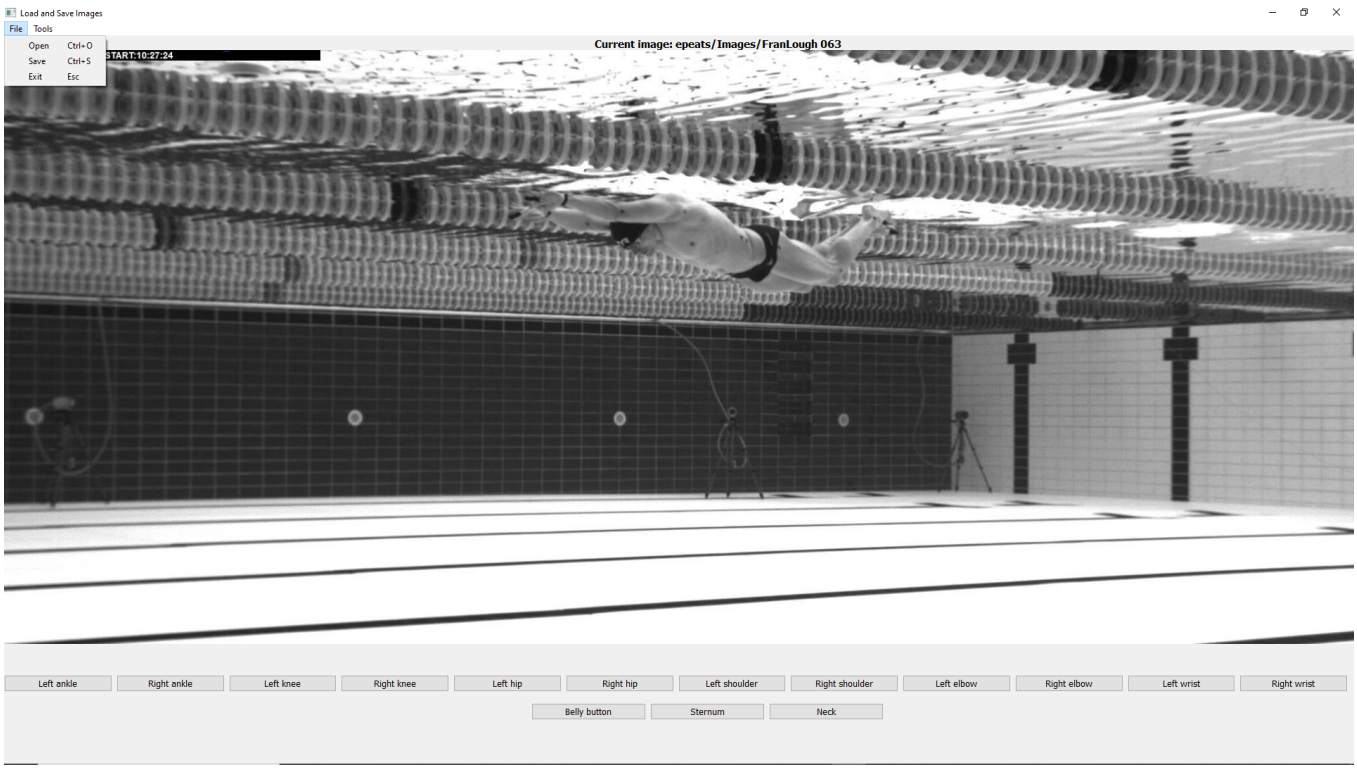


Figure 6.5: The labelling tool consists of a simple GUI shown here and of a list of buttons that users can press to select which joint to label. (Image created by the author)

click on the corresponding tab at the bottom of the GUI, or they can cycle through the list of

joints by pressing the W and S keys on their keyboard. Once a joint is selected, users can click on a point in the image, and the coordinates of that point are saved under the name of the joint selected. The coordinates saved are in the reference system of the image, which most Python packages (including PyQt5) define as having origin in the top-left corner of the image, with the x axis going from left to right and the y axis going from top to bottom. Joints can be labelled in two ways: with a left mouse click for joints that are visible, and with a right click for occluded joints. After a joint has been labelled, a green (for visible joints) or yellow (for occluded joints) dot appears around the pixel that was clicked on, to allow users to judge whether the joint was labelled in the correct position. If it was not, the joint can be re-labelled an indefinite number of times, without needing to re-start the entire process. Furthermore, users have the option to visualise on screen all the joints that have been labelled for the current image, with lines connecting related joints (see Figure 6.6). This functionality is beneficial during the labelling

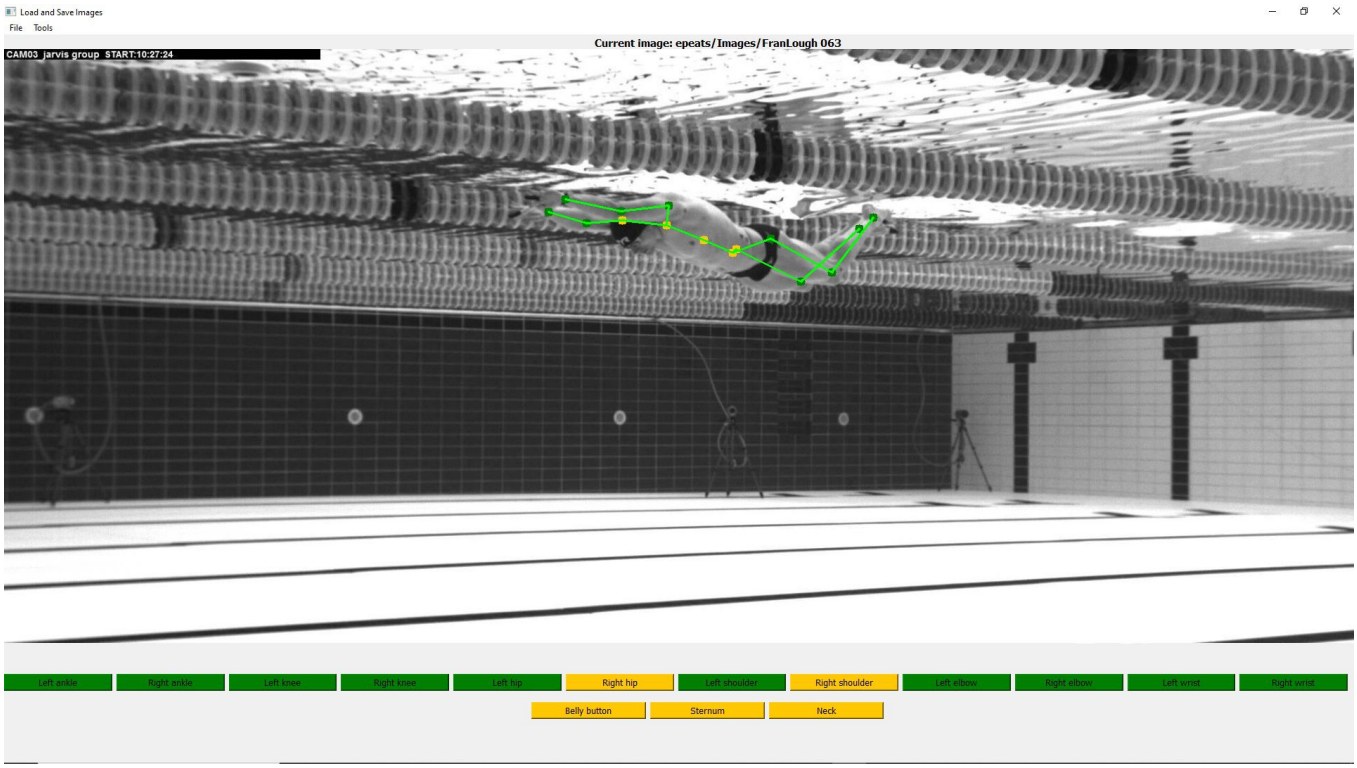


Figure 6.6: Users can press CTRL + H to show/hide all the joints that have been labelled for the current image. The links that appear between joints can be useful to more accurately estimate the position of occluded joints. (Image created by the author)

process, as it can guide users to better estimate the location of occluded joints. For example, in Figure 6.6 the right shoulder is occluded, but its position can be estimated fairly accurately by

visualising the links between the other joints: Since the arms of the swimmer in this image are parallel, one would expect the elbow-shoulder segment of the right arm to be parallel to—and of proportional length, depending on the perspective—the elbow-shoulder segment of the left arm. Therefore, even though the right shoulder is occluded, it can be labelled at the point that satisfies these assumptions. After all joints have been labelled, the results are saved in a .csv file (one per image) with 15 rows (one per joint) and four columns: one for the name of the joint; two for the x and y coordinates of the joint; and one with a binary value indicating whether the joint was visible ('visible' = 1) or occluded ('visible' = 0).

To assist in labelling the 8,000 images of Charybdis, a biomechanics MSc student was recruited. This second labeller was given an overview of the task of 2D pose detection, as well as an explanation of how building the Charybdis dataset fit into the grander scheme of the PhD. She was then given the labelling guidelines listed above, as well as a guided tutorial on how to use the labelling tool. A further guideline was established regarding how to label occluded joints. Namely, only joints whose position cannot be estimated reasonably well should be labelled as occluded ('visibility' = 0, i.e. right mouse click); joints that are not fully visible but whose position is obvious should be labelled as visible. For example, in Figure 6.5 the right ankle is not fully visible, as the two landmarks used to identify its location (the two malleoli) are occluded. However, its location can still be estimated reliably, and therefore this joint should be labelled as visible: if humans can locate it reliably from context, we would like algorithms to learn to do the same.

After all 8,000 images had been labelled (which took about 3 months), both operators labelled the same set of 100 images five times, to estimate human reliability on 2D pose detection. To calculate this estimate, it would be appropriate to use a metric that is familiar to researchers in 2D pose detection, like the PCK@0.1 metric—or, ideally, the PCKh@0.5 metric described in Section 2.5.2, which is the metric used to evaluate models on the MPII dataset[5]. However, computing the PCK metric requires that the bounding box that encloses the full person be

[5]As a reminder, the Percentage of Correct Keypoints (PCK) metric measures the percentage of predicted keypoints (i.e. joints) whose distance from their respective ground truth joints is smaller than a certain threshold. For PCK@0.1, this threshold is calculated as 0.1 times the largest dimension (in pixels) of the bounding box that tightly encloses *the full body* of the person. For PCKh@0.5, the threshold is calculated as 0.5 times the largest dimension (in pixels) of the bounding box that tightly encloses *the head* of the person.

labelled; for the PCKh metric, the bounding box that encloses the head of the person needs to be labelled instead. Since no bounding boxes were labelled for the images of Charybdis (as this task would have almost doubled the time required to label the Charybdis dataset), I devised the following strategy to derive a metric comparable to PCKh@0.5.

For a given image, and given the x and y coordinates of all labelled joints, the tightest bounding box possible that encloses all the labelled joints is the box that horizontally goes from x_{min} to x_{max} and vertically goes from y_{min} to y_{max} (see Figure 6.7). However, this bounding

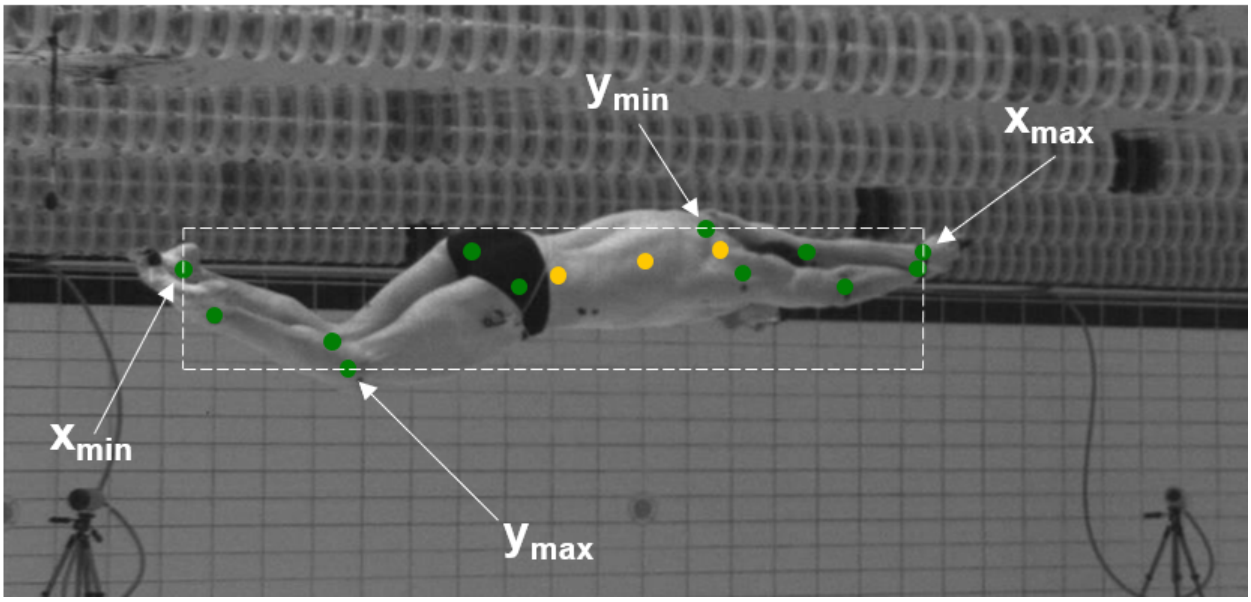


Figure 6.7: Example of how to obtain the tightest bounding box possible. The reference frame considered here is the one that most Python packages use: the origin of the system is in the top-left corner of the image, the x axis points to the right, and the y axis points downward. (Image created by the author)

box is tighter (by an amount that cannot be calculated[6]) than the bounding boxes used to calculate PCK or PCKh, since those boxes enclose not just the joints, but also the full body of the person (in the case of PCK). I found empirically that if the tightest bounding box possible is extended by 50 pixels in all four directions, the resulting box, on average, encloses the entire swimmer (see Figure 6.8). The advantage of such a bounding box is that it does not require any explicit labelling, as it is derived trivially from the joint labels[7]. Given such a bounding

[6]This amount cannot be calculated because it is not clear how the bounding boxes used in MPII were obtained.

[7]Though convenient, this strategy to acquire bounding boxes does require that the joints be labelled, and therefore could not be used on new, unlabelled data. Future research should seek to develop an algorithm to automatically detect the bounding boxes of swimmers in new, unlabelled images.

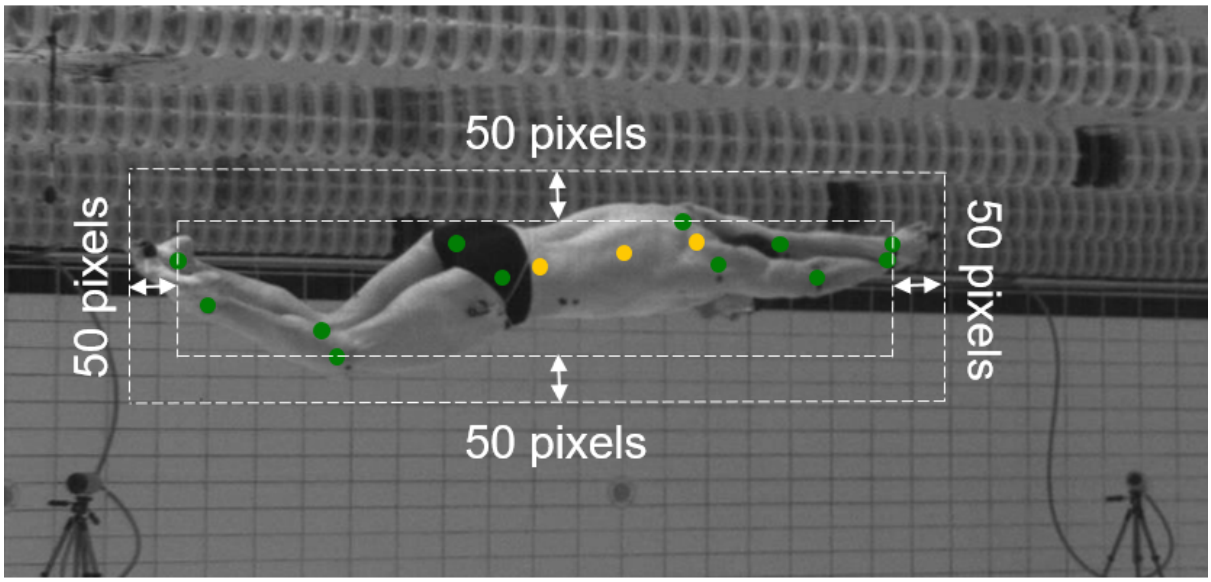


Figure 6.8: If the tightest bounding box possible is expanded by 50 pixels in all directions, for most images in the Charybdis dataset the resulting bounding box will fully enclose the swimmer. (Image created by the author)

box, it is possible to calculate PCK, but not PCKh. However, as discussed in Section 2.5.2, there is a relationship between PCK and PCKh. Given the examples discussed in Section 2.5.2, this relationship can be formalised mathematically as[8]:

$$PCK@\,0.025 \leq PCKh@\,0.5 \leq PCK@\,0.08 \tag{6.1}$$

Where on this spectrum PCKh lies depends on the pose of the person in the image: for images in which the largest dimension of the bounding box exceeds the height of the person[9], PCKh@0.5 will be closer to PCK@0.025; for images in which the largest dimension of the bounding box is shorter than the height of the person, PCK@0.5 will be closer to PCK@0.08[10]. In the images of Charybdis, the swimmers more often than not are in a fully outstretched position, with arms overhead. For such images, the largest dimension of the bounding box would exceed the height of the person, meaning if I wanted to use a PCK@α metric that was comparable to PCKh@0.5, I would need to use $\alpha = 0.025$. However, in most of the images of Charybdis, the swimmers are at an angle relative to the camera; therefore, even if the swimmer is in an outstretched position

[8]The upper bound of 0.08 was estimated using the example shown in Figure 2.19. In reality, this upper bound could be even closer to 1, for body configurations in which the spine is flexed.

[9]'Height of the person' here refers to the vertical distance (in pixels) between the feet and the head of a person, as calculated from a perfectly sagittal or frontal view.

[10]The justification for this statement is discussed in Section 2.5.2.

with arms overhead, it is possible, due to perspective, that the largest dimension of the resulting bounding box be shorter than the height of the person. Therefore, I estimated that, for most of the images in the Charybdis dataset, PCK@0.05 would approximate PCKh@0.5 reasonably well. For this reason, I chose PCK@0.05 as the metric with which to evaluate the reliability of human labellers.

The PCK@0.05 of each joint was computed separately for three levels of joint visibility: 'visible' (i.e. for a given joint of a given image, both labellers labelled the joint as 'visible = 1'); 'partly visible' (i.e. for a given joint of a given image, one labeller labelled the joint as 'visible = 1' and the other labeller as 'visible = 0'); 'occluded' (i.e. for a given joint of a given image, both labellers labelled the joint as 'visible = 0'). The 'partly visible' category may seem confusing: a joint is either occluded or it is not. However, as explained earlier in this section, labellers were instructed to label as visible joints whose bony landmarks were occluded but whose location could be estimated from context. However, different people likely have a different perception of where a joint that is not entirely visible could be (thus leading to them labelling different points), or of how sure they are of its possible location (thus leading to them giving the joint a discordant 'visible' label). Therefore, the 'partly visible' category captures those joints whose location *might* be evinced from context, but whose corresponding bony landmarks are occluded.

The results of this analysis are reported in Table 6.1[11], along with the percentage at which the three levels of visibility occurred for each joint. The last row of Table 6.1 confirms the intuition that occluded[12] joints cannot be labelled as reliably as visible joints. Interestingly, the left hip, shoulder, elbow, and wrist have higher PCK@0.05 when they are occluded than when they are visible; indeed, their PCK@0.05 when occluded is 1. This seeming contradiction might be due to the different sample sizes of the visibility categories. For example, only 27% of all left shoulders were occluded, while 57% of them were visible. Therefore, the 'visible' value of PCK@0.05 for the left shoulder is a more robust statistic than the 'occluded' value. Also interesting is that, while those joints (all of which are on the left side of the swimmers) are better

[11]The tables reported in most scientific papers on 2D pose detection, as well as the leaderboard on MPII's official website (http://human-pose.mpi-inf.mpg.de/#results), only feature the 'Overall' column shown in Table 6.1, thus not providing any information about the prevalence—and impact on accuracy—of occlusions.

[12]For the rest of this section, the word 'occluded' will refer to joints to which both labellers assigned a 'visible' value of 0.

Table 6.1: Inter-operator Reliability of 2D Pose Detection by Humans

	Visible		Partly Visible		Occluded		Overall
Joint Name	PCK@0.05	%	PCK@0.05	%	PCK@0.05	%	PCK@0.05
Left ankle	100.0	72	99.28	26	41.67	2	98.48
Right ankle	99.74	72	98.47	25	87.50	3	99.05
Left knee	100.0	54	100.0	36	99.65	10	99.24
Right knee	99.66	57	98.66	28	98.72	15	99.24
Left hip	99.60	47	98.78	31	100.0	22	99.43
Right hip	99.61	49	99.08	20	99.36	31	99.43
Left shoulder	99.66	57	97.67	16	100.0	27	99.42
Right shoulder	99.64	54	94.55	10	96.86	36	98.10
Left elbow	99.43	67	96.15	10	100.0	23	99.24
Right elbow	100.0	65	98.59	15	91.23	20	97.90
Left wrist	99.42	98	100.0	1	100.0	1	99.43
Right wrist	99.38	92	92.86	3	61.54	5	97.33
'Belly button'	-	0	100.0	1	87.98	99	88.00
Sternum	-	0	100.0	1	93.13	99	93.14
Neck	-	0	100.0	10	99.57	90	99.61
Overall	99.65	52	98.27	16	90.50	32	97.85

when occluded, the left ankle is worse when occluded. Indeed, occluded left ankles achieved the lowest PCK value of all joints under all visibility categories. Though this result could once again be explained by considering that the sample size was probably too small to make the statistic meaningful (since only 2% of left ankles were occluded), it might be interesting to see an example of an image for which the left ankle was labelled as occluded. Such an example is shown in Figure 6.9. In this case, the location of the left ankle is difficult to estimate, since neither the x nor the y coordinates of the joint are discernible from context.

Finally, Figure 6.10 illustrates the effect that the value of α in PCK@α has on the measure of the overall reliability of human labellers (i.e. on the value in the bottom-right corner of Table 6.1). This figure further illustrates that occluded joints cannot be labelled as accurately as visible joints: as the value of α decreases, the reliability of visible joints barely changes, while the reliability of occluded joints deteriorates almost exponentially.

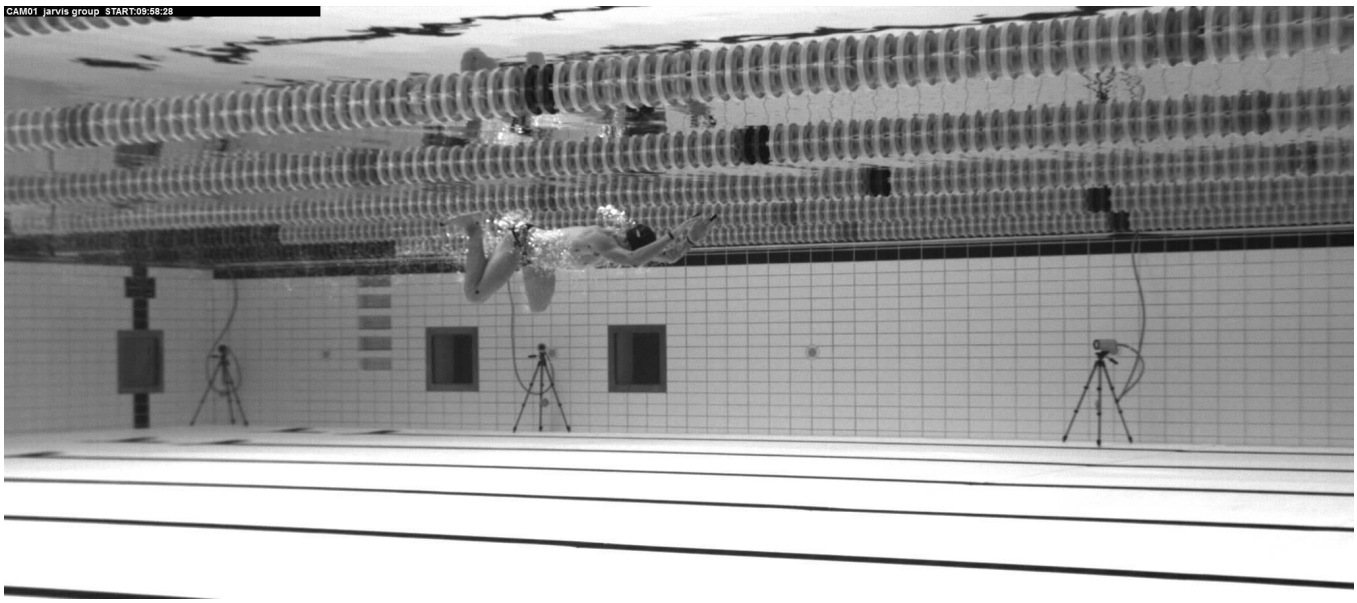


Figure 6.9: Example of an image for which the left ankle was labelled as 'visible = 0' by both labellers. (Image created by the author)

6.4 Conclusions

Even after having observed that the Scylla dataset did not contain enough images, to determine the ideal size of the Charybdis dataset I used the same heuristic used for Scylla: looking at the most popular datasets in the field (in this case, MPII and LSP), considering the difference in variability that these datasets are supposed to describe, and then estimating the number of images that should be in the dataset. This led me to estimate that 8,000 images (split into 80% for training and 20% of testing) should be enough for the Charybdis dataset—provided that they be more varied than the images in Scylla. To this end, a data capture session was held in the Manchester Aquatics Centre. During this session, new videos were recorded. From these videos, 2,113 images were extracted using the methodology described in Section 6.2. To these, 6,000 images from The Collection were added, for a total of 8,113 images.

The images in Charybdis were labelled by myself and an MSc student, using a custom-made labelling tool. The joints labelled were the ankles, knees, hips, shoulders, elbows, wrists, 'belly button', sternum, and neck, for a total of 15 joints. Each joint was also given a label indicating if the joint was visible ('visible' = 1) or occluded ('visible' = 0). Human reliability for 2D pose detection was estimated using the PCK@0.05 metric, which, in theory, should be equivalent to

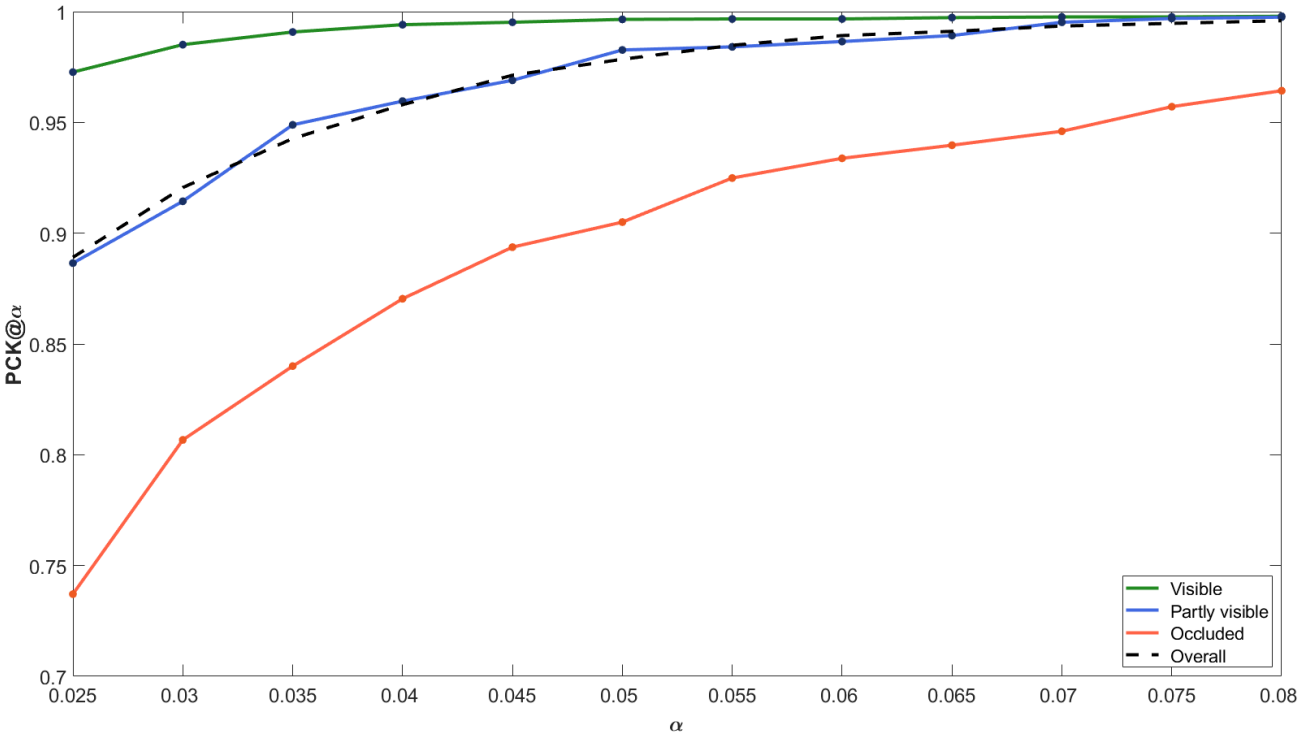


Figure 6.10: PCK@α between the two labellers of the Charybdis dataset, for different values of α, calculated for three categories of joint visibility.

the PCKh@@0.5 metric used to evaluate models on MPII. Separate values of PCK@0.05 were computed for three categories of joint visibility: visible, partly visible, occluded. The results of this analysis demonstrated that occluded joints are labelled less accurately than visible joints for all values α of PCK@α.

Chapter 7

Development of POSEidon, a 2D Pose Detection Algorithm for Images of Swimmers

7.1 Input Shape and Data Augmentation

Before a model could be developed and trained on the Charybdis dataset, two questions had to be addressed: what shape the inputs should have, and what kind of data augmentation should be used. As discussed in Section 2.5.3, most 2D pose detection algorithms crop images around the given bounding boxes, and then resize the cropped images to have a resolution of 256 x 256 pixels, thus almost invariably distorting the image (since the ratio of the bounding box might have been different from 1:1). In Section 2.5.3, an alternative was proposed: to avoid images being distorted, they could be resized to the mean resolution of the bounding boxes in the dataset. For Charybdis, the mean resolution of the bounding boxes was about 256 x 512. Therefore, all images in Charybdis were first cropped around the bounding boxes obtained as described in Section 6.3, and then resized to have a resolution of 256 x 512 pixels.

For 2D pose detection, the types of data augmentation used most often are horizontal

flipping, rotation, and scaling. Rafi et al. [170] were the only authors who quantified the contribution of these data augmentation techniques to the performance of 2D pose detection algorithms—a contribution which they estimated to be about 4% PCKh for their network. However, since neural networks require inputs of a pre-determined, rectangular shape, rotating or scaling images means that either parts of the images would be cut off, or the image would need to be down-sampled to fit the desired resolution (see Figure 7.1); in both cases, information would be lost. Therefore, rotation and scaling, though popular augmentation strategies,

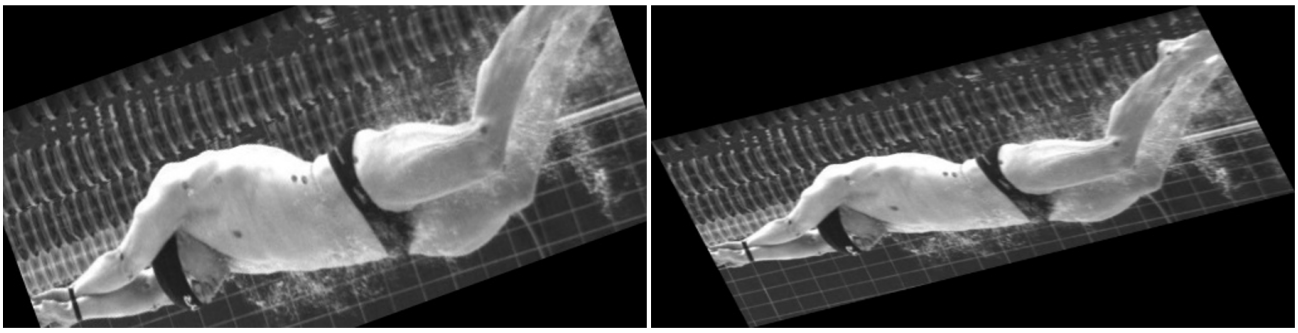


Figure 7.1: Example of the two types of results of rotating an image while under the restriction of maintaining a fixed resolution (in this case, 256 x 512 pixels): the image loses information either because parts of it are cut off (left image) or because it is down-sampled (right image). (Image created by the author)

are likely ineffective. However, just using horizontal flipping is unlikely to impact the performance of a neural network. To this end, some researchers [172, 195] have developed adversarial augmentation techniques[1] specifically for 2D pose detection—for example, by pasting random body parts at random places in images. Bin et al. [195], who trained Stacked Hourglass using this data augmentation strategy, obtained 94.1% PCKh, which, as of the time of writing, puts their model at the top of MPII's leaderboard. However, as the database they used is not publicly available and would be time consuming to obtain, it could not be used in this PhD. Therefore, new ways to augment the images in Charybdis (other than horizontal flipping) had to be devised.

The purpose of data augmentation is to expose algorithms to more variability than is present in the training set of a given dataset, to make the algorithm learn a more generalisable probability distribution. A simple way to understand how to perform data augmentation is to

[1] See Section 2.5.5.4 for more details.

ask the question: 'If we gathered new data under different conditions, how would we expect the new data to differ from the data already gathered?'. If the sources of variability to be expected in new data can be traced back to linear transformations (e.g. rotations, changes in colour, etc), applying these transformations to the training data available is likely to increase the performance of the model. Therefore, the best way to augment the images in Charybdis is to apply transformations to them that might occur naturally in new images. As discussed in Section 4.1, the main sources of variability for images of underwater swimmers relate to the conditions under which the videos are captured: images could be more or less out of focus; the subject being recorded might be blurred due to the camera having a low shutter speed; the brightness of the image could vary because of several factors. Therefore, I decided to adopt the following three data augmentation techniques (see Figure 7.2): motion blur (which could be caused by an improperly set shutter speed); Gaussian blur, (which could be caused by an improperly set camera focus); and altered brightness/contrast (which could be caused by an improperly set camera aperture or shutter speed, but also by differences in lighting that naturally occur across different swimming pools, or within the same swimming pool due to changes in external lighting).

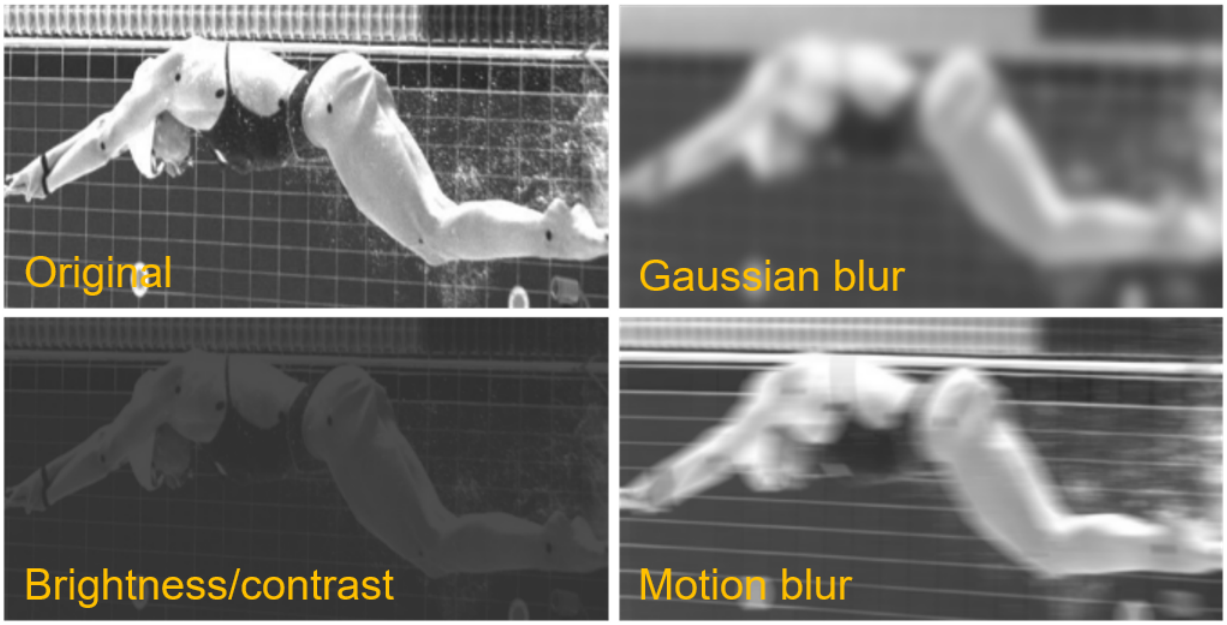


Figure 7.2: Examples of the types of swimming-specific data augmentation developed to train 2D pose detection algorithms on Charybdis. Each of these transformations is likely to be found naturally when recording images of swimmers. (Images created by the author)

Gaussian blur was implemented by convolving images with a Gaussian filter with a parameter $\sigma \in [1, 10]$ (where higher values give more blur)[2]. Motion blur was implemented by convolving images with a square filter of size $\in [15, 30]$ (where higher values give more blur) in which the middle row of pixels had a value of one and all others were zero. Brightness and contrast were altered by adapting the code from a public code repository[3]. In this implementation (the details of which can be found in the introduction to the repository), there are two ranges that determine the magnitude of the transformation: one for brightness (range: [100, 350], where lower values give darker images), and one for contrast (range: [100, 150], where higher values give more contrast). Finally, horizontal flipping was also used. The probability of performing each type of augmentation on a given training image was set to 50%, meaning there was a 6.25% chance that all four augmentations be applied to a given image.

7.2 Model Development

In Section 2.5.4 it was discussed how heatmaps (see Fig 2.22) are used as labels in most modern 2D pose detection algorithms. However, using heatmaps requires models to output predictions at resolutions as close as possible to the input resolution. This is because, as noted in Section 5.4, models whose outputs need to be up-sampled to match the resolution of the input tend to perform worse than models whose outputs have the same resolution as the input, because the down-sampling operation is non-reversible (meaning some information is lost that cannot be recovered via up-sampling). For deep models, this can be too computationally expensive to make training practical: even Stacked Hourglass [5], whose inputs have a resolution of 256 x 256 pixels (16 times smaller than the inputs used by FISHnet, and twice as small as the crop-resized images in Charybdis), first has to lower the resolution of the input to 64 x 64 pixels to make training manageable. This means that the heatmaps outputted by Stacked Hourglass (and by most of the algorithms described in Section 2.5.5, which derive from Stacked

[2]Data augmentation techniques are implemented as statistical distributions (usually a Gaussian or uniform distribution) of the value that determines the magnitude of the transformation. For example, if rotation is used, when an image is fed to a model it is first rotated by a random angle that is usually in the range [-35; 35] [170].

[3]https://www.life2coding.com/change-brightness-and-contrast-of-images-using-opencv-python/

Hourglass) are 16 times smaller than the input, and need to be up-sampled when calculating PCKh, likely leading to errors. Therefore, instead of using heatmaps, it might be preferable to use a model that directly predicts the x and y coordinates of the joints (i.e. a regression model). Whereas the outputs of Stacked Hourglass would have shape *image_height*/4 x *image_width*/4 x *num_joints*, the outputs of a regression model would have shape *num_joints* x 2. This type of output has two advantages. First, since it allows models to be smaller (as they would not need to compute the transformation from low resolution features to high resolution heatmaps) it makes the training of the neural network faster, making it possible to test more architectures and to fine-tune the hyperparameters of the model more precisely. Second, the coordinates predicted by a regression model are in the reference system of the input image, which effectively means they have the same 'resolution' as the input. Therefore, no information would be lost at any stage of the prediction process. Because of these two reasons, I decided to develop a regression model instead of a model that uses heatmaps as inputs. Though this choice differs from the trend in the literature, it is also true that regression-based deep learning models have never been compared to heatmap-based models; indeed, modern regression models have never even been applied to 2D pose detection. Therefore, the assumption that heatmap-based models are by default superior to regression models for this task may be misguided.

Regression models can be seen as being just the encoder part of encoder-decoder models like U-Net: they condense large inputs (in this case, images) into small, semantically rich features (in this case, of shape 15 x 2). One of the simplest (but complex enough to realistically be able to perform well) regression models to implement is U-Net's encoder, which is simply a series of convolutional, batch normalisation, and average pooling layers. Therefore, it was the first model that was trained on Charybdis[4]. In this early stage of model development, models were trained only for 50 epochs: empirically, 50 epochs were found to be a good compromise between training algorithms to near-optimal values and being able to quickly test several algorithms[5]. After 50 epochs, and using the data augmentation techniques described in the previous section,

[4]This model was not expected to perform well on Charybdis. The rationale behind its implementation was to begin with the simplest model possible, and progressively add complexity, to understand what kind of algorithmic changes affected performance the most.

[5]In the initial stages of model development, it is beneficial to quickly test a wide range of architectures and identify which ones work best; afterwards, the best-performing models can be fine-tuned to improve their performance.

U-Net's encoder achieved a PCK@0.05 of 62.63% on the training set, and of 35.28% on the test set[6]. These results indicate that U-Net's encoder not only overfits badly, but also does not have enough capacity to reach good performance on the training set.

The second algorithm tested was the encoder in Stacked Hourglass's hourglass modules, whose main difference from U-Net's encoder is that it uses residual modules instead of simple series of convolutional and batch normalisation layers. As expected, this more refined encoder achieved higher performance than U-Net's encoder, with a training PCK of 77.96% and a test PCK of 54.39%. At this stage, the concept of stacking modules was introduced. However, since the output of the model had to be of shape *num_joints* x 2 (i.e. the shape at the end of an encoder, whereas the shape at the end of a decoder is the same as the input), and since the first stack was just an encoder, each stack after the first one had to be constructed as a decoder-encoder pair (see Figure 7.3). This structure was used to train two-, three-, and four-stack models (which will be referred to as 'n-stack Truncated Hourglass models'[7]), the results for which are reported in Table 7.1. As in the original implementation of Stacked Hourglass, intermediate loss functions were added at the end of each stack, to make training faster and less susceptible to vanishing gradients. For all models, and for each of their stacks, the loss function used was Mean Squared Error, which is the loss function used in Stacked Hourglass and in most other 2D pose detection algorithms.

Table 7.1: Results for Truncated Hourglass models with different numbers of stacks

Number of stacks	Training PCK@0.05 (%)	Test PCK@0.05 (%)
1	77.96	54.39
2	82.19	**57.43**
3	75.87	52.38
4	63.17	39.21

The results in Table 7.1 indicate that the performance of Truncated Stacked Hourglass improves if a second stack is used, but deteriorates if further stacks are added. The cause of this

[6]For simplicity of notation, for the remainder of this chapter a model's performance on the training and test sets of Charybdis will be denoted as 'training PCK' and 'test PCK', respectively. Likewise, these initial tests did not differentiate between PCK values for visible and visible joints. Such a fine-grained analysis of the performance of models was beyond the objective of this initial stage of algorithm development, which was to quickly test several architectures.

[7]If the word Hourglass refers to a model, it is capitalised; if it refers to a module, it is not.

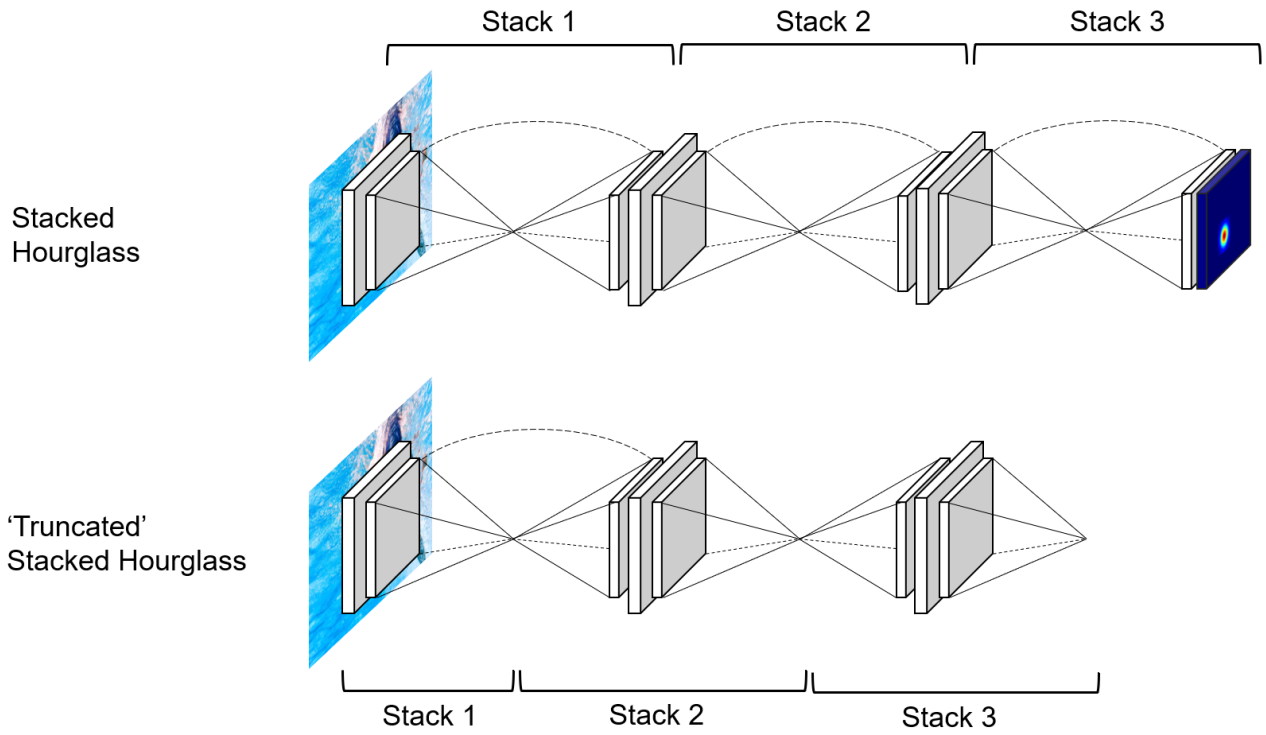


Figure 7.3: Example of using three stacks to build a Stacked Hourglass model and a 'Truncated' Stacked Hourglass model. Stacking encoders requires that each stack after the first be constructed as a decoder-encoder pair, whereas in a normal Stacked Hourglass network each stack is an encoder-decoder pair. For the 'Truncated' Stacked Hourglass, the tip of each encoder has shape 15 x 2. (Image created by the author)

phenomenon cannot be overfitting caused by the extra capacity introduced by the additional stacks, as the training PCK also deteriorates as more stacks are added. It is possible that, even though intermediate losses were used, the gradients they provide are too weak to prevent the gradients of later stacks from vanishing (which is a consequence of using a regression model instead of a heatmap-based model). Whatever the reason for this behaviour, it is also interesting to understand why adding a second stack improves performance. In Stacked Hourglass, adding more stacks was beneficial because each stack further refined the prediction of the previous one. The evidence that this was the mechanism by which adding stacks improved performance was that the loss of a given stack i was always lower than the loss of stack $i + 1$. This behaviour was observed also in the two-stack Truncated Stacked Hourglass model reported in Table 7.1, for which, after 50 epochs, the loss of the first encoder was 578 and the loss of the second encoder was 449. Therefore, adding stacks seems to have the same effect regardless of the type of output used by the model.

A limitation of using hourglass modules is that it was not possible to use pre-trained weights for them, as a pre-trained model would have a structure (and supervision) different from that needed for a regression-based model[8]. As discussed in Section 5.2, pre-training a model tends to increase performance. Indeed, when developing FISHnet it was observed that the factor that contributed to performance the most was using a pre-trained encoder (VGG16). Therefore, re-implementing the Truncated Stacked Hourglass model using powerful pre-trained encoders would likely improve the performance of the model. However, using pre-trained encoders would mean losing the symmetry between encoders and decoders, which is one of the key features of Stacked Hourglass. But is this feature needed in a regression-based model? The purpose of having symmetrical encoders and decoders is to link (via skip connections) layers of the encoder with layers of the decoder at the same depth. The purpose of these skip connections, as discussed in Section 5.1, is to allow the decoder to restore the spatial resolution of its features. In models that use heatmaps as outputs, this feature is essential, otherwise the model will output inaccurate, low-resolution predictions. But in a regression-based model, the output has no spatial resolution, and therefore spatial resolution does not need to be restored. Therefore, for a regression-based model not only is it unnecessary for the encoder and the decoder to be symmetrical, but it is also probably unnecessary to use decoders altogether[9], since their function is irrelevant for regression-based models. Therefore, instead of stacking pre-trained encoders in 'truncated hourglass' fashion, they could be stacked as in Figure 7.4: the features at the bottom of each encoder are up-sampled by a factor that gives them the same resolution as the first layers of the next encoder, to which they are connected. This architecture is lighter than that of a Truncated Hourglass, because no parameters are required to connect the encoders to one another. Since the shape of this type of network no longer resembles a series of hourglasses, the name 'Cascaded Encoders' will be used to refer to this type of network.

Three types of pre-trained encoder were tested as the encoders for one-, two-, and three-stack Cascaded Encoders networks (for a total of nine models): ResNet50V2 [217], DenseNet201 [12], and EfficientNetB3 [218][10]. Four-stack Cascaded Encoders networks could not be tested:

[8]This is the same reason why FISHnet could not be re-purposed for this task.

[9]This hypothesis will be tested later in this section.

[10]VGG16 was not tested because, when these tests were conducted, its performance had been surpassed by the other encoders listed.

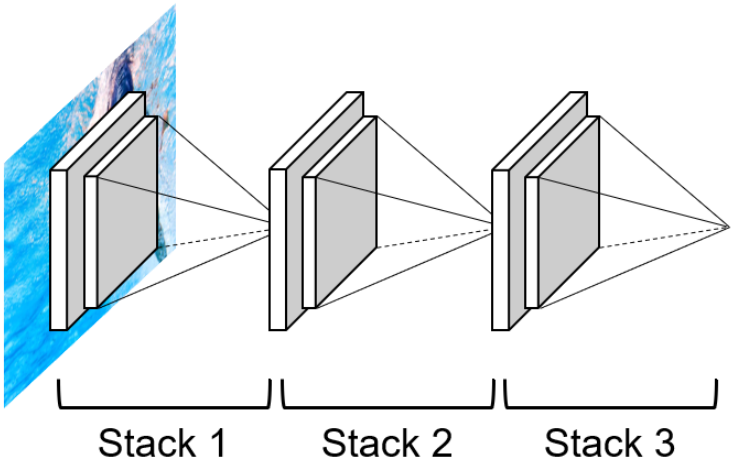


Figure 7.4: Example of the architecture of a Cascaded Encoders network. (Image created by the author)

their number of parameters is so high that the models would not have fit on the GPUs available during this PhD[11]. As for the models tested previously, these models were trained using MSE as the loss function and the data augmentation techniques described in the previous section, and they were only allowed to train for 50 epochs. The results of these tests are reported in Table 7.2.

Table 7.2: Results for Cascaded Encoders models with different numbers of stack

Encoder type	Number of stacks	Training PCK@0.05 (%)	Test PCK@0.05 (%)
ResNet50V2	1	94.54	68.99
	2	94.11	72.04
	3	93.14	68.83
DenseNet201	1	95.76	71.63
	2	94.74	**75.32**
	3	92.33	65.41
EfficientNetB3	1	86.15	71.31
	2	86.46	73.60
	3	80.38	72.38

Two observations can be made regarding the results in Table 7.2. First, even for Cascaded Encoders networks, two stacks give the best performance. Once again, this finding can be

[11]A four-stack Cascaded Encoders network with ResNet50V2 as the encoder would have 100 million parameters, making it more than eight times larger than a four-stack Stacked Hourglass network. Even on a powerful GPU, training a model so large would take several days.

attributed to the idea that regression-based deep supervision (i.e. the addition of intermediate losses after each encoder) likely does not provide a strong enough gradient to prevent it from vanishing as it reaches the earlier stacks of a three-stack network. Second, for all configurations tested except three-stack networks, DenseNet201 was the encoder that performed best. Therefore, all subsequent tests were done using a two-stack Cascaded Encoders network in which the two encoders were pre-trained DenseNet201 models (a model which will be referred to as 'POSEidon').

To test the hypothesis that decoders are unnecessary in a multi-stack regression model, a two-stack Truncated Stacked Hourglass model was implemented using DenseNet201s as encoders and building decoders with the same layer arrangement of a DenseNet201 (but in which the average pooling layers were replaced by upsampling layers). This model achieved a training PCK of 94.27% and a test PCK of 71.81%—3.5% worse than the test PCK achieved by the two-stack Cascaded DenseNet201s network. This result agrees with the hypothesis that decoders are unnecessary for regression models, but does not necessarily confirm it. Although the decoders used to train the symmetrical two-stack Truncated Stacked Hourglass had a similar structure to the encoders, only the encoders had pre-trained weights (since the decoders were implemented by hand). Therefore, it is possible that, had the decoders been pre-trained as the encoders were, using decoders might have improved performance. Nevertheless, the simpler interpretation is that decoders are unnecessary for regression-based models.

7.3 Final Architecture of POSEidon, and Implementation Details

The final architecture of POSEidon (Figure 7.5) consists of two DenseNet201 networks stacked in series[12]. The two networks are connected to each other by upsampling the bottom of the first until it has the same resolution as the first layer of the second, to which it is then merged.

[12]The two DenseNet201 networks were not implemented by hand. Instead, TensorFlow was used to download and instantiate them, using weights pre-trained on ImageNet.

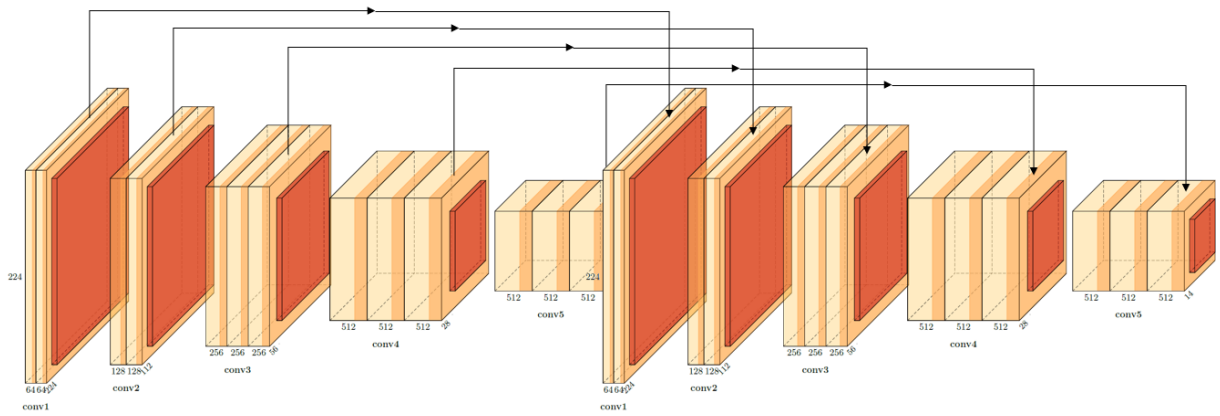


Figure 7.5: Final architecture of POSEidon, which consists of two DenseNet201 networks stacked in series, and whose blocks are connected via skip connections. For the two DenseNet networks represented here, the skip connections between blocks within the same network are not represented to avoid confusion with the skip connections between blocks of the two networks. For a full description of DenseNet201's architecture, please refer to [12].

Because it is a regression-based model, POSEidon does not need a decoder between its two encoders, thus saving parameters.

POSEidon was trained using the following data augmentation techniques: Gaussian blur ($\sigma \in [1, 10]$), motion blur (kernel size $\in [15, 30]$), and altered brightness/contrast ($\alpha_c \in [100, 150], \alpha_b \in [100, 350]$); the values of each augmentation technique were sampled using a uniform distribution. POSEidon was trained with a batch size of four and with early stopping, meaning training was forced to stop once the validation loss had not decreased by at least 0.0001 for six consecutive epochs. The optimiser used was RMSprop with an initial learning rate of 0.001, which was annealed by a factor of ten if the validation loss did not decrease for five consecutive epochs. POSEidon was implemented using Keras with TensorFlow backend. All training and testing was done on an Nvidia Titan X GPU.

7.4 Evaluation

To evauate the performance of POSEidon, it was re-trained on Charybdis until convergence (which took 94 epochs). The results of this test are reported in Table 7.3, in which separate values of PCK@0.05 are reported for visible and occluded joints. The performance of POSEidon

improved only marginally when it was allowed to train until convergence, compared to when it was allowed to train for only 50 epochs. This finding confirms the hypothesis that evaluating models after training them for only 50 epochs did provide a good indication of the true performance capacity of the models. One result in Table 7.3 that is surprising is that POSEidon performs better on occluded joints than on visible ones. Indeed, by retrospectively calculating the training and test PCK@0.05 for visible and occluded joints of all previously tested models, it was found that all models performed better on occluded joints than on visible ones. A likely explanation for this phenomenon is that one (or more) of the data augmentation techniques used was not working as intended. To test this hypothesis, POSEidon was re-trained without using any data augmentation. The results of this test (reported in Table 7.4) indicate that indeed one or more of the data augmentation technique used were responsible for POSEidon performing worse on visible joints than on occluded ones, and therefore performing worse overall. To test whether the problem resided in one of the three novel data augmentation techniques implemented (Gaussian blur, motion blur, and altered brightness/contrast), POSEidon was re-trained once using only horizontal flipping, and once using only the three novel augmentation techniques. The results for this test (reported in Table 7.5) are surprising: they indicate that horizontal flipping, not the novel augmentation techniques, is responsible for the decrease in performance on visible joints. No explanation was found for this counter-intuitive result. A second result from Table 7.5 that should be highlighted is the fact that using the three novel augmentation techniques improved performance by 1.76% over the performance obtained with no data augmentation (Table 7.4). This confirms the hypothesis that task-specific data augmentation techniques are more beneficial than generic ones—perhaps even more so than what the results in Table 7.5 suggest. The purpose of data augmentation is to teach an algorithm how to output accurate predictions even for challenging images. However, most of the images

Table 7.3: Results for POSEidon on Charybdis

Visibility	Training PCK@0.05 (%)	Test PCK@0.05 (%)
Visible	98.28	76.35
Occluded	98.41	79.86
Overall	**98.34**	**77.73**

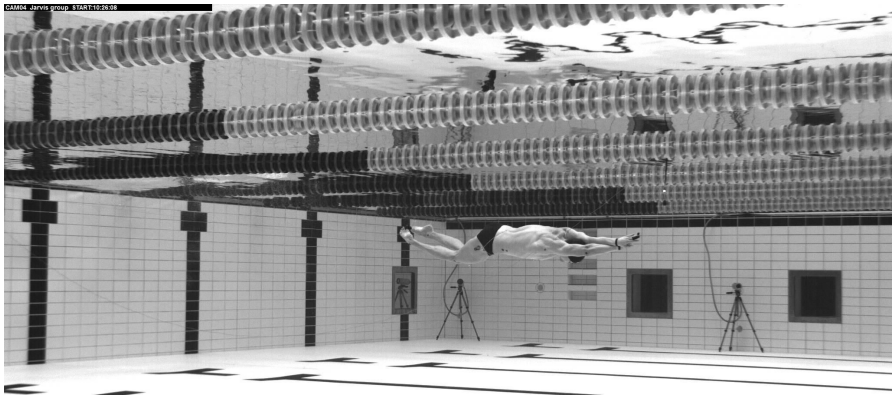
Table 7.4: Results for POSEidon (no augmentation)

Visibility	Training PCK@0.05 (%)	Test PCK@0.05 (%)
Visible	99.01	81.88
Occluded	99.04	79.21
Overall	**99.02**	**80.83**

Table 7.5: Results for POSEidon with different augmentation strategies

Augmentation	Visibility	Training PCK@0.05 (%)	Test PCK@0.05 (%)
Horizontal flipping	Visible	96.57	75.02
	Occluded	97.27	79.91
	Overall	**96.86**	**76.94**
Gaussian blur, motion blur, altered brightness/contrast	Visible	99.18	83.18
	Occluded	99.12	81.68
	Overall	**99.15**	**82.59**

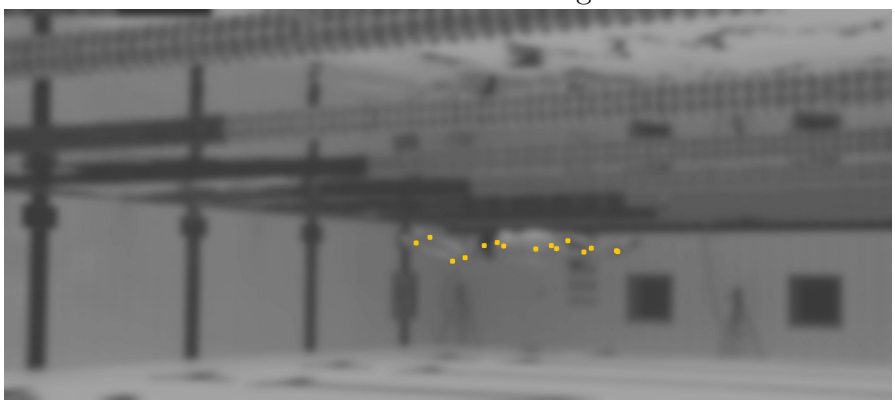
in the test set of Charybdis are quite 'clean'—for instance, there is minimal blurring. Therefore, the true benefit of using data augmentation does not emerge from the results in Table 7.5 because such sources of variability are not abundant in the test set of Charybdis (although they might be in newly recorded data). To illustrate this point, I decided to augment the test set of Charybdis using the same three task-specific techniques described in the previous section. Each test image was augmented with at least one of the three techniques, and the probability that an image would be augmented by all three was set to 12.5% (to maintain the same probability with which the training data had been augmented). Two versions of POSEidon were then re-tested on this augmented test set: the version that had been trained with no data augmentation, and the version that had been trained with the three task-specific data augmentation techniques (but without horizontal flipping). The results of this test, reported in Table 7.6, demonstrate the true benefit of using these task-specific augmentation techniques: on the augmented test set, POSEidon is about 20% more accurate if it is trained using data augmentation. As shown in Figure 7.6, training POSEidon with these data augmentation techniques enables it to perform well even on images that would be challenging for humans to label. Therefore, POSEidon would likely perform well under less ideal conditions than those under which the images in Charybdis were recorded—for example, it would likely perform equally well on images recorded by lower-quality cameras, and whose parameters are not tuned optimally. Furthermore, Figure



Original test image

Augmented test image (Gaussian blur + lowered brightness)

POSEidon trained without augmentation



POSEidon trained with augmentation

Figure 7.6: Training POSEidon with task-specific data augmentation techniques makes it more robust against variations in the data that can be found naturally in images of swimmers.

Table 7.6: Results for POSEidon on augmented test data

Augmentation	Visibility	Test PCK@0.05 (%)
None	Visible	63.85
	Occluded	59.62
	Overall	**61.28**
Gaussian blur, motion blur, altered brightness/contrast	Visible	82.15
	Occluded	80.81
	Overall	**81.63**

7.6 shows that not using these data augmentation methods to train POSEidon results in it failing on challenging images.

Finally, Table 7.7 reports the per-joint PCK@0.05 of POSEidon (trained with data augmentation and tested on the non-augmented test set of Charybdis). There are a few noticeable differences between the per-joint accuracy of POSEidon and the per-joint inter-operator reliability reported in Table 6.1. First, POSEidon performed worst on occluded right wrists, only about 17% of which were correctly identified. Occluded right wrists were difficult to label also for humans, who achieved 61% PCK@0.05 on them. Therefore, given that the labels for these joint are unreliable and that right wrists are occluded in only about 6% of the images in the training set of Charybdis, it is unsurprising that POSEidon could not learn well how to detect these joints. More difficult to interpret are the low PCK values for the visible wrists and left ankle—for which the gap between POSEidon and humans was about 24% on average. No explanation for this behaviour could be found. Second, POSEidon performed best on the spinal joints, which it located only about 4% less reliably than humans (whereas for the other joints the mean gap between humans and POSEidon was about 18%). As discussed in Chapter 6.1, the spinal joints are especially important when fitting a parametric model to 2D joints, as they provide constraints with which the trunk can quickly be placed in the correct position. Once the trunk has been positioned correctly, the orientation of the limbs can be estimated using the constraints provided by the other joints and by the silhouettes. As POSEidon is about as accurate as humans at identifying spinal joints, it could function well in a 2D-to-3D pipeline.

Table 7.7: Per-joint PCK@0.05 of Poseidon on Charybdis

	Visible		Occluded		Overall
Joint Name	PCK@0.05	%	PCK@0.05	%	PCK@0.05
Left ankle	75.61	82	87.28	18	77.64
Right ankle	86.05	85	82.61	15	85.57
Left knee	80.58	66	87.50	34	82.92
Right knee	82.37	78	80.11	22	81.88
Left hip	85.97	37	88.75	63	87.71
Right hip	82.14	50	77.14	50	79.67
Left shoulder	86.13	65	83.63	35	85.26
Right shoulder	87.63	63	56.32	37	76.23
Left elbow	81.04	77	68.51	23	78.26
Right elbow	85.05	83	50.74	17	79.36
Left wrist	75.30	91	58.45	9	73.83
Right wrist	77.42	94	17.24	6	74.20
'Belly button'	90.79	4	88.14	96	88.27
Sternum	93.33	5	87.58	95	87.90
Neck	92.59	23	92.64	77	92.63
Overall	83.18	60	81.68	40	82.59

7.5 Conclusions

This chapter has shown that it is possible to use regression models for 2D pose detection. Such models have two advantages over heatmap-based models: they do not require decoders (as demonstrated by the results reported in this chapter), meaning more computational power can be allocated to the encoders; and they function equally well on images of all sizes, since their output does not need to have the same resolution as the input.

The algorithm developed, called POSEidon, was trained using a novel data augmentation strategy, which sought to address sources of variability that might impact images of underwater swimmers more meaningfully than by small rotations or translations. Using these novel data augmentation techniques increased the performance of POSEidon by about 2% on the test set of Charybdis. This effect was increased to 20% when POSEidon was tested on an artificially augmented version of the test set of Charybdis, showing that training POSEidon with these

novel data augmentation techniques makes it usable even on images recorded under sub-optimal conditions. Furthermore, as POSEidon is about as accurate as humans at identifying spinal joints, it could function well within a markerless motion capture system.

Chapter 8

Conclusions, Limitations, and Future Work

To improve the performance of athletes close to their genetic potential, tools are needed for studying their technique in fine-grained detail. Such tools are routinely used in many sports by researchers (to study the aspects of technique most related to good performance) and coaches (to closely monitor the progress of their athletes). However, currently no such tool can accurately and conveniently extract the 3D kinematics (i.e. parameters that relate to technique) of swimmers. When used to record the movements of swimmers, established motion capture systems have limitations that make them impractical: sensor- and marker-based systems increase the drag of the swimmer, thus reducing the validity of the analysis; depth-based systems are too inaccurate and impractical for underwater use; and manual digitisation is too slow to be practical. The best solution for swimming motion capture would be a markerless system that only needs a few cameras. Such a system would first extract 'intermediate' information (i.e. silhouettes and 2D joint locations) from the videos recorded by the cameras, and then fit a generic 3D model to these constraints. One of the simplest systems of this kind is the visual hull, which only requires the silhouettes of an object (from all available camera views) to reconstruct its volume. However, as shown in Chapter 3, the visual hull relies on the silhouettes extracted being near perfect, which is currently unachievable in a swimming pool. Furthermore,

research on image-based markerless motion capture systems has shown that using a combination of silhouettes and 2D joints as constraints to which to fit a parametric model gives better results than using only silhouettes.

Before this PhD it was not possible to automatically (and accurately) extract from images the silhouettes and 2D joint locations of swimmers, as the algorithms available for these tasks are trained using datasets that do not contain images of swimmers. One of the main outcomes of this PhD was the construction of two datasets of images of swimmers: Scylla (3,100 images), to train algorithms for silhouette extraction; and Charybdis (8,000 images), to train algorithms for 2D pose detection. These datasets were constructed with the aim of covering as many sources of variability as possible, to allow algorithms trained on them to generalise well to new images. The construction of the Scylla and Charybdis datasets (which corresponds to Objectives 2 and 4 of this PhD) enabled the development of new, swimming-specific algorithms for silhouette extraction and 2D pose detection. Indeed, it is not optimal to simply re-train on Scylla and Charybdis general-purpose, off-the-shelf algorithms. Such algorithms are designed to perform well on general-purpose datasets, which typically have lower resolution images than those used within a markerless motion capture system. Therefore, such algorithms are fundamentally unsuited to perform well on Scylla and Charybdis. For example, the results reported in Section 5.4 showed that DeepLabv3+, the best-performing algorithm for general-purpose silhouette extraction, outputs coarse silhouettes when trained on Scylla. This is because DeepLabv3+'s output, which is 16 times smaller than its input, needs to be heavily upsampled, thus losing information.

To address the limitations of existing methods, two new algorithms were developed during this PhD: FISHnet, for silhouette extraction (Objective 3); and POSEidon, for 2D pose detection (Objective 5). One of the defining (and novel) characteristics of FISHnet is that it outputs silhouettes at a resolution of 1024 x 1024 pixels—16 times higher than DeepLabv3+. This allows FISHnet to output more fine-grained silhouettes than DeepLabv3+, even though FISHnet is a simpler, lighter model. FISHnet also includes two new modules (the modified Semantic Embedding Branch and the Spatial Resolution Enhancer module), whose function is to improve the network's intra-connectivity. FISHnet performed as accurately on Scylla as

humans, indicating that it could be used effectively within a markerless motion capture system.

POSEidon, the second algorithm developed during this PhD, is the first 2D pose detection algorithm based on deep learning that directly regresses the x and y coordinates of joints, rather than first outputting heatmaps. In other words, whereas most 2D pose detection algorithms output predictions of shape *img_height* x *img_width* x *num_joints*, POSEidon outputs predictions of shape *num_joints* x 2. This allows POSEidon to discard the network components that are typically used to output high resolution predictions, enabling it to devote more computational power to adopting complex pre-trained encoders, such as DenseNet201. A further novelty in the way POSEidon was trained lies in the data augmentation techniques that were used. Rather than using generic data augmentation techniques such as horizontal flipping (which was found to actually be detrimental to POSEidon's performance) or random rotations, POSEidon was trained using bespoke, task-specific data augmentation techniques, which randomly modified the images by adding to them Gaussian and motion blur, and altering their brightness and contrast. The results in Section 7.4 showed that using these data augmentation techniques during training improved POSEidon's performance by about 2% when tested on 'clean' images, and by about 20% when tested on challenging images; this indicates that task-specific data augmentation is much more effective than generic data augmentation. Importantly, POSEidon is almost as accurate as humans at locating the spinal joints of swimmers. Since these joints are particularly important constraints when fitting a 3D model to 2D joints, it can be concluded that POSEidon would fit well within an image-based markerless motion capture system.

8.1 Limitations and Future Work

The development of FISHnet and POSEidon has laid the foundations on which an accurate, fast, automatic markerless motion capture system for underwater use can be developed. The final step to obtain such a system is to design an algorithm that can fit a parametric model to the constraints provided by FISHnet and POSEidon (i.e. silhouettes and 2D joints). A likely

candidate for such an algorithm is the Iterative Closest Points (ICP) algorithm, which was used in the past for similar tasks [22]. This work, which is left to future research, would likely need to follow these steps:

1. Either use the videos recorded during this PhD, or record (and calibrate) new videos;

2. Use the calibration files of each camera to obtain the internal and external camera parameters (needed for the ICP algorithm);

3. Use FISHnet and POSEidon to extract silhouettes and 2D pose from each frame of each video;

4. Use ICP to fit a parametric model to the silhouettes and 2D pose, one frame at a time. The optimisation process should begin by relying more on the 2D joints (in particular the spinal ones), which would allow to quickly position the 3D model in a roughly correct pose. Then, the pose and shape of the 3D model could be refined by relying more on the silhouettes;

5. Validate the system by using manual digitisation as a gold standard.

Furthermore, future research should address some of the limitations of Scylla, Charybdis, FISHnet, and POSEidon that have been discussed throughout this thesis. In particular, although Scylla and Charybdis were constructed to be as varied as possible, some sources of variability could not be included—namely, neither dataset contains images of dark-skinned swimmers; future research should supplement such images (along with corresponding labels). Furthermore, during the development of FISHnet it was discovered that Scylla—and in particular its test set—might not be large enough for FISHnet to express its full capacity. Therefore, future research should expand the Scylla dataset by gathering more images from as many swimming pools and swimmers—and under as varied recording conditions—as possible, to add more data to it and to increase its variability. This would likely improve FISHnet's performance even further, by enabling the SEB and SRE modules to express their full capacity.

Similarly, researchers should take steps to improve POSEidon's performance, which, though likely high enough to perform well within a markerless motion capture system, is still about 10% below that of humans. One way in which POSEidon could be improved would be to adopt more powerful encoders than DenseNet201, or to devise a novel way to perform intermediate supervision, so that more than two stacks could be used without the gradients vanishing. Futhermore, POSEidon requires a tool to automatically detect the bounding boxes of swimmers from images—a tool that currently is unavailable. Therefore, before POSEidon can be used within a markerless motion capture system, such a tool must be developed.

Finally, researchers should explore the option of using multiple frames to improve the performance of both models. Multiple frames could be used:

- as a post-processing step, to smooth out any inconsistencies in FISHnet and POSEidon's prediction of consecutive frames. This would have the effect of forcing the predictions to follow a constraint of temporal consistency;
- directly during the training of FISHnet and POSEidon, feeding them, for example, n frames at each training step (instead of one). This would directly embed in the models a notion of temporal consistency between frames, likely resulting in improved results. However, this procedure would require considerable computational power.

Finally, whereas the application of Scylla and Charybdis is limited to the domain of underwater swimming, FISHnet and POSEidon have no such limitation. Indeed, they could function well within markerless motion capture systems for other sports, provided that they be trained on task-specific datasets, or as standalone algorithms for silhouette extraction and 2D pose detection, if trained on large, generic datasets. The main feature of both FISHnet and POSEidon is that they can perform well even on large images, a feature which is useful universally. Therefore, an additional route that future research could take is to adapt these algorithms to domains different from underwater swimming.

Appendix A

Marker-based Systems

Formally known as optoelectronic[1] stereophotogrammetric[2] systems, or OSSs, marker-based systems consist of a variable number of cameras (3-20) and of spherical markers of variable dimension (3-25 mm) that are attached to the surface of the object whose motion is to be captured. In the most popular OSSs, the markers passively reflect infrared light emitted by a crown of LEDs coaxial with the camera; the reflection is then detected by the optoelectronic sensors inside the cameras; the position in space of the markers is then reconstructed automatically via stereophotogrammetry by the software that comes with the OSS. For human motion capture, the spherical markers are attached on anatomical landmarks from whose position it is possible to reconstruct—using standard formulas and anthropometric measurements—the position of the joint centres [32]. For example, to reconstruct the centre of an elbow joint one would place a marker on either side of the elbow (specifically, on the lateral and medial epycondyles) and connect the two markers via an imaginary line: the joint centre is estimated to be somewhere on that line (intuitively, one could pick the middle of the line) [219]. For most OSSs, given the position of the markers the estimation of the joint centres is done automatically by the software that comes with the OSS, under the assumption that the markers be placed precisely on the anatomical landmarks that the software expects.

[1] Optoelectronic sensors, which in this context indicate the sensors inside of special cameras, are sensors that emit electrical impulses upon being hit by light.

[2] Stereophotogrammetry is the process of estimating the three-dimensional coordinates of points by using measurements made in two or more photographic images taken simultaneously from different positions.

The distinguishing feature of OSSs is that, at least in theory, they are extremely accurate: the most recent study on the accuracy of Vicon (Oxford, UK), one of the most popular OSSs, in reconstructing the 3D position of markers reported a Mean Average Error (MAE) of 0.15 mm and a standard deviation of 0.015 mm for markers placed on static objects, and a MAE of 0.2823-0.3543 mm and standard deviation of 0.1682-0.2439 mm for markers placed on objects moving at <1 m/s or >3 m/s, respectively [220]. Because of their accuracy, OSSs are often used for clinical and biomechanical research, both of which require extremely accurate measurements. In the case of swimming, however, specialised equipment has to be used: The cameras need to be waterproof or be placed in waterproof, transparent housings, and the markers need to be made of a material that will reflect light well enough even underwater[3]. Of the major companies that sell OSSs, Qualisys (Gothenburg, Sweden), with their Oqus line, is the only one that sells hardware and software specifically designed for underwater motion capture, and a number of studies have used this OSS for swimming biomechanics research [222–225]. For example, Olstad et al. [222] used six Oqus cameras and 20 Qualisys markers to reconstruct the kinematics of the legs during breaststroke kicking, while Lauer et al. [223] used ten Oqus cameras and twelve Qualisys markers to reconstruct the kinematics of the arm during sculling. Unfortunately, no study to date has validated the accuracy of Qualisys for underwater motion capture.

Though remarkably accurate, OSSs do have significant drawbacks. For instance, the accuracy of OSSs depends on the accurate placement of the markers on the anatomical landmarks the software expects. The algorithms that use the position of the markers to reconstruct the position of the joint centres assume that the markers be placed exactly on the anatomical landmarks specified. Let us clarify this statement by using again the example of the elbow joint centre. The software of the OSS has within it a biomechanical model which tells it that the elbow joint centre is (for example) the midpoint between the lateral and the medial epycondyles, and it assumes that the two markers placed on the elbow correspond exactly to the epycondyles; this is the case, for example, with Vicon's model [226]. If, however, the markers are placed even slightly askew of these landmarks, the software of the OSS will reconstruct

[3]Raghu et al. [221] recently found that markers covered in Safety of Life at Sea (SOLAS) Grade 3150–A tape, which is made of 3MTM ScotchliteTM Reflective material, reflect infrared light well enough for it to be picked up by cameras placed underwater or even above the water.

the joint's reference frame (i.e. the joint centre and the three axes of rotation that originate from it) in a position and with an orientation that will be proportionally askew from its true position and orientation. To locate the anatomical landmarks on which to place the markers, the operator has to palpate the body of the subject and 'feel' the landmarks, a process which is inherently prone to variability: It has been reported that the average intra-operator variability in locating anatomical landmarks is 9.87 mm, while the average inter-operator variability is 16.73 mm [227]. Such variability, in turn, is responsible for variability in the orientation of the axes of the reconstructed joint centres, which will deviate from their true position and orientation: Della Croce et al. [228] reported an intra-operator variability in the orientation of the axes of the joint centres of up to 4.7° (for the femur), and an inter-operator variability of up to 9.4° (for the tibia).

A second factor that affects the accuracy of OSSs is a phenomenon known as *soft-tissue artefacts* [229]. The algorithms that use the position of the markers to reconstruct the position of the joint centres assume that the human body behaves like a rigid body (i.e. that it is not subject to deformations). In other words, the reconstruction algorithms assume that the position of the markers relative to the underlying bony landmarks remain constant over time. When a human moves, this assumption fails almost invariably, as the skin to which the markers are attached moves slightly relative to the underlying bones (leading to the so called *soft-tissue artefacts*, also known as *skin artefacts*). This problem has been thoroughly researched [230–237], and a review paper by Leardini et al. [229] concluded that soft-tissue artefacts may be responsible for errors in the estimation of the joint centres of up to 20 mm[4]. The same review paper also reported that the magnitude of soft-tissue artefacts varies based on the motor task being performed, on the person, and on the type of joint under analysis. For example, Benoit et al. [33] reported an average error due to soft-tissue artefacts of 7.47 mm when estimating the position of the knee joint centre, whereas, for the same joint but under different testing conditions, Lafortune and Lake [34] reported errors of 20 mm. Although there exist analytical methods to reduce the magnitude of soft-tissue artefacts [243–247], none of them are able to

[4]To estimate the magnitude of errors caused by soft-tissue artefacts, one of the most common techniques involves inserting a metal pin into a bone and attaching a marker to the pin and one to the adjacent skin: the relative movement between the two markers will indicate the magnitude of the soft-tissue artefact [34, 238–242]

remove the soft-tissue artefacts altogether, and none of them are universally consistent [229].

The cumulative effect of these two factors (marker misplacement and soft-tissue artefacts) means that OSSs, which are extremely accurate in measuring the position of markers in space, may nonetheless incur in considerable reconstruction errors, because the markers may not be in the position that the biomechanical model of the software expects. It is hard to say whether these errors are consistently above the 5 mm threshold that has been suggested for research on biomechanics [26–28], but it is fair to say that the theoretical accuracy of OSSs can hardly ever be achieved for human motion capture tasks. Nevertheless, OSSs remain the most accurate systems available.

When OSSs are used for swimming motion capture they also present a third major drawback, which is not as relevant for the motion capture of other sports. Though the markers that are attached to the skin of the swimmer do not necessarily impede movement[5], they do increase drag enough to affect performance, which consequently ceases to be representative of race conditions. Let us estimate[6] by how much drag is increased when markers are attached to a swimmer. The drag D to which an object moving through water is subject is calculated using the formula:

$$D = \frac{1}{2}\rho C_d A v^2 \tag{A.1}$$

where ρ is the density of the medium in which the motion takes place (which for water is 997 kg m^{-3}), C_d is the drag coefficient (which is fixed and depends on the geometry of the object), A is the cross-sectional area (measured perpendicular to the direction of the flow), and v is the swimming speed of the object relative to the fluid. To simplify equation A.1 we can express D as a funcion of v: by doing so, all the terms on the right side of the equation become constants, which can be grouped into a single constant K:

$$\frac{D}{v^2} = \frac{1}{2}\rho C_d A = K \tag{A.2}$$

[5]In a study by Washino et al. [248], 14 elite swimmers performed a few trials of front crawl swimming while wearing the full Qualisys marker set and reported no discomfort.

[6]The following analysis will be based on the assumption that the only type of drag present is passive drag, which is the drag on the swimmers while holding a fixed position, e.g. gliding following a dive entry or push-off. Outside of this simplification, each marker will have a different velocity through the water, and therefore a separate calculation should be performed for each marker.

Using equation A.2 and knowing that the markers sold by Qualisys for underwater capture have a diameter of 19 mm and that the drag coefficient of a sphere[7] is 0.47 [249], we can calculate that, for a single marker i, $K_i = 0.06642$ kg/m; if a full Qualisys marker set is used (25 markers), we obtain that the cumulative effect of the markers is to increase drag by $K_{markers}$ = 1.66 kg/m. To determine if an increase of this magnitude is significant, we need to know the value of K for an average swimmer. The value of K for a human being who is swimming at speed v is much more difficult to calculate than it is for a sphere, since for a swimmer the value of A is not known a priori and is not constant: it changes based on the sex, body composition, and anthropometric measures of the swimmer; more importantly, the value of A changes significantly based on the type of stroke being performed and on the speed at which it is performed. Different authors have reported different values for $A_{swimmer}$, ranging from 0.08 m^2 [250] to 0.40 m^2 [251]. The most complete work in this field, which is also the work that used the most modern measuring equipment, is that of Gatta et al. [252]. Using a markerless OSS to estimate the value of $A_{swimmer}$ for each swimming stroke (front crawl, backstroke, butterfly, breststroke; each stroke was swum at a speed of roughly 1.3 m/s), Gatta et al. came to the formulation of the following values of $K_{swimmer}$:

$$K_{swimmer}(front\,crawl) = 30.0 \quad kg/m$$

$$K_{swimmer}(backstroke) = 26.9 \quad kg/m$$

$$K_{swimmer}(butterfly) = 28.5 \quad kg/m$$

$$K_{swimmer}(breaststroke) = 37.5 \quad kg/m$$

We can now use the following equation to calculate the effect that adding 25 spherical markers would have on the drag of the swimmer:

$$K_{total} = K_{swimmer} + K_{markers} \tag{A.3}$$

Using the value $K_{markers} = 1.66$ calculated above and taking as reference the values of $K_{swimmer}$

[7] C_d is actually not an absolute constant: it is inversely proportional to the Reynolds number, which determines if the flow is laminar (slow) or turbulent (fast). For turbulent flow (a condition which is taken for granted when analysing the flow in the immediate vicinity of a swimming human being), the C_d of a sphere is 0.47.

calculated by Gatta et al., we obtain:

Front crawl: $K_{total} = 31.66 \quad kg/m \longrightarrow 5.53\%$ increase

Backstroke: $K_{total} = 28.56 \quad kg/m \longrightarrow 6.17\%$ increase

Butterfly: $K_{total} = 30.16 \quad kg/m \longrightarrow 5.82\%$ increase

Breaststroke: $K_{total} = 39.16 \quad kg/m \longrightarrow 4.43\%$ increase

These results are lower than those reported by other authors. Since $K_{markers}$ remains constant (assuming a full marker set o 25 markers is used), the source of variability must reside in the way different authors calculate $K_{swimmer}$. In particular, Washino et al. [248], who analysed the drag caused by a full Qualisys marker set during front crawl swimming, used $K_{swimmer} = 21.55$ (as suggested by Zamparo et al. [251]) and obtained $K_{total} = 23.21$, which would indicate that the presence of the markers would increase drag by 7.7%. Differences in the estimation of $K_{swimmer}$ could be due to several factors: instrument errors, variability in the actual cross-sectional area of the swimmers recruited for different studies, different level of mastery of technique, variability in the speed of the swimmers (although the swimmers analysed by Washino et al. were the fastest, at 1.75 m/s, so in theory they should have had a lower $K_{swimmer}$ than those analysed by Gatta et al., who swam at 1.3 m/s). Whatever the reason for these different results, the question remains: is an increase in drag by 4.4-7.7% enough to impact the performance of the swimmers, thus violating Criterion 2, non-invasiveness? According to Washino et al., the answer is yes. In their study they correlated the 7.7% increase in drag to a clear decrease in performance. Specifically, they reported that swimmers who wore the full marker set swam 3.3% slower and had a 2.5% shorter stroke length. Finally, though this theory has not been tested, it is reasonable to assume that the drag acting on the spherical markers would exacerbate the issue of soft-tissue artefacts.

Appendix B

Sensor-based Systems

The generic term 'sensor-based motion capture system' refers to the use of *inertial measurement units* (IMUs) for motion capture. IMUs are self-contained electronic devices that give information relative to their orientation in space. They are typically made up of three types of sensor:

1. **Gyroscopes**. Gyroscopes are sensors that measure angular velocity. Because gyroscopes are subject to a certain bias b and to a certain Gaussian measurement error η, the actual angular velocity they measure is:

$$\tilde{\boldsymbol{\omega}} = \boldsymbol{\omega} + b + \eta_{\text{gyro.}} \tag{B.1}$$

   where $\boldsymbol{\omega}$ is the true angular velocity of the device. From $\tilde{\boldsymbol{\omega}}$, which is the measured angular velocity, it is then possible to calculate the orientation of the device relative to what are known as the 'aircraft principal axes' [253]: yaw, pitch, and roll (see Figure B.1). If the orientation $\boldsymbol{\theta}$ of the sensor at time *t=0* is known[1], the orientation at time t can be reconstructed using the formula:

$$\tilde{\boldsymbol{\theta}}^{(\text{t})} = \tilde{\boldsymbol{\theta}}^{(\text{t-1})} + \tilde{\boldsymbol{\omega}}\Delta t + \epsilon \tag{B.2}$$

[1]This can be accomplished by initialising the gyroscope before the data-capturing session.

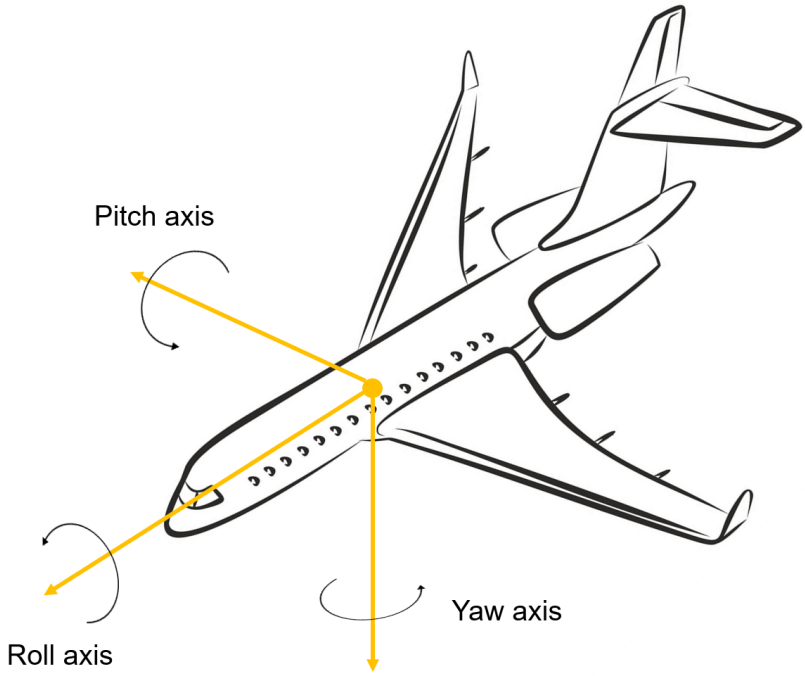


Figure B.1: Aircraft principal axes, used to describe the orientation of IMUs.

where Δt is the time step to go from *t-1* to t and ϵ is an approximation error[2]. Equation B.2 needs to be calculated for each time step at which the gyroscope measures $\tilde{\boldsymbol{\omega}}$. This means that the errors η and ϵ accumulate at each time step, leading to a phenomenon known as *drift* [253]: after a few measurements, the orientation measured by the gyroscope quickly diverges (drifts) from the true orientation of the device[3].

2. **Accelerometers**. Accelerometers are sensors that measure linear acceleration. Because accelerometers are subject to a certain Gaussian measurement error η, the actual linear acceleration they measure is:

$$\tilde{\mathbf{a}} = \mathbf{a}_{\text{external}} + \eta_{\text{acc.}} \tag{B.3}$$

where $\mathbf{a}_{\text{external}}$ is the acceleration due to the sum of the external forces (including gravity) acting on the sensor. If the position $\mathbf{p}$ of the sensor at time *t=0* is known, the position at time t can be reconstructed by integrating $\tilde{\mathbf{a}}$ twice over time (a process known as *dead*

[2]Formula B.2 is obtained via Taylor expansion, which means that the result is subject to an error $\epsilon \sim O(\Delta t)$.
[3]How quickly the two values diverge depends on the magnitude of η (which depends on the quality of the sensor) and on the magnitude of Δt: with higher sampling rates (i.e. low Δt) the errors are accumulated faster, but with lower sampling rates (i.e. high Δt) the magnitude of the cumulative error ϵ, which scales with the square of Δt, is much greater.

reckoning [254]), yielding:

$$\tilde{\mathbf{p}}^{(\mathrm{t})} = \tilde{\mathbf{p}}^{(\mathrm{t\text{-}1})} + \iint \tilde{\mathbf{a}}\, dt \tag{B.4}$$

However, because a double numerical integration is performed, the error associated with dead reckoning grows quadratically with time ($\epsilon \sim O(\Delta t^2)$), meaning that $\tilde{\mathbf{a}}$ drifts from $\mathbf{a}$ much faster than $\tilde{\boldsymbol{\omega}}$ drifts from $\boldsymbol{\omega}$ in gyroscopes (since it was said earlier that gyroscopes drift with $\epsilon \sim O(\Delta t)$). Without specialised algorithms to correct this drift, accelerometers can hardly be used to measure the position in space of the IMU.

The true reason accelerometers are used in IMUs is that they can function as feedback tools to correct the drift of $\tilde{\boldsymbol{\theta}}$ as estimated by the gyroscope. The signal produced by the accelerometer ($\tilde{\mathbf{a}}$) can be used to calculate the orientation of the IMU relative to the pitch and the roll axes (but not the yaw axis) using the following equations (in which roll is denoted by x and pitch by y):

$$\tilde{\theta}_{\mathrm{x}} = -atan2(\tilde{a}_{\mathrm{y}},\ sign(\tilde{a}_{\mathrm{z}}) \cdot \sqrt{\tilde{a}_{\mathrm{x}}{}^2 + \tilde{a}_{\mathrm{z}}{}^2}) \tag{B.5}$$

$$\tilde{\theta}_{\mathrm{y}} = -atan2(-\tilde{a}_{\mathrm{x}},\ \tilde{a}_{\mathrm{z}}) \tag{B.6}$$

These estimates of $\tilde{\theta}_{\mathrm{x}}$ and $\tilde{\theta}_{\mathrm{x}}$ do not suffer from drift like they do when they are calculated from $\tilde{\boldsymbol{\omega}}$, but they do suffer from a measurement error $\eta_{\mathrm{acc.}}$, which is typically larger than $\eta_{\mathrm{gyro.}}$. Although neither measurement is accurate enough on its own, their combination (a technique called *sensor fusion*, which is usually done using a Kalman filter) drastically increases the accuracy of the IMU's estimation of $\boldsymbol{\theta}$. Nevertheless, it is not currently possible to eliminate sensor drift completely, which means that the longer the data capturing session lasts, the more severe the errors will be, unless the sensors are recalibrated [32, 255].

3. **Magnetometers**. Magnetometers are sensors that measure the intensity of magnetic fields. The function of magnemeters in IMUs is to measure Earth's magnetic field and use its magnitude and direction as feedback tools to correct the yaw calculated by the gyroscope, much like the function of accelerometers in IMUs is to correct pitch and roll.

Earth's magnetic field, however, is weak, and ferromagnetic objects near the IMU may produce a magnetic field of comparable magnitude to Earth's, making magnetometers unreliable. For this reason, they are not always included in IMUs.

To measure 3D kinematics using IMUs, several of them need to be strapped or glued to the skin or clothes of the person, in locations and orientations that correspond to specific body segments (i.e. an IMU on the thigh, one on the shank, etc) [36, 256]. Each IMU would then give the orientation (but not the position) of that body segment in time. To get to the joint angles, the axes of the IMU need to be rotated to match the axes of the body segment to which the IMU is attached, and then rotated to match the axes of the joint centre; this process is usually performed by specialised software that comes with the IMU motion capture system. Because the software bases its formulas on the assumption that the IMUs be placed exactly in very specific locations, IMU-based motion capture is susceptible to misplacement errors. Whereas the misplacement errors of OSSs are systematic (the marker may be in the wrong position, but it will not move from that position), the misplacement errors of IMU-based systems can vary over time, as the straps to which the IMUs are secured may shift along the body segment and not come back to their original position (for example as a result of highly dynamic movements, or, in the case of swimming, of the drag acting on the IMUs). This means that IMU-based motion capture systems are susceptible to a sort of 'misplacement error drift', which can severely impact the accuracy of the measurements. To avoid this, the straps need to be fastened quite tight around the body segments, but this may cause discomfort for the person, or it could obstruct the free movement of certain joints. Therefore, IMU-based motion capture violates Criterion 2 (non-invasiveness). This conclusion is substantiated by the fact that fastening several IMUs to the swimmer, though it may not necessarily obstruct movement, will surely increase drag. The exact amount by which drag is increased depends on the number of IMUs used and on their size. The Xsens MTw Awinda (Xsens Technologies BV, Enschede, Netherlands) is the most accurate IMU-based full-body 3D motion capture system on the market [29, 257], and as such it will be used as a reference for the following calculations. It consists of 20 IMUs (of size 47 x 30 x 13 mm) which are not waterproof and would need to be placed inside plastic boxes or bags to be used for underwater capture; for the purposes of this analysis, let us assume that the

increment in IMU size caused by the presence of a waterproof plastic box would be negligible. Equation A.2 can be used to find by how much the Xsens system would increase drag, but the drag coefficient C_d and the area A perpendicular to the flow are not as easy to calculate for an IMU as they were for a sphere, because the orientation of the IMU relative to the direction of the flow is not constant as it is for a sphere. To simplify the calculations, we will assume that the IMU is always oriented relative to the flow so that its face with the smallest area is perpendicular to the flow. For such a configuration, and considering the presence of 20 Xsens IMUs[4] of size 47 x 30 x 13 mm, we obtain that $K_{\text{IMUs}} = 4.08$ kg/m. Using $K_{\text{swimmer}} = 23.21$ as suggested by Zamparo et al. [251], we obtain that using Xsens for underwater motion capture would increase the drag of the swimmers by 23.21%, thus violating Criterion 2.

Apart from concerns about the invasiveness of IMUs, a more generic question could be asked: can they even be used for underwater 3D motion capture? In theory, the answer is yes, and indeed IMUs have been used quite extensively for swimming research. However, the vast majority of studies that used IMUs for swimming biomechanics research used IMUs to measure parameters related to performance, rather than to technique. Thus, a plethora of studies have been published on the use of IMUs to measure stroke rate and stroke count [258–271], or to identify the type of stroke being performed [272–280], or to measure the speed and acceleration profiles of the swimmers [255, 281–297], but only two studies have been published to date that used IMUs to measure 3D joint angles of underwater swimmers: Seifert et al. [298] used 4 IMUs to measure knee and elbow angles of swimmers performing breaststroke, while Phillips et al. [299] used 4 IMUs to measure knee and hip angles of swimmers performing the fly kick. Phillips et al. compared their IMU-based system with manually-digitised knee and hip angles, and found that the IMU-based system had small errors (0.1° mean) in the estimation of knee angles, but larger errors (4.0° mean) in the estimation of hip angles. The 4.0° error reported by Phillips et al. was obtained by using manually-digitised angles (which are susceptible to human error) as the validation data; this means that the error from the true orientation of the joint centre may have been much greater than 4.0°. Indeed, the accuracy of IMUs in sports motion capture in general, which has been reported to range from 1.38° to 6.69° [300],

[4]This is the total number of IMUs that come with Xsens, and it's the number of IMUs required for a full-body analysis. For a more circumscribed analysis, fewer IMUs may be used.

has never been validated using measurements of the true joint centre position (as measured using bone pins, for example). Instead, IMU-based motion capture has always been validated using either OSSs [35,36] (which are prone to skin-artefact and misplacement errors) or manual digitisation [256] (which is prone to human error). Therefore, it is probable that the accuracy of IMU-based motion capture systems has been overestimated.

Appendix C

Manual Digitisation

If a point in space is seen by at least two cameras with non co-linear axes, its 3D coordinates can be reconstructed using stereophotogrammetry, which works as follows. Let us assume that we have two cameras, each recording an image in which appears a point of interest. The point, whose coordinates in the global reference frame are expressed by the 3D variable X, has two sets of 2D coordinates (one in each camera reference frame): x_1 for camera 1, x_2 for camera 2. To express x_1 and x_2 in terms of the global reference frame—and thus obtain the 3D coordinates of the point—the axes of each camera reference frame need to be rotated by a certain amount. Therefore, we have that:

$$x_1 = P_1 X; \quad x_2 = P_2 X \tag{C.1}$$

where P_1 and P_2 express the rotation needed to get from the camera reference frames to the global one; P_1 and P_2 are known if the cameras have been calibrated, using for example Direct Linear Transformation [301]. The two equations above can be combined into:

$$AX = 0 \tag{C.2}$$

which is an equation linear in X. If the camera parameters P_1 and P_2 are known, equation C.2 can be solved exactly for any point X that is seen by both cameras.

This method allows to reconstruct the 3D coordinates of human joints using nothing but

images: provided that at least two cameras see the joint, an operator can digitise the joint (i.e. mark it on the image, thus determining its 2D coordinates in camera space) and use equation C.2 to reconstruct its 3D coordinates. This process can be repeated for all joints of interest, over all recorded frames; and it can be done by hand (hence the term *manual digitisation*[1]), or automatically, using software like SIMI Motion (SIMI, Unterschleissheim, Germany) and APAS (Ariel Performance Analysis System, Ariel Dynamics, Inc., USA).

The working mechanism (stereophotogrammetry) behind this method is the same that is behind OSSs, but three major differences exist between the two methods:

1. Manual digitisation does not require markers to be placed on the person whose movements are to be recorded. This makes manual digitisation a completely non-invasive motion capture method. Nevertheless, small circular tape markers are often attached to bony landmarks—usually on the same positions that an OSS's spherical markers would be—to make it easier to identify the points (i.e. the joints) that need to be digitised [38]; if using software for automatic digitisation, the presence of markers is mandatory, as they provide the software with specific, easily recognisable points to follow from one frame to the other. It has also been suggested that the presence of markers increases the accuracy of manual digitisation as performed by a human operator [310]. The study in which these conclusions were drawn was performed in a highly controlled laboratory, where the lighting was excellent and the contrast between the markers and the skin and between the person and the background was sharp. Under such ideal conditions, having skin markers as a reference certainly does make manual digitisation easier and faster. As pointed out by some authors, though, skin markers should only ever be used as guides [31, 311]: the decision of where to digitise the joint in the 2D image should ultimately come from the (supposed[2]) sound knowledge, on the part of the operator, of the muscolo-skeletal system

[1] In swimming, manual digitisation is the method for full-body 3D motion capture that has been used the most [302–309].

[2] Bahamonde and Stevens [312] estimated that the joints digitised by 'experienced' operators are up to 10 mm more accurate than those digitised by complete novices. However, the 'experienced operators' they recruited for their study were undergraduates, who are unlikely to already have had enough training in anatomy and biomechanics to be certain about their estimation of where the joint centres lie. Therefore, the true influence of experience on the accuracy of manual digitisation is currently unknown, but it is likely significant.

underlying the joint being digitised. This is because tape markers, like any other kind of marker, are susceptible to misplacement and soft-tissue artefacts, and may therefore not be indicative of the real position in space of the joint centre.

2. Whereas OSSs require cameras with optoelectronic sensors, manual digitisation can be done with any type of camera. Normal cameras, however, are bound to operate in the visible spectrum of light, meaning that any reflection on the skin of the person being filmed and any change in the lighting conditions of the scene will likely make it harder, sometimes impossible, to precisely identify the joint centres. Conversely, OSSs operate in the infrared spectrum, which grants them more resistance against changes in lighting conditions.

3. Every OSS commercially available comes with a software that performs stereophotogrammetry automatically; the only input requested of the user is the occasional manual digitisation of markers that may have been occluded during capture. This makes OSSs extremely quick at calculating the 3D coordinates of the desired joint centres, which are available to the operator minutes after the capture session has ended. Manual digitisation, on the other hand, is incredibly time consuming: Magalhaes et al. [38] estimated that it took Psycharis et al. [313] 27 hours to manually digitise 19 joints of a single swimmer performing 200 m freestyle trials[3]. This estimation of the time required for manual digitisation is conservative, since it does not take into account two factors: a) To provide a measure of the intra-operator reliability of the data, at least one trial should be analysed multiple times by the same operator; b) To provide a measure of repeatability, at least one trial should be analysed also by a second operator [311]. Using automatic-digitisation software speeds up the process considerably (according to Ceccon et al. [314], by at least 50%), but such software require a highly controlled environment, which a swimming pool certainly is not. When the software is not confident in its prediction of where the joint is in a particular frame, the user has to intervene and manually digitise the joint. Since the turbulence, bubbles, and changing lighting conditions of underwater videos often cause

[3]Each trial consisted of 1620 frames, coming from 4 cameras; 10 swimmers were analysed during the study, adding up to an estimated 270 hours of manual digitisation.

the automatic-digitisation software to lose track of the joints, such software does not constitute a significant-enough improvement over fully-manual digitisation to make them a viable option.

We can conclude that manual digitisation respects Criterion 2 (non-invasiveness) but not Criterion 1 (speed). Criterion 3 (accuracy) is harder to evaluate, because manual digitisation has only ever been validated using OSSs as the ground truth [24,31,310,315]; in some occasions, the accuracy of manual digitisation has been evaluated using automatic digitisation software as the ground truth [38,312,316] without providing any measure of the accuracy of such software. Further complicating matters is the fact that the studies that compared manual digitisation to OSSs did so under highly controlled laboratory settings or by analysing simple, slow movements that only occurred in one plane (for example, knee flexion [31]), all factors which make it easier to perform manual digitisation. For instance, Elliott et al. [315] found that manual digitisation is only 1.18° less accurate than an OSS when analysing a motion constrained to take place in a single plane (elbow flexion), but that the error increases to 10° when analysing multi-planar motions (elbow flexion + shoulder internal rotation). Therefore, when Hanley et al. [31] report that manual digitisation is 1-8° less accurate than an OSS in estimating the knee flexion of a person walking, they do not provide enough information regarding the accuracy of manual digitisation for different tasks and under different experimental conditions. Similar critiques can be raised regarding the validity of Wilson et al.'s study, often cited as a reference for the validity of manual digitisation [312, 317–319]. In their study, Wilson et al. recorded the oscillations of a T-shaped pendulum made of metal on which spherical markers were affixed; the markers were then digitised manually and using automatic software. Since the dimensions of the pendulum were known down to 1 μm, their experimental design allowed Wilson et al. to exactly quantify the accuracy of manual and automatic digitisation (both of which were 1-2° off the real measurements of the positions of the markers), but the simplifications it introduced (planar motion, ideal contrast between markers and background, no soft-tissue artefacts, no misplacement of markers) make it so their results are hardly generalisable to in-the-wild[4] motion

[4]In computer vision, the term *in-the-wild* refers to images recorded under non-ideal conditions. All images recorded for sports biomechanics purposes would be classified as in-the-wild images.

capture.

Another element that affects the accuracy of manual digitisation is the order in which the joints are digitised. There exist two possibilities: either the operator digitises all the joints present in a frame and then moves on to the next frame (this method takes the name of *frame by frame*, or *FXF* [312]); or he/she digitises a single joint across all frames, then repeats the process for each joint (this method takes the name of *points over frame*, or *POF* [312]). The POF method has a distinct advantage over the FXF method: it is more accurate. Using an automatic digitisation software as the ground truth, Bahamonde and Stevens [312] calculated that the POF method was between 1 and 6.7 mm more accurate, on average, than the FXF method. Though Bahamonde and Stevens did not offer an explanation for their results, the reason the POF method is more accurate than the FXF method may be that it is easier for a person to follow a single joint from one frame to the next. Once the joint centre has been identified, moving to the next frame will cause the joint to 'move' on the image by a certain amount. If the original position of the joint is still fresh in the memory of the operator, the movement of the joint can be followed precisely, and the operator can learn to predict where the joint will be based on the previous positions that it has occupied and on the speed at which it has been moving. Conversely, in the FXF method, the operator cannot reasonably hold in his or her short memory the position of all joints from one frame to the next; in other words, when the operator moves on to a new frame, he or she would have to identify the position of the joints anew, thus increasing the probability of digitising the joint with a few pixels of variability from one frame to the next. Though this error may seem trivial, Hanley et al. [31] have estimated that errors of 1 pixel in the digitisation of a joint can lead to errors of 0.17° in the orientation of its reconstructed axes. Whatever the reason may be for the superior accuracy of the POF method, authors who use manual digitisation tend not to mention which method was used in their study, perhaps out of a lack of knowledge of the existence of the POF, or of its superior accuracy. Not knowing which manual digitisation method was used further complicates the evaluation of the studies that compared manual digitisation to OSSs.

Appendix D

Depth-based Systems

Depth cameras are cameras that are able to measure the depth—which is to say, the 3D shape—of the objects in their field of view. An example of a depth camera is Microsoft's Kinect: Originally designed as a videogaming accessory, it has since become an important tool in many aspects of scientific research [320–322], not the least biomechanics. For example, Schmitz et al. [323] used a Kinect to study the biomechanics of the squat, while many studies have been published regarding the application of the Kinect to gait analysis [324–329]. When used for motion capture, depth cameras have important advantages over marker-based systems [32]: they are highly portable; they are easy to use; they take only a couple of minutes to set up; they provide data in real time; they are inexpensive; and they do not require the fixation of external markers or sensors, thus eliminating the issue of soft-tissue artefacts. To understand if depth cameras can be used for swimming motion capture, we will discuss here the two depth cameras most commonly used in scientific research[1], both of which are developed by Microsoft: the Kinect 1 and the Kinect 2.

- **Kinect 1.** The first version of Microsoft's Kinect was released in 2010. It consists of an RGB camera, a near-infrared (NIR) projector, and a NIR camera (see Figure D.1).

[1] Other examples of depth cameras that have been used for scientific research are the PMD CamBoard pico flexx [330, 331] and Intel's RealSense [332, 333]. Microsoft's Kinect, however, is far more popular [32, 334]—not necessarily because of its higher accuracy (in fact, the pico flexx has shown accuracy comparable to that of the Kinect 2 [331]), but perhaps just because it is a more well-known, easily accessible brand.

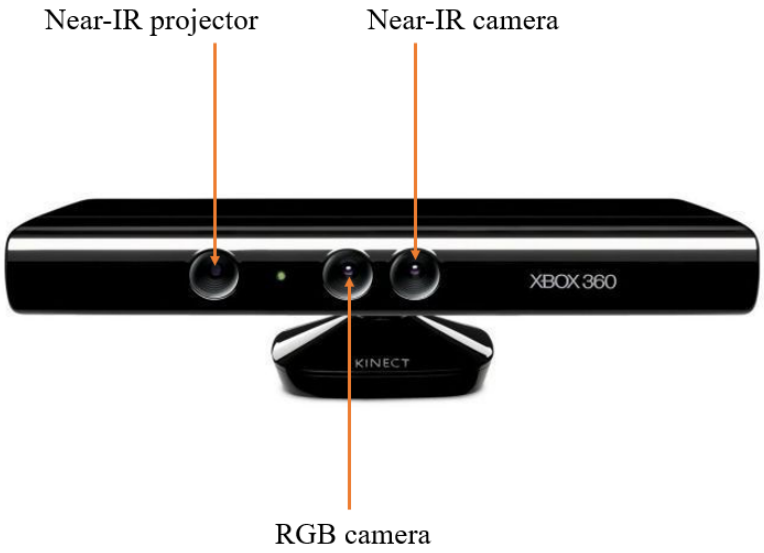


Figure D.1: Microsoft's Kinect 1.

To measure the depth of objects in a scene, the projector casts onto the scene a known pattern of NIR light[2]. The pattern will be distorted differently by surfaces that are at different depths, and this distortion is observed by the NIR camera. Knowing the focal length f of the camera and knowing what the original pattern was supposed to look like, the Kinect 1 uses the following equation [41] to reconstruct the depth of each pixel:

$$d = bf/m \tag{D.1}$$

where $b = 7.5$ cm is the distance between the NIR camera and the projector, and m, which is called the *disparity value*[3] [41], measures the distortion of the pattern that was expected to be on that pixel. Applied to the whole image, equation D.1 yields a *depth map* of the scene (see Figure D.2). There are several sources of error that affect the accuracy of the Kinect 1's depth maps:

– **Multi-device interference.** If multiple Kinect 1s are pointed at the same object, the patterns projected onto it will overlap and will not be distinguishable enough to calculate the disparity value m for each device [41].

[2]The Kinect 1 is also sometimes called *Kinect*^{SL}, where *SL* stands for *structured light*.

[3]As shown in Figure D.1, the NIR camera and projector of the Kinect 1 are displaced only horizontally. Therefore, the disparity value of the Kinect 1 is one-dimensional, thus simplifying its computation.

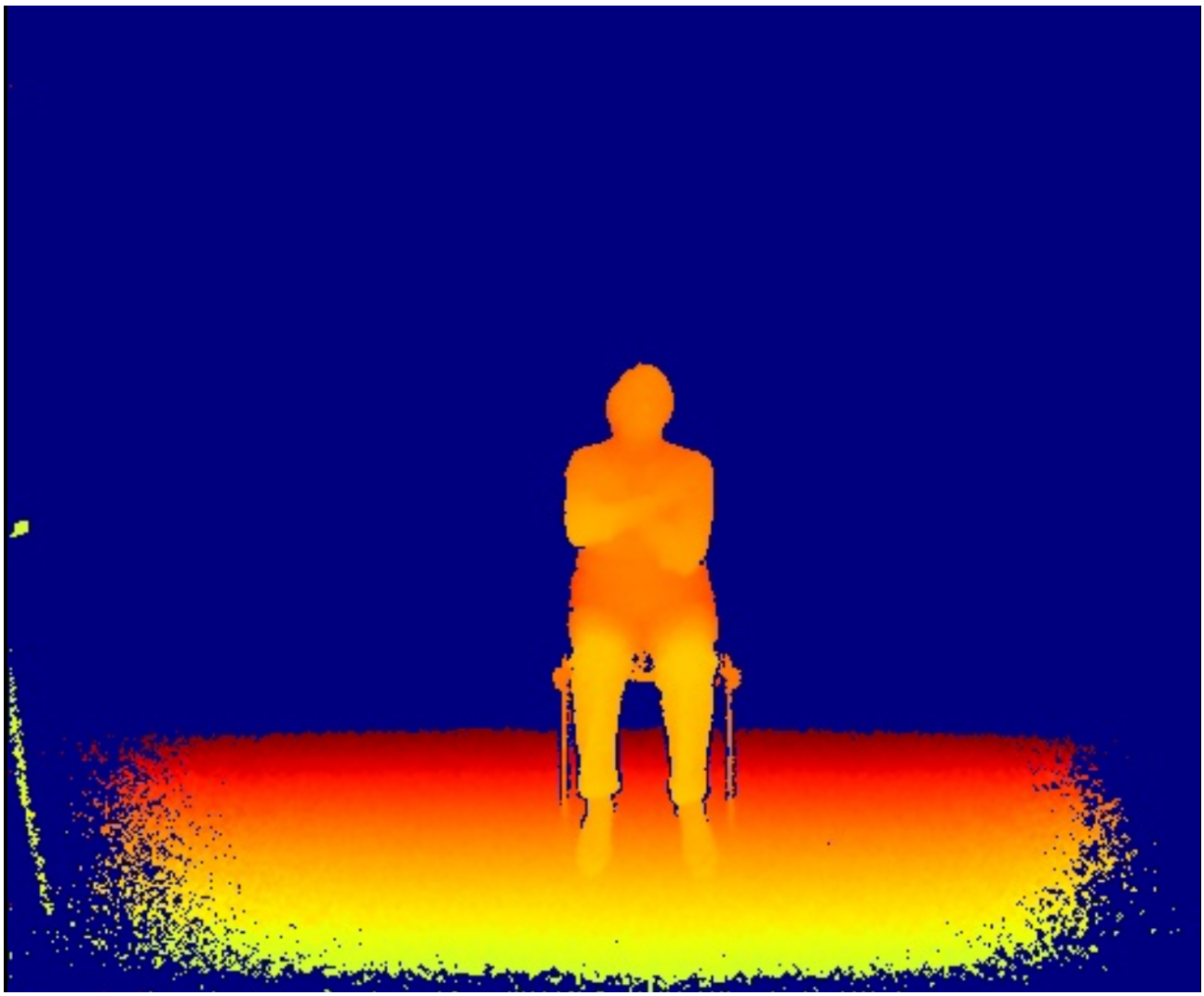


Figure D.2: Example of a depth map. Points closer to the camera (i.e. with lower depth, as calculated using equation D.1) appear as lighter pixels; points farther from the camera appear as darker pixels.

- **Ambient light.** The presence of ambient light is another factor that can distort the patterns projected by the Kinect 1 [41]. This makes the Kinect 1 unsuitable for outdoor (and, likely, underwater) capture [41, 335].
- **Systematic distance errors.** The accuracy with which the Kinect 1 can measure the depth of an object decreases with the square of the distance between the camera and the object; in other words, objects that are further from the camera are reconstructed with less accuracy [336]. This source of error is negligible for objects that are 1 metre or closer to the camera, but it can be as high as 4 cm per pixel for objects that are 2 metres away from the camera [13].

To get from a depth map to an articulated body model, Kinect cameras use Software

Development Kits (SDKs) provided by Microsoft. In the SDK for the Kinect 1, a random decision forests algorithm [337] is used that compares the depth maps to a large training set of synthetically-generated depth images of people in many different poses and of different shapes [338], and estimates where each joint (20 in total) is.

- **Kinect 2.**The Kinect 2, released in 2014, also has three main components: an RGB camera, a NIR projector, and a NIR camera (see Figure D.3). The depth map, however,

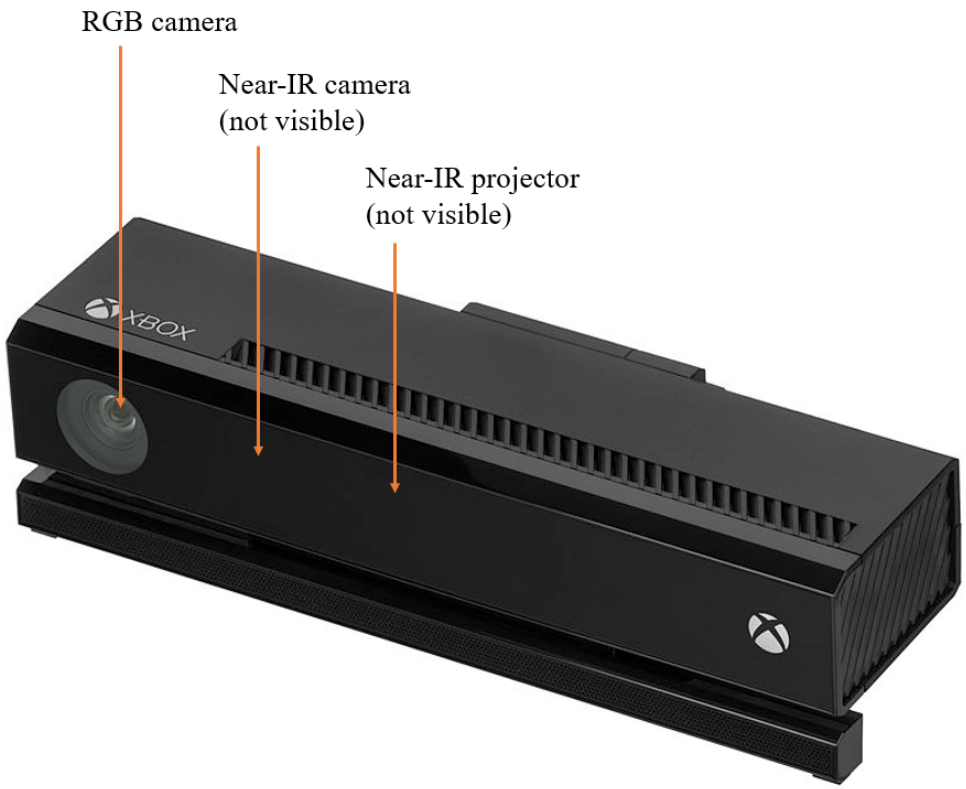


Figure D.3: Microsoft's Kinect 2.

is computed using a different mechanism. Instead of a structured light pattern, the projector of the Kinect 2 emits a beam of modulated[4] NIR light, which is reflected by whichever surface it hits and is then detected by the NIR camera. Having traveled a certain distance, the beam of light, when detected by the NIR camera, will be shifted in phase with respect to when it was emitted [41]. By measuring the amount of phase shift that the incoming beam displays, it is possible to calculate the distance it has traveled, which can then be halved to obtain the distance from the projector to the point

[4]This means that the frequency of the electromagnetic wave is known exactly and is unique [339].

it illuminated (i.e. the 'depth' of the point). There are several sources of error that affect the accuracy of the Kinect 2's depth maps:

– **Overheating.** The Kinect 2, which emits a NIR ray instead of a sparse pattern, requires more power than the Kinect 1 to produce a full depth map of its field of view, and therefore it requires active cooling (performed via fan; in the Kinect 1, the cooling is passive) to prevent the projector from overheating [13]. Nevertheless, the heat generated by the projector is enough to introduce errors in the measurements. These errors have been modelled by Wasenmüller and Stricker [13] (see Figure D.4) and can, in theory, be corrected: if, before capture, the Kinect 2 is left to warm up for 25 minutes, the offset caused by the overheated projector will plateau to a certain value, which can be measured and removed.

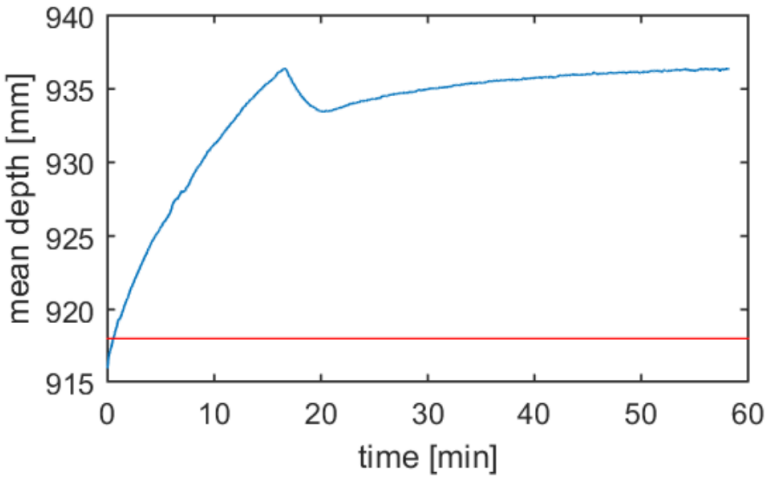


Figure D.4: Red: true depth of the point on which the Kinect 2 is focused. Blue: depth measured over time by the Kinect 2. The dip in the blue line around 16 minutes corresponds to the moment the cooling fan of the Kinect 2 activates itself. (Image adapted from: [13])

– **Flying pixels.** Flying pixels are erroneous depth estimates that occur close to discontinuities in depth (i.e. on the edges of objects) [13, 39, 40]. This phenomenon only occurs for the Kinect 2 and is due to the way it measures depth. If a point belongs to the edge of an object, the light that the camera of the Kinect 2 detects upon having irradiated that point will be a mixture of the light that was reflected by the point and the light reflected by the (much more distant) background [40]. This effect creates the sparse erroneous depth estimates shown in Figure D.5.

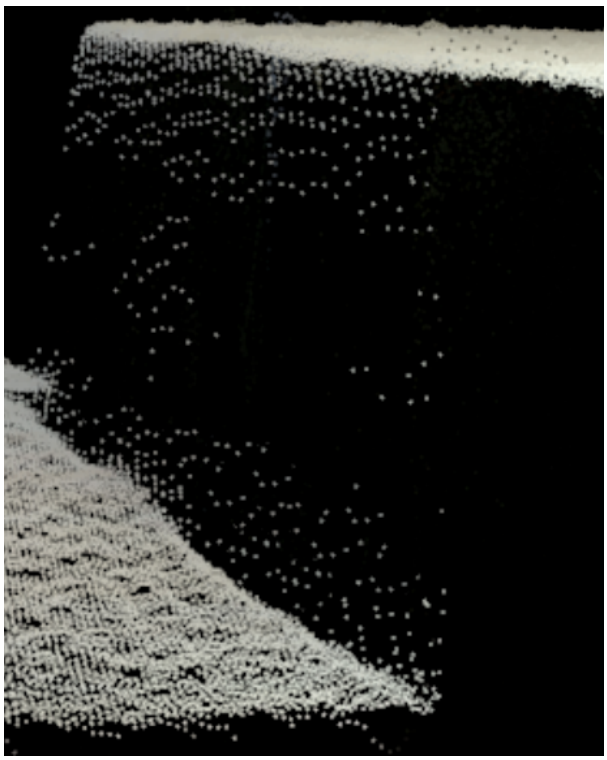


Figure D.5: Flying pixels caused by the Kinect 2 erroneously estimating that the pixels near the edge of the two adjacent objects are at a depth that is halfway between the depth of the objects and that of the background. (Image adapted from: [13])

– **Systematic distance errors.** Like the Kinect 1, the Kinect 2 is only accurate at estimating the depth of objects that are at most 2 metres away from it [13]. Furthermore, the corners of the depth map produced by the Kinect 2 are highly inaccurate (5 cm mean errors). The area of these corners of low accuracy increases the farther the Kinect 2 is from the object, as shown in Figure D.6; for objects that are 2 metres away from the Kinect 2, only a circular region with a radius of 300 pixels will have systematic errors lower than 2 cm per pixel [13].

The algorithm the Kinect 2 uses to turn the depth maps into articulated body models has not entirely been made public, but it is speculated to be similar to the one used by the Kinect 1 [336].

Having described in detail both devices, it is immediately clear how their sources of error (mainly the systematic distance errors, and, for the Kinect 1, the vulnerability to ambient light) make them unsuitable for the motion capture of swimming, since errors of 2-5 cm per point are common for both devices [32, 336, 340]. Furthermore, because water has higher absorbance

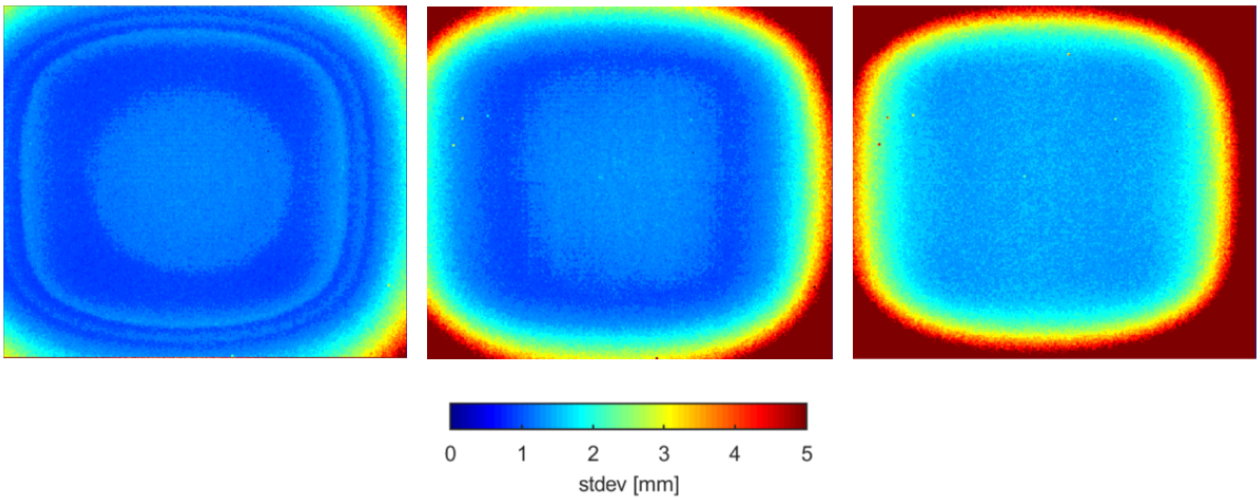


Figure D.6: The images above display the standard deviation in a Kinect 2's measurement of the depth of a wall placed at 0.7 metres (left), 1.4 metres (middle), and 2.1 metres (right) from the Kinect 2. (Image adapted from: [13])

than air, the light emitted by a NIR source would be attenuated more than it would be in air [341]. This means that a Kinect 2 device placed underwater would need to be even closer than 2 metres to the object it has to record, making it even more impractical for swimming motion capture.

Bibliography

[1] D. Drover, C.-H. Chen, A. Agrawal, A. Tyagi, and C. Phuoc Huynh, "Can 3d pose be learned from 2d projections alone?," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 0–0, 2018.

[2] C.-H. Chen, A. Tyagi, A. Agrawal, D. Drover, S. Stojanov, and J. M. Rehg, "Unsupervised 3d pose estimation with geometric self-supervision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5714–5724, 2019.

[3] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *Computer Vision (ICCV), 2017 IEEE International Conference on*, pp. 843–852, IEEE, 2017.

[4] D. Lin, Y. Ji, D. Lischinski, D. Cohen-Or, and H. Huang, "Multi-scale context intertwining for semantic segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 603–619, 2018.

[5] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *European conference on computer vision*, pp. 483–499, Springer, 2016.

[6] W. Tang and Y. Wu, "Does learning specific features for related parts help human pose estimation?," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1107–1116, 2019.

[7] K. Sun, B. Xiao, D. Liu, and J. Wang, "Deep high-resolution representation learning for human pose estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5693–5703, 2019.

[8] Z. Su, M. Ye, G. Zhang, L. Dai, and J. Sheng, "Cascade feature aggregation for human pose estimation," *arXiv preprint arXiv:1902.07837*, 2019.

[9] R. L. Figueroa, Q. Zeng-Treitler, S. Kandula, and L. H. Ngo, "Predicting sample size required for classification performance," *BMC medical informatics and decision making*, vol. 12, no. 1, p. 8, 2012.

[10] J. Cho, K. Lee, E. Shin, G. Choy, and S. Do, "How much data is needed to train a medical image deep learning system to achieve necessary high accuracy?," *arXiv preprint arXiv:1511.06348*, 2015.

[11] Z. Zhang, X. Zhang, C. Peng, D. Cheng, and J. Sun, "Exfuse: Enhancing feature fusion for semantic segmentation," *arXiv preprint arXiv:1804.03821*, 2018.

[12] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

[13] O. Wasenmüller and D. Stricker, "Comparison of kinect v1 and v2 depth images in terms of accuracy and precision," in *Asian Conference on Computer Vision*, pp. 34–45, Springer, 2016.

[14] A. Lees, "Technique analysis in sports: a critical review," *Journal of sports sciences*, vol. 20, no. 10, pp. 813–828, 2002.

[15] A. Von Loebbecke, R. Mittal, R. Mark, and J. Hahn, "A computational method for analysis of underwater dolphin kick hydrodynamics in human swimming," *Sports Biomechanics*, vol. 8, no. 1, pp. 60–77, 2009.

[16] R. J. Neal and B. D. Wilson, "3d kinematics and kinetics of the golf swing," *Journal of Applied Biomechanics*, vol. 1, no. 3, pp. 221–232, 1985.

[17] K. Williams, "A comparison of 2-d versus 3-d analyses of distance running kinematics," *Biomechanics, IX-B*, vol. 5, pp. 331–336, 1985.

[18] G. Ferrigno, N. Borghese, and A. Pedotti, "Pattern recognition in 3d automatic human motion analysis," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 45, no. 4, pp. 227–246, 1990.

[19] W. C. Whiting, R. J. Gregor, and G. A. Finerman, "Kinematic analysis of human upper extremity movements in boxing," *The American journal of sports medicine*, vol. 16, no. 2, pp. 130–136, 1988.

[20] R. Mooney, G. Corley, A. Godfrey, C. Osborough, J. Newell, L. R. Quinlan, and G. ÓLaighin, "Analysis of swimming performance: perceptions and practices of us-based swimming coaches," *Journal of sports sciences*, vol. 34, no. 11, pp. 997–1005, 2016.

[21] G. Pavlakos, L. Zhu, X. Zhou, and K. Daniilidis, "Learning to estimate 3d human pose and shape from a single color image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 459–468, 2018.

[22] S. Corazza, L. Mündermann, E. Gambaretto, G. Ferrigno, and T. P. Andriacchi, "Markerless motion capture through visual hull, articulated icp and subject specific model generation," *International journal of computer vision*, vol. 87, no. 1-2, p. 156, 2010.

[23] E. Ceseracciu, Z. Sawacha, S. Fantozzi, M. Cortesi, G. Gatta, S. Corazza, and C. Cobelli, "Markerless analysis of front crawl swimming," *Journal of biomechanics*, vol. 44, no. 12, pp. 2236–2242, 2011.

[24] B. Elliott and J. Alderson, "Laboratory versus field testing in cricket bowling: A review of current and past practice in modelling techniques," *Sports biomechanics*, vol. 6, no. 1, pp. 99–108, 2007.

[25] S. Blair, G. Duthie, S. Robertson, W. Hopkins, and K. Ball, "Concurrent validation of an inertial measurement system to quantify kicking biomechanics in four football codes," *Journal of biomechanics*, vol. 73, pp. 24–32, 2018.

[26] C. J. Payton and R. M. Bartlett, "Estimating propulsive forces in swimming from three-dimensional kinematic data," *Journal of sports sciences*, vol. 13, no. 6, pp. 447–454, 1995.

[27] R. Sanders, S. Psycharakis, C. McCabe, R. Naemi, C. Connaboy, L. Shuping, G. Scott, and A. Spence, "Analysis of swimming technique: state of the art; applications and implications.," *Portuguese Journal of Sports Sciences*, vol. 6, no. 2, pp. 20–24, 2006.

[28] G. Corley, R. Mooney, G. Ó Laighin, and L. Quinlan, "Application of video-based methods for competitive swimming analysis: a systematic review," *Sports and Exercise Medicine*, 2015.

[29] M. Paulich, M. Schepers, N. Rudigkeit, and G. Bellusci, "Xsens mtw awinda: Miniature wireless inertial-magnetic motion tracker for highly accurate 3d kinematic applications," *Xsens: Enschede, The Netherlands*, 2018.

[30] T. Seel, J. Raisch, and T. Schauer, "Imu-based joint angle measurement for gait analysis," *Sensors*, vol. 14, no. 4, pp. 6891–6909, 2014.

[31] B. Hanley, C. B. Tucker, and A. Bissas, "Differences between motion capture and video analysis systems in calculating knee angles in elite-standard race walking," *Journal of sports sciences*, vol. 36, no. 11, pp. 1250–1255, 2018.

[32] S. L. Colyer, M. Evans, D. P. Cosker, and A. I. Salo, "A review of the evolution of vision-based motion analysis and the integration of advanced computer vision methods towards developing a markerless system," *Sports medicine-open*, vol. 4, no. 1, p. 24, 2018.

[33] D. L. Benoit, D. K. Ramsey, M. Lamontagne, L. Xu, P. Wretenberg, and P. Renström, "Effect of skin movement artifact on knee kinematics during gait and cutting motions measured in vivo," *Gait & posture*, vol. 24, no. 2, pp. 152–164, 2006.

[34] M. Lafortune and M. Lake, "Errors in 3d analysis of human movement," in *Proc I Int Symp on 3-D Analysis of Hum Mov*, pp. 59–62, 1991.

[35] M. Cortesi, A. Giovanardi, G. Gatta, A. L. Mangia, S. Bartolomei, and S. Fantozzi, "Inertial sensors in swimming: Detection of stroke phases through 3d wrist trajectory," *Journal of Sports Science and Medicine*, vol. 18, no. 3, pp. 438–447, 2019.

[36] S. Fantozzi, A. Giovanardi, F. A. Magalhães, R. Di Michele, M. Cortesi, and G. Gatta, "Assessment of three-dimensional joint kinematics of the upper limb during simulated swimming using wearable inertial-magnetic measurement units," *Journal of sports sciences*, vol. 34, no. 11, pp. 1073–1080, 2016.

[37] K. Cahill-Rowley and J. Rose, "Temporal–spatial reach parameters derived from inertial sensors: Comparison to 3d marker-based motion capture," *Journal of biomechanics*, vol. 52, pp. 11–16, 2017.

[38] F. A. Magalhaes, Z. Sawacha, R. Di Michele, M. Cortesi, G. Gatta, and S. Fantozzi, "Effectiveness of an automatic tracking software in underwater motion analysis," *Journal of sports science & medicine*, vol. 12, no. 4, p. 660, 2013.

[39] A. Sabov and J. Krüger, "Identification and correction of flying pixels in range camera data," in *Proceedings of the 24th Spring Conference on Computer Graphics*, pp. 135–142, ACM, 2008.

[40] A. Kolb, E. Barth, R. Koch, and R. Larsen, "Time-of-flight cameras in computer graphics," in *Computer Graphics Forum*, vol. 29, pp. 141–159, Wiley Online Library, 2010.

[41] H. Sarbolandi, D. Lefloch, and A. Kolb, "Kinect range sensing: Structured-light versus time-of-flight kinect," *Computer vision and image understanding*, vol. 139, pp. 1–20, 2015.

[42] M. Livne, L. Sigal, M. Brubaker, and D. Fleet, "Walking on thin air: Environment-free physics-based markerless motion capture," in *2018 15th Conference on Computer and Robot Vision (CRV)*, pp. 158–165, IEEE, 2018.

[43] B. Carignan, M. Lauzé, C. Lavigne-Pelletier, H. Nguyen, L. Frossard, and C. Duval, "Effect of position on precision within the recording field of a markerless motion capture system," in *International Society for Posture and Gait Research (ISPGR) World Congress, 28 June - 2 July 2015, Seville, Spain*, 2015.

[44] H. R. Martinez, A. Garcia-Sarreon, C. Camara-Lemarroy, F. Salazar, and M. L. Guerrero-González, "Accuracy of markerless 3d motion capture evaluation to differentiate between

on/off status in parkinson's disease after deep brain stimulation," *Parkinson's Disease*, vol. 2018, 2018.

[45] A. Arnab, C. Doersch, and A. Zisserman, "Exploiting temporal context for 3d human pose estimation in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3395–3404, 2019.

[46] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik, "End-to-end recovery of human shape and pose," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7122–7131, 2018.

[47] Y. Huang, F. Bogo, C. Lassner, A. Kanazawa, P. V. Gehler, J. Romero, I. Akhter, and M. J. Black, "Towards accurate marker-less human shape and pose estimation over time," in *2017 International Conference on 3D Vision (3DV)*, pp. 421–430, IEEE, 2017.

[48] L. Sigal, A. O. Balan, and M. J. Black, "Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion," *International journal of computer vision*, vol. 87, no. 1-2, p. 4, 2010.

[49] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu, "Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 7, pp. 1325–1339, 2013.

[50] H. Rhodin, M. Salzmann, and P. Fua, "Unsupervised geometry-aware representation for 3d human pose estimation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 750–767, 2018.

[51] W. Yang, W. Ouyang, X. Wang, J. Ren, H. Li, and X. Wang, "3d human pose estimation in the wild by adversarial learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5255–5264, 2018.

[52] G. Pavlakos, X. Zhou, K. G. Derpanis, and K. Daniilidis, "Harvesting multiple views for marker-less 3d human pose annotations," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6988–6997, 2017.

[53] X. Zhou, Q. Huang, X. Sun, X. Xue, and Y. Wei, "Weakly-supervised transfer for 3d human pose estimation in the wild," in *IEEE International Conference on Computer Vision, ICCV*, vol. 3, p. 7, 2017.

[54] G. Pavlakos, X. Zhou, K. G. Derpanis, and K. Daniilidis, "Coarse-to-fine volumetric prediction for single-image 3d human pose," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7025–7034, 2017.

[55] D. Tome, C. Russell, and L. Agapito, "Lifting from the deep: Convolutional 3d pose estimation from a single image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2500–2509, 2017.

[56] A.-I. Popa, M. Zanfir, and C. Sminchisescu, "Deep multitask architecture for integrated 2d and 3d human sensing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6289–6298, 2017.

[57] J. Martinez, R. Hossain, J. Romero, and J. J. Little, "A simple yet effective baseline for 3d human pose estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2640–2649, 2017.

[58] D. Mehta, S. Sridhar, O. Sotnychenko, H. Rhodin, M. Shafiei, H.-P. Seidel, W. Xu, D. Casas, and C. Theobalt, "Vnect: Real-time 3d human pose estimation with a single rgb camera," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, p. 44, 2017.

[59] G. Rogez, P. Weinzaepfel, and C. Schmid, "Lcr-net: Localization-classification-regression for human pose," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3433–3441, 2017.

[60] B. Tekin, P. Márquez-Neila, M. Salzmann, and P. Fua, "Learning to fuse 2d and 3d image cues for monocular body pose estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3941–3950, 2017.

[61] H. Yasin, U. Iqbal, B. Kruger, A. Weber, and J. Gall, "A dual-source approach for 3d pose estimation from a single image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4948–4956, 2016.

[62] S. Corazza, E. Gambaretto, L. Mündermann, and T. P. Andriacchi, "Automatic generation of a subject-specific model for accurate markerless motion capture and biomechanical applications," *IEEE Transactions on biomedical engineering*, vol. 57, no. 4, pp. 806–812, 2008.

[63] H. Joo, T. Simon, and Y. Sheikh, "Total capture: A 3d deformation model for tracking faces, hands, and bodies," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8320–8329, 2018.

[64] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, "Smpl: A skinned multi-person linear model," *ACM transactions on graphics (TOG)*, vol. 34, no. 6, p. 248, 2015.

[65] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis, "Scape: shape completion and animation of people," in *ACM transactions on graphics (TOG)*, vol. 24, pp. 408–416, ACM, 2005.

[66] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[67] S. Saini, N. Zakaria, D. R. A. Rambli, and S. Sulaiman, "Markerless human motion tracking using hierarchical multi-swarm cooperative particle swarm optimization," *PLoS One*, vol. 10, no. 5, p. e0127833, 2015.

[68] R. C. Cohen, P. W. Cleary, S. M. Harrison, B. R. Mason, and D. L. Pease, "Pitching effects of buoyancy during four competitive swimming strokes," *Journal of applied biomechanics*, vol. 30, no. 5, pp. 609–618, 2014.

[69] S. Corazza, L. Muendermann, A. Chaudhari, T. Demattio, C. Cobelli, and T. P. Andriacchi, "A markerless motion capture system to study musculoskeletal biomechanics: Visual hull and simulated annealing approach," *Annals of biomedical engineering*, vol. 34, no. 6, pp. 1019–1029, 2006.

[70] A. Laurentini, "The visual hull concept for silhouette-based image understanding," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 16, no. 2, pp. 150–162, 1994.

[71] A. L. Sheets, G. D. Abrams, S. Corazza, M. R. Safran, and T. P. Andriacchi, "Kinematics differences between the flat, kick, and slice serves measured using a markerless motion capture method," *Annals of biomedical engineering*, vol. 39, no. 12, p. 3011, 2011.

[72] G. D. Abrams, A. H. Harris, T. P. Andriacchi, and M. R. Safran, "Biomechanical analysis of three tennis serve types using a markerless system," *Br J Sports Med*, vol. 48, no. 4, pp. 339–342, 2014.

[73] M. Rayat Imtiaz Hossain and J. J. Little, "Exploiting temporal information for 3d human pose estimation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 68–84, 2018.

[74] C. Lassner, G. Pons-Moll, and P. V. Gehler, "A generative model of people in clothing," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 853–862, 2017.

[75] F. Bogo, A. Kanazawa, C. Lassner, P. Gehler, J. Romero, and M. J. Black, "Keep it smpl: Automatic estimation of 3d human pose and shape from a single image," in *European Conference on Computer Vision*, pp. 561–578, Springer, 2016.

[76] M. A. Perrott, T. Pizzari, J. Cook, and J. A. McClelland, "Comparison of lower limb and trunk kinematics between markerless and marker-based motion capture systems," *Gait & posture*, vol. 52, pp. 57–61, 2017.

[77] T. Andriacchi, G. Andersson, R. Fermier, D. Stern, and J. Galante, "A study of lower-limb mechanics during stair-climbing.," *The Journal of bone and joint surgery. American volume*, vol. 62, no. 5, pp. 749–757, 1980.

[78] D. H. Wolpert, W. G. Macready, *et al.*, "No free lunch theorems for search," tech. rep., Technical Report SFI-TR-95-02-010, Santa Fe Institute, 1995.

[79] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7167–7176, 2017.

[80] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[81] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *International conference on machine learning*, pp. 1180–1189, PMLR, 2015.

[82] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, "Deep domain confusion: Maximizing for domain invariance," *arXiv preprint arXiv:1412.3474*, 2014.

[83] B. Sun, J. Feng, and K. Saenko, "Return of frustratingly easy domain adaptation," *arXiv preprint arXiv:1511.05547*, 2015.

[84] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[85] G. Ascenso, M. H. Yap, T. Allen, S. S. Choppin, and C. Payton, "A review of silhouette extraction algorithms for use within visual hull pipelines," *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, pp. 1–22, 2020.

[86] S.-C. S. Cheung and C. Kamath, "Robust background subtraction with foreground validation for urban traffic video," *EURASIP Journal on Advances in Signal Processing*, vol. 2005, no. 14, p. 726261, 2005.

[87] Y. Tian, A. Senior, and M. Lu, "Robust and efficient foreground analysis in complex surveillance videos," *Machine vision and applications*, vol. 23, no. 5, pp. 967–983, 2012.

[88] A. W. Senior, Y. Tian, and M. Lu, "Interactive motion analysis for video surveillance and long term scene monitoring," in *Asian Conference on Computer Vision*, pp. 164–174, Springer, 2010.

[89] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez, "A review on deep learning techniques applied to semantic segmentation," *arXiv preprint arXiv:1704.06857*, 2017.

[90] T. Bouwmans, "Traditional and recent approaches in background modeling for foreground detection: An overview," *Computer Science Review*, vol. 11, pp. 31–66, 2014.

[91] S. Y. Elhabian, K. M. El-Sayed, and S. H. Ahmed, "Moving object detection in spatial domain using background removal techniques-state-of-art," *Recent patents on computer science*, vol. 1, no. 1, pp. 32–54, 2008.

[92] L. Unzueta, M. Nieto, A. Cortés, J. Barandiaran, O. Otaegui, and P. Sánchez, "Adaptive multicue background subtraction for robust vehicle counting and classification," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 2, pp. 527–540, 2012.

[93] P. Blauensteiner and M. Kampel, *Visual surveillance of an airport's apron-An overview of the AVITRACK project*. na, 2004.

[94] A. Leykin and M. Tuceryan, "Detecting shopper groups in video sequences," in *Advanced Video and Signal Based Surveillance, 2007. AVSS 2007. IEEE Conference on*, pp. 417–422, IEEE, 2007.

[95] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, "Wallflower: Principles and practice of background maintenance," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 1, pp. 255–261, IEEE, 1999.

[96] D. Sakkos, H. Liu, J. Han, and L. Shao, "End-to-end video background subtraction with 3d convolutional neural networks," *Multimedia Tools and Applications*, pp. 1–19, 2017.

[97] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[98] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[99] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge & Data Engineering*, no. 9, pp. 1263–1284, 2008.

[100] Y. Wang, Z. Luo, and P.-M. Jodoin, "Interactive deep learning method for segmenting moving objects," *Pattern Recognition Letters*, vol. 96, pp. 66–75, 2017.

[101] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1915–1929, 2013.

[102] L. A. Lim and H. Y. Keles, "Foreground segmentation using a triplet convolutional neural network for multiscale feature encoding," *arXiv preprint arXiv:1801.02225*, 2018.

[103] C. Wojek and B. Schiele, "A dynamic conditional random field model for joint labeling of object and scene classes," in *European Conference on Computer Vision*, pp. 733–747, Springer, 2008.

[104] P. Krähenbühl and V. Koltun, "Efficient inference in fully connected crfs with gaussian edge potentials," in *Advances in neural information processing systems*, pp. 109–117, 2011.

[105] Z. Luo, P.-M. Jodoin, S.-Z. Li, and S.-Z. Su, "Traffic analysis without motion features," in *Image Processing (ICIP), 2015 IEEE International Conference on*, pp. 3290–3294, IEEE, 2015.

[106] F. Seide, G. Li, and D. Yu, "Conversational speech transcription using context-dependent deep neural networks," in *Twelfth annual conference of the international speech communication association*, 2011.

[107] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for lvcsr using rectified linear units and dropout," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pp. 8609–8613, IEEE, 2013.

[108] D. Cireşan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," *arXiv preprint arXiv:1202.2745*, 2012.

[109] A. Newswanger and C. Xu, "One-shot video object segmentation with iterative online fine-tuning," in *CVPR Workshop*, vol. 1, 2017.

[110] M. Mirza and S. Osindero, "Conditional generative adversarial networks," *Manuscript: https://arxiv. org/abs/1709.02023*, 2014.

[111] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015.

[112] L. A. Lim and H. Y. Keles, "Foreground segmentation using convolutional neural networks for multiscale feature encoding," *Pattern Recognition Letters*, vol. 112, pp. 256–262, 2018.

[113] L. A. Lim and H. Y. Keles, "Learning multi-scale features for foreground segmentation," *arXiv preprint arXiv:1808.01477*, 2018.

[114] N. Goyette, P.-M. Jodoin, F. Porikli, J. Konrad, and P. Ishwar, "changedetection.net: A new change detection benchmark dataset," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 06 2012.

[115] H. Olkkonen and P. Pesola, "Gaussian pyramid wavelet transform for multiresolution analysis of images," *Graphical Models and Image Processing*, vol. 58, no. 4, pp. 394–398, 1996.

[116] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[117] N. V. Chawla, N. Japkowicz, and A. Kotcz, "Special issue on learning from imbalanced data sets," *ACM Sigkdd Explorations Newsletter*, vol. 6, no. 1, pp. 1–6, 2004.

[118] A. Vacavant, T. Chateau, A. Wilhelm, and L. Lequièvre, "A benchmark dataset for outdoor foreground/background extraction," in *Asian Conference on Computer Vision*, pp. 291–300, Springer, 2012.

[119] J. C. Crane, M. P. Olson, and S. J. Nelson, "Sivic: open-source, standards-based software for dicom mr spectroscopy workflows," *Journal of Biomedical Imaging*, vol. 2013, p. 12, 2013.

[120] M. Bakkay, H. Rashwan, H. Salmane, L. Khoudour, D. Puigtt, and Y. Ruichek, "Bscgan: Deep background subtraction with conditional generative adversarial networks," in *2018 25th IEEE International Conference on Image Processing (ICIP)*, pp. 4018–4022, IEEE, 2018.

[121] P.-M. Jodoin, L. Maddalena, A. Petrosino, and Y. Wang, "Extensive benchmark and survey of modeling methods for scene background initialization," *IEEE Transactions on Image Processing*, vol. 26, no. 11, pp. 5244–5256, 2017.

[122] V. Mahadevan and N. Vasconcelos, "Spatiotemporal saliency in dynamic scenes," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 1, pp. 171–177, 2010.

[123] S. Boughorbel, F. Jarray, and M. El-Anbari, "Optimal classifier for imbalanced data using matthews correlation coefficient metric," *PloS one*, vol. 12, no. 6, p. e0177678, 2017.

[124] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[125] C. Lallier, E. Reynaud, L. Robinault, and L. Tougne, "A testing framework for background subtraction algorithms comparison in intrusion detection context," in *2011 8th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 314–319, IEEE, 2011.

[126] M. Thoma, "A survey of semantic segmentation," *arXiv preprint arXiv:1602.06541*, 2016.

[127] H. Zhu, F. Meng, J. Cai, and S. Lu, "Beyond pixels: A comprehensive survey from bottom-up to semantic image segmentation and cosegmentation," *Journal of Visual Communication and Image Representation*, vol. 34, pp. 12–27, 2016.

[128] X. Hou, A. Yuille, and C. Koch, "Boundary detection benchmarking: Beyond f-measures," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2123–2130, 2013.

[129] S. Chilamkurthy, "A 2017 guide to semantic segmentation with deep learning," 2017.

[130] H.-D. Cheng, X. H. Jiang, Y. Sun, and J. Wang, "Color image segmentation: advances and prospects," *Pattern recognition*, vol. 34, no. 12, pp. 2259–2281, 2001.

[131] Y. Raja, S. J. McKenna, and S. Gong, "Segmentation and tracking using colour mixture models," in *Asian Conference on Computer Vision*, pp. 607–614, Springer, 1998.

[132] B. Kim, J. Son, and K. Sohn, "Illumination invariant road detection based on learning method," in *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, pp. 1009–1014, IEEE, 2011.

[133] B. Li and M. Q.-H. Meng, "Computer-based detection of bleeding and ulcer in wireless capsule endoscopy images by chromaticity moments," *Computers in biology and medicine*, vol. 39, no. 2, pp. 141–147, 2009.

[134] C. Rotaru, T. Graf, and J. Zhang, "Color image segmentation in hsi space for automotive applications," *Journal of Real-Time Image Processing*, vol. 3, no. 4, pp. 311–322, 2008.

[135] L. Bourdev and J. Malik, "Poselets: Body part detectors trained using 3d human pose annotations," in *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 1365–1372, IEEE, 2009.

[136] L. Bourdev, S. Maji, T. Brox, and J. Malik, "Detecting people using mutually consistent poselet activations," in *European conference on computer vision*, pp. 168–181, Springer, 2010.

[137] T. Brox, L. Bourdev, S. Maji, and J. Malik, "Object segmentation by alignment of poselet activations to image contours," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pp. 2225–2232, IEEE, 2011.

[138] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[139] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, pp. 740–755, Springer, 2014.

[140] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[141] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[142] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2018.

[143] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," *arXiv preprint arXiv:1802.02611*, 2018.

[144] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," *arXiv preprint*, pp. 1610–02357, 2017.

[145] H. Qi, Z. Zhang, B. Xiao, H. Hu, B. Cheng, Y. Wei, and J. Dai, "Deformable convolutional networks–coco detection and segmentation challenge 2017 entry," in *ICCV COCO Challenge Workshop*, vol. 15, 2017.

[146] P. Dollár and C. L. Zitnick, "Structured forests for fast edge detection," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1841–1848, 2013.

[147] G. Lin, A. Milan, C. Shen, and I. D. Reid, "Refinenet: Multi-path refinement networks for high-resolution semantic segmentation.," in *Cvpr*, vol. 1, p. 5, 2017.

[148] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 2881–2890, 2017.

[149] G. Ghiasi and C. C. Fowlkes, "Laplacian pyramid reconstruction and refinement for semantic segmentation," in *European Conference on Computer Vision*, pp. 519–534, Springer, 2016.

[150] C. Peng, X. Zhang, G. Yu, G. Luo, and J. Sun, "Large kernel matters—improve semantic segmentation by global convolutional network," in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pp. 1743–1751, IEEE, 2017.

[151] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

[152] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pp. 5987–5995, IEEE, 2017.

[153] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[154] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," in *Artificial Intelligence and Statistics*, pp. 562–570, 2015.

[155] A. Aitken, C. Ledig, L. Theis, J. Caballero, Z. Wang, and W. Shi, "Checkerboard artifact free sub-pixel convolution: A note on sub-pixel convolution, resize convolution and convolution resize," *arXiv preprint arXiv:1707.02937*, 2017.

[156] L.-C. Chen, M. D. Collins, Y. Zhu, G. Papandreou, B. Zoph, F. Schroff, H. Adam, and J. Shlens, "Searching for efficient multi-scale architectures for dense image prediction," *arXiv preprint arXiv:1809.04184*, 2018.

[157] D. Golovin, B. Solnik, S. Moitra, G. Kochanski, J. Karro, and D. Sculley, "Google vizier: A service for black-box optimization," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1487–1495, ACM, 2017.

[158] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3213–3223, 2016.

[159] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *European Conference on Computer Vision*, pp. 746–760, Springer, 2012.

[160] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille, "The role of context for object detection and semantic segmentation in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 891–898, 2014.

[161] S. Song, S. P. Lichtenberg, and J. Xiao, "Sun rgb-d: A rgb-d scene understanding benchmark suite," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 567–576, 2015.

[162] T. Arlen, "Understanding the map evaluation metric for object detection," 2018.

[163] A. Bulat and G. Tzimiropoulos, "Human pose estimation via convolutional part heatmap regression," in *European Conference on Computer Vision*, pp. 717–732, Springer, 2016.

[164] Q. Dang, J. Yin, B. Wang, and W. Zheng, "Deep learning based 2d human pose estimation: A survey," *Tsinghua Science and Technology*, vol. 24, no. 6, pp. 663–676, 2019.

[165] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele, "2d human pose estimation: New benchmark and state of the art analysis," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[166] S. Johnson and M. Everingham, "Clustered pose and nonlinear appearance models for human pose estimation," in *Proceedings of the British Machine Vision Conference*, 2010. doi:10.5244/C.24.12.

[167] V. Ferrari, M. Marin-Jimenez, and A. Zisserman, "Progressive search space reduction for human pose estimation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[168] B. Sapp and B. Taskar, "Modec: Multimodal decomposable models for human pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3674–3681, 2013.

[169] S. Maji, "Large scale image annotations on amazon mechanical turk," *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2011-79*, 2011.

[170] U. Rafi, B. Leibe, J. Gall, and I. Kostrikov, "An efficient convolutional network for human pose estimation.," in *BMVC*, vol. 1, p. 2, 2016.

[171] X. Chu, W. Yang, W. Ouyang, C. Ma, A. L. Yuille, and X. Wang, "Multi-context attention for human pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1831–1840, 2017.

[172] L. Ke, M.-C. Chang, H. Qi, and S. Lyu, "Multi-scale structure-aware network for human pose estimation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 713–728, 2018.

[173] Y. Chen, C. Shen, X.-S. Wei, L. Liu, and J. Yang, "Adversarial posenet: A structure-aware convolutional network for human pose estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1212–1221, 2017.

[174] Y. Yang and D. Ramanan, "Articulated human detection with flexible mixtures of parts," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 12, pp. 2878–2890, 2012.

[175] S. Plagenhoef, F. G. Evans, and T. Abdelnour, "Anatomical data for analyzing human motion," *Research quarterly for exercise and sport*, vol. 54, no. 2, pp. 169–178, 1983.

[176] W. Tang, P. Yu, and Y. Wu, "Deeply learned compositional models for human pose estimation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 190–206, 2018.

[177] W. Yang, S. Li, W. Ouyang, H. Li, and X. Wang, "Learning feature pyramids for human pose estimation," in *proceedings of the IEEE international conference on computer vision*, pp. 1281–1290, 2017.

[178] D. A. Winter, *Biomechanics and motor control of human movement*. John Wiley & Sons, 2009.

[179] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler, "Efficient object localization using convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 648–656, 2015.

[180] M. Fieraru, A. Khoreva, L. Pishchulin, and B. Schiele, "Learning to refine human pose estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 205–214, 2018.

[181] Y. Raaj, H. Idrees, G. Hidalgo, and Y. Sheikh, "Efficient online multi-person 2d pose tracking with recurrent spatio-temporal affinity fields," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4620–4628, 2019.

[182] G. Gkioxari, A. Toshev, and N. Jaitly, "Chained predictions using convolutional neural networks," in *European Conference on Computer Vision*, pp. 728–743, Springer, 2016.

[183] C.-J. Chou, J.-T. Chien, and H.-T. Chen, "Self adversarial training for human pose estimation," in *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pp. 17–30, IEEE, 2018.

[184] V. Belagiannis and A. Zisserman, "Recurrent human pose estimation," in *2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*, pp. 468–475, IEEE, 2017.

[185] B. Xiao, H. Wu, and Y. Wei, "Simple baselines for human pose estimation and tracking," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 466–481, 2018.

[186] Z. Tang, X. Peng, S. Geng, L. Wu, S. Zhang, and D. Metaxas, "Quantized densely connected u-nets for efficient landmark localization," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 339–354, 2018.

[187] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, vol. 1. MIT press Cambridge, 2016.

[188] Y. S. Abu-Mostafa, "The vapnik-chervonenkis dimension: Information versus complexity in learning," *Neural Computation*, vol. 1, no. 3, pp. 312–317, 1989.

[189] R. Sutton, "The bitter lesson," *Incomplete Ideas (blog), March*, vol. 13, p. 12, 2019.

[190] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *2008 IEEE conference on computer vision and pattern recognition*, pp. 1–8, IEEE, 2008.

[191] L. Pishchulin, M. Andriluka, P. Gehler, and B. Schiele, "Strong appearance and expressive spatial models for human pose estimation," in *Proceedings of the IEEE international conference on Computer Vision*, pp. 3487–3494, 2013.

[192] S. Johnson and M. Everingham, "Learning effective human pose estimation from inaccurate annotation," in *CVPR 2011*, pp. 1465–1472, IEEE, 2011.

[193] L. Ladicky, P. H. Torr, and A. Zisserman, "Human pose estimation using a joint pixel-wise and part-wise formulation," in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3578–3585, 2013.

[194] I. S. Dhillon, Y. Guan, and B. Kulis, "Kernel k-means: spectral clustering and normalized cuts," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 551–556, 2004.

[195] Y. Bin, X. Cao, X. Chen, Y. Ge, Y. Tai, C. Wang, J. Li, F. Huang, C. Gao, and N. Sang, "Adversarial semantic data augmentation for human pose estimation," *arXiv preprint arXiv:2008.00697*, 2020.

[196] K. Gong, X. Liang, D. Zhang, X. Shen, and L. Lin, "Look into person: Self-supervised structure-sensitive learning and a new benchmark for human parsing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 932–940, 2017.

[197] P. KaewTraKulPong and R. Bowden, "An improved adaptive background mixture model for real-time tracking with shadow detection," in *Video-based surveillance systems*, pp. 135–144, Springer, 2002.

[198] K. Grauman, G. Shakhnarovich, and T. Darrell, "A bayesian approach to image-based visual hull reconstruction," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, vol. 1, pp. I–I, IEEE, 2003.

[199] J. Gall, C. Stoll, E. De Aguiar, C. Theobalt, B. Rosenhahn, and H.-P. Seidel, "Motion capture using joint skeleton tracking and surface estimation," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 1746–1753, IEEE, 2009.

[200] S. Nobuhara, Y. Tsuda, I. Ohama, and T. Matsuyama, "Multi-viewpoint silhouette extraction with 3d context-aware error detection, correction, and shadow suppression," *IPSJ Transactions on Computer Vision and Applications*, vol. 1, pp. 242–259, 2009.

[201] D. Vlasic, I. Baran, W. Matusik, and J. Popović, "Articulated mesh animation from multi-view silhouettes," in *ACM Transactions on Graphics (TOG)*, vol. 27, p. 97, ACM, 2008.

[202] Y. Furukawa and J. Ponce, "Carved visual hulls for image-based modeling," in *European Conference on Computer Vision*, pp. 564–577, Springer, 2006.

[203] L. Mundermann, S. Corazza, A. M. Chaudhari, E. J. Alexander, and T. P. Andriacchi, "Most favorable camera configuration for a shape-from-silhouette markerless motion capture system for biomechanical analysis," in *Videometrics VIII*, vol. 5665, p. 56650T, International Society for Optics and Photonics, 2005.

[204] S. Nobuhara and T. Matsuyama, "Deformable mesh model for complex multi-object 3d motion estimation from multi-viewpoint video," in *3D Data Processing, Visualization, and Transmission, Third International Symposium on*, pp. 264–271, IEEE, 2006.

[205] S. Lazebnik, Y. Furukawa, and J. Ponce, "Projective visual hulls," *International Journal of Computer Vision*, vol. 74, no. 2, pp. 137–165, 2007.

[206] M. Mikhnevich and P. Hebert, "Shape from silhouette under varying lighting and multi-viewpoints," in *2011 Canadian Conference on Computer and Robot Vision*, pp. 285–292, IEEE, 2011.

[207] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan, "Image-based visual hulls," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pp. 369–374, 2000.

[208] M. Mikhnevich and D. Laurendeau, "Shape from silhouette in space, time and light domains," in *Computer Vision Theory and Applications (VISAPP), 2014 international conference on*, vol. 3, pp. 368–377, IEEE, 2014.

[209] M. Mikhnevich, P. Hébert, and D. Laurendeau, "Unsupervised visual hull extraction in space, time and light domains," *Computer Vision and Image Understanding*, vol. 125, pp. 55–71, 2014.

[210] A. Krishnan, A. Almadan, and A. Rattani, "Understanding fairness of gender classification algorithms across gender-race groups," *arXiv preprint arXiv:2009.11491*, 2020.

[211] A. Torralba and A. A. Efros, "Unbiased look at dataset bias," in *CVPR 2011*, pp. 1521–1528, IEEE, 2011.

[212] J. Cohen, "Statistical power analysis," *Current directions in psychological science*, vol. 1, no. 3, pp. 98–101, 1992.

[213] G. Ascenso, M. H. Yap, T. B. Allen, S. S. Choppin, and C. Payton, "Fishnet: Learning to segment the silhouettes of swimmers," *IEEE Access*, vol. 8, pp. 178311–178321, 2020.

[214] K. Team, "Carvana image masking challenge–1st place winner's interview." https://medium.com/kaggle-blog/carvana-image-masking-challenge-1st-place-winners-interview-78fcc5c887a8 (accessed: 09.12.2020).

[215] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

[216] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," *arXiv preprint arXiv:1602.07261*, 2016.

[217] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European conference on computer vision*, pp. 630–645, Springer, 2016.

[218] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *arXiv preprint arXiv:1905.11946*, 2019.

[219] M. Windolf, N. Götzen, and M. Morlock, "Systematic accuracy and precision analysis of video motion capturing systems—exemplified on the vicon-460 system," *Journal of biomechanics*, vol. 41, no. 12, pp. 2776–2780, 2008.

[220] P. Merriaux, Y. Dupuis, R. Boutteau, P. Vasseur, and X. Savatier, "A study of vicon system positioning performance," *Sensors*, vol. 17, no. 7, p. 1591, 2017.

[221] S. L. Raghu, C.-k. Kang, P. Whitehead, A. Takeyama, and R. Conners, "Static accuracy analysis of vicon t40s motion capture cameras arranged externally for motion capture in constrained aquatic environments," *Journal of biomechanics*, vol. 89, pp. 139–142, 2019.

[222] B. H. Olstad, C. Zinner, D. Haakonsen, J. Cabri, and P.-L. Kjendlie, "A new approach for identifying phases of the breaststroke wave kick and calculation of feet slip using 3d automatic motion cracking," *BMS 2014—Proceedings*, pp. 195–199, 2014.

[223] J. Lauer, A. H. Rouard, and J. P. Vilas-Boas, "Upper limb joint forces and moments during underwater cyclical movements," *Journal of biomechanics*, vol. 49, no. 14, pp. 3355–3361, 2016.

[224] B. H. Olstad, J. R. Vaz, C. Zinner, J. M. Cabri, and P.-L. Kjendlie, "Muscle coordination, activation and kinematics of world-class and elite breaststroke swimmers during submaximal and maximal efforts," *Journal of sports sciences*, vol. 35, no. 11, pp. 1107–1117, 2017.

[225] B. H. Olstad, C. Zinner, J. R. Vaz, J. M. Cabri, and P.-L. Kjendlie, "Muscle activation in world-champion, world-class, and national breaststroke swimmers," *International journal of sports physiology and performance*, vol. 12, no. 4, pp. 538–547, 2017.

[226] L. D. Duffell, N. Hope, and A. H. McGregor, "Comparison of kinematic and kinetic parameters calculated using a cluster-based model and vicon's plug-in gait," *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine*, vol. 228, no. 2, pp. 206–210, 2014.

[227] U. Della Croce, A. Cappozzo, and D. C. Kerrigan, "Pelvis and lower limb anatomical landmark calibration precision and its propagation to bone geometry and joint angles," *Medical & biological engineering & computing*, vol. 37, no. 2, pp. 155–161, 1999.

[228] U. Della Croce, A. Leardini, L. Chiari, and A. Cappozzo, "Human movement analysis using stereophotogrammetry: Part 4: assessment of anatomical landmark misplacement and its effects on joint kinematics," *Gait & posture*, vol. 21, no. 2, pp. 226–237, 2005.

[229] A. Leardini, L. Chiari, U. Della Croce, and A. Cappozzo, "Human movement analysis using stereophotogrammetry: Part 3. soft tissue artifact assessment and compensation," *Gait & posture*, vol. 21, no. 2, pp. 212–225, 2005.

[230] D. Karlsson and A. Lundberg, "Accuracy estimation of kinematic data derived from bone anchored external markers," in *Proc III Int Symp on 3-D Analysis of Hum Mov*, pp. 27–30, 1994.

[231] C. Reinschmidt, A. Van Den Bogert, B. Nigg, A. Lundberg, and N. Murphy, "Effect of skin movement on the analysis of skeletal knee joint motion during running," *Journal of biomechanics*, vol. 30, no. 7, pp. 729–732, 1997.

[232] P. Westblad, K. Halvorsen, T. Hashimoto, I. Winson, and A. Lundberg, "Ankle-joint complex motion during stance phase of walking as measured by skin or bone anchored markers," in *Proceedings of the 6th International Symposium on 3D Analysis of Human Movement*, pp. 49–51, 2000.

[233] K. Manal, I. McClay, J. Richards, B. Galinat, and S. Stanhope, "Knee moment profiles during walking: errors due to soft tissue movement of the shank and the influence of the reference coordinate system," *Gait & posture*, vol. 15, no. 1, pp. 10–17, 2002.

[234] J. P. Holden, J. A. Orsini, K. L. Siegel, T. M. Kepple, L. H. Gerber, and S. J. Stanhope, "Surface movement errors in shank kinematics and knee kinetics during gait," *Gait & Posture*, vol. 5, no. 3, pp. 217–227, 1997.

[235] S. Tashman and W. Anderst, "Skin motion artifacts at the knee during impact movements," *Gait & Posture*, vol. 16, pp. 11–12, 2002.

[236] R. Stagni, S. Fantozzi, A. Cappello, F. Brigliadori, and A. Leardini, "Effect of skin motion artefacts on knee joint kinematics," in *IV World Congress of Biomechanics*, pp. 4–9, 2002.

[237] S. Fantozzi, R. Stagni, A. Cappello, M. Bicchierini, A. Leardini, and F. Catani, "Skin motion artefact characterization from 3d fluoroscopy and stereophotogrammetry," in *IV World Congress of Biomechanics*, pp. 4–9, 2002.

[238] C. Reinschmidt, A. Van Den Bogert, A. Lundberg, B. Nigg, N. Murphy, A. Stacoff, and A. Stano, "Tibiofemoral and tibiocalcaneal motion during walking: external vs. skeletal markers," *Gait & Posture*, vol. 6, no. 2, pp. 98–109, 1997.

[239] E. S. Grood and W. J. Suntay, "A joint coordinate system for the clinical description of three-dimensional motions: application to the knee," *Journal of biomechanical engineering*, vol. 105, no. 2, pp. 136–144, 1983.

[240] G. Cole, B. Nigg, J. Ronsky, and M. Yeadon, "Application of the joint coordinate system to three-dimensional joint attitude and movement representation: a standardization proposal," *Journal of biomechanical engineering*, vol. 115, no. 4A, pp. 344–349, 1993.

[241] J. Fuller, L.-J. Liu, M. Murphy, and R. Mann, "A comparison of lower-extremity skeletal kinematics measured using skin-and pin-mounted markers," *Human movement science*, vol. 16, no. 2-3, pp. 219–242, 1997.

[242] J. Houck, H. J. Yack, and T. Cuddeford, "Validity and comparisons of tibiofemoral orientations and displacement using a femoral tracking device during early to mid stance of walking," *Gait & Posture*, vol. 19, no. 1, pp. 76–84, 2004.

[243] K. A. Ball and M. R. Pierrynowski, "Modeling of the pliant surfaces of the thigh and leg during gait," in *Laser-Tissue Interaction IX*, vol. 3254, pp. 435–446, International Society for Optics and Photonics, 1998.

[244] E. J. Alexander and T. P. Andriacchi, "Correcting for deformation in skin-based marker systems," *Journal of biomechanics*, vol. 34, no. 3, pp. 355–361, 2001.

[245] E. J. Alexander, C. Bregler, and T. P. Andriacchi, "Non-rigid modeling of body segments for improved skeletal motion estimation," *COMPUTER MODELING IN ENGINEERING AND SCIENCES*, vol. 4, no. 3/4, pp. 351–364, 2003.

[246] T.-W. Lu and J. O'connor, "Bone position estimation from skin marker co-ordinates using global optimisation with joint constraints," *Journal of biomechanics*, vol. 32, no. 2, pp. 129–134, 1999.

[247] T. Lu and J. O'Connor, "Three-dimensional computer graphics-based modelling and mechanical analysis of the human locomotor system," in *Sixth International Symposium on the 3D Analysis of Human Movement*, pp. 1–4, 2000.

[248] S. Washino, D. L. Mayfield, G. A. Lichtwark, H. Mankyu, and Y. Yoshitake, "Swimming performance is reduced by reflective markers intended for the analysis of swimming kinematics," *Journal of biomechanics*, vol. 91, pp. 109–113, 2019.

[249] S. F. Hoerner, *Fluid-dynamic drag: theoretical, experimental and statistical information.* Hoerner Fluid Dynamics, 1992.

[250] J. Clarys, "Human morphology and hydrodynamics," *Swimming III*, vol. 8, pp. 3–41, 1979.

[251] P. Zamparo, G. Gatta, D. Pendergast, and C. Capelli, "Active and passive drag: the role of trunk incline," *European journal of applied physiology*, vol. 106, no. 2, pp. 195–205, 2009.

[252] G. Gatta, M. Cortesi, S. Fantozzi, and P. Zamparo, "Planimetric frontal area in the four swimming strokes: Implications for drag, energetics and speed," *Human movement science*, vol. 39, pp. 41–54, 2015.

[253] V. Passaro, A. Cuccovillo, L. Vaiani, M. De Carlo, and C. E. Campanella, "Gyroscope technology and applications: A review in the industrial perspective," *Sensors*, vol. 17, no. 10, p. 2284, 2017.

[254] R. W. Levi and T. Judd, "Dead reckoning navigational system using accelerometer to measure foot impacts," Dec. 10 1996. US Patent 5,583,776.

[255] F. Dadashi, F. Crettenand, G. P. Millet, and K. Aminian, "Front-crawl instantaneous velocity estimation using a wearable inertial measurement unit," *Sensors*, vol. 12, no. 10, pp. 12927–12939, 2012.

[256] Z. Zhang, D. Xu, Z. Zhou, J. Mai, Z. He, and Q. Wang, "Imu-based underwater sensing system for swimming stroke classification and motion analysis," in *2017 IEEE International Conference on Cyborg and Bionic Systems (CBS)*, pp. 268–272, IEEE, 2017.

[257] H. Schepers, M. Giuberti, and G. Bellusci, "Xsens mvn: Consistent tracking of human motion using inertial sensing," 2018.

[258] N. P. Davey, M. E. Anderson, and D. A. James, "An accelerometer-based system for elite athlete swimming performance analysis," in *Smart Structures, Devices, and Systems II*, vol. 5649, pp. 409–415, International Society for Optics and Photonics, 2005.

[259] S. Daukantas, V. Marozas, and A. Lukosevicius, "Inertial sensor for objective evaluation of swimmer performance," in *2008 11th International Biennial Baltic Electronics Conference*, pp. 321–324, IEEE, 2008.

[260] M. Bächlin, K. Förster, and G. Tröster, "Swimmaster: a wearable assistant for swimmer," in *Proceedings of the 11th international conference on Ubiquitous computing*, pp. 215–224, ACM, 2009.

[261] T. Le Sage, A. Bindel, P. Conway, L. Justham, S. Slawson, and A. West, "Development of a real time system for monitoring of swimming performance," *Procedia Engineering*, vol. 2, no. 2, pp. 2707–2712, 2010.

[262] T. Le Sage, P. Conway, L. Justham, S. Slawson, A. Bindel, and A. West, "A component based integrated system for signal processing of swimming performance," in *2010 International Conference on Signal Processing and Multimedia Applications (SIGMAP)*, pp. 73–79, IEEE, 2010.

[263] T. Le Sage, A. Bindel, P. Conway, L. Justham, S. Slawson, and A. West, "Embedded programming and real-time signal processing of swimming strokes," *Sports Engineering*, vol. 14, no. 1, p. 1, 2011.

[264] M. Bächlin and G. Tröster, "Swimming performance and technique evaluation with wearable acceleration sensors," *Pervasive and mobile computing*, vol. 8, no. 1, pp. 68–81, 2012.

[265] R. M. Hagem, D. V. Thiel, S. G. O'Keefe, N. Dahm, A. Stamm, and T. Fickenscher, "Smart optical wireless sensor for real time swimmers feedback," in *SENSORS, 2012 IEEE*, pp. 1–4, IEEE, 2012.

[266] N. Chakravorti, T. Le Sage, S. E. Slawson, P. P. Conway, and A. A. West, "Design and implementation of an integrated performance monitoring tool for swimming to extract stroke information at real time," *IEEE transactions on human-machine systems*, vol. 43, no. 2, pp. 199–213, 2013.

[267] R. M. Hagema, T. Haelsig, S. G. O'Keefe, A. Stamm, T. Fickenscher, and D. V. Thiel, "Second generation swimming feedback device using a wearable data processing system based on underwater visible light communication," *Procedia Engineering*, vol. 60, pp. 34–39, 2013.

[268] R. Hagem, D. Thiel, S. O'Keefe, and T. Fickenscher, "Real-time swimmers' feedback based on smart infrared (ssir) optical wireless sensor," *Electronics Letters*, vol. 49, no. 5, pp. 340–341, 2013.

[269] R. M. Hagem, S. G. O'Keefe, T. Fickenscher, and D. V. Thiel, "Self contained adaptable optical wireless communications system for stroke rate during swimming," *IEEE Sensors Journal*, vol. 13, no. 8, pp. 3144–3151, 2013.

[270] G. Andreoni, P. Perego, M. C. Fusca, R. Lavezzari, and G. C. Santambrogio, "Smart garments for performance and training assessment in sport," in *2014 4th International Conference on Wireless Mobile Communication and Healthcare-Transforming Healthcare Through Innovations in Mobile and Wireless Technologies (MOBIHEALTH)*, pp. 267–270, IEEE, 2014.

[271] E. Beanland, L. C. Main, B. Aisbett, P. Gastin, and K. Netto, "Validation of gps and accelerometer technology in swimming," *Journal of Science and Medicine in Sport*, vol. 17, no. 2, pp. 234–238, 2014.

[272] N. P. Davey, D. A. James, and M. E. Anderson, "Signal analysis of accelerometry data using gravity-based modeling," in *Microelectronics: Design, Technology, and Packaging*, vol. 5274, pp. 362–370, International Society for Optics and Photonics, 2004.

[273] D. A. James, N. Davey, and T. Rice, "An accelerometer based sensor platform for insitu elite athlete performance analysis," in *SENSORS, 2004 IEEE*, pp. 1373–1376, IEEE, 2004.

[274] N. Davey, M. Anderson, and D. A. James, "Validation trial of an accelerometer-based sensor platform for swimming," *Sports Technology*, vol. 1, no. 4-5, pp. 202–207, 2008.

[275] O. Thomas, P. Sunehag, G. Dror, S. Yun, S. Kim, M. Robards, A. Smola, D. Green, and P. Saunders, "Wearable sensor activity analysis using semi-markov models with a grammar," *Pervasive and Mobile Computing*, vol. 6, no. 3, pp. 342–350, 2010.

[276] A. S. Silva, M. V. Correia, A. J. Salazar, and C. M. Silva, "Wimu: wearable inertial monitoring unit-a mems-based device for swimming performance analysis," in *BIODEVICES 2011 - Proceedings of the International Conference on Biomedical Electronics and Devices*, pp. 87–93, SciTePress, 2011.

[277] U. Jensen, F. Prade, and B. M. Eskofier, "Classification of kinematic swimming data with emphasis on resource consumption," in *2013 IEEE International Conference on Body Sensor Networks*, pp. 1–5, IEEE, 2013.

[278] J. Lecoutere and R. Puers, "Wireless communication with miniaturized sensor devices in swimming," *Procedia Engineering*, vol. 72, pp. 398–403, 2014.

[279] Y. Ohgi, K. Kaneda, and A. Takakura, "Sensor data mining on the kinematical characteristics of the competitive swimming," *Procedia Engineering*, vol. 72, pp. 829–834, 2014.

[280] M. Topalovic, S. Eyers, V. Exadaktylos, J. Olbrecht, D. Berckmans, and J.-M. Aerts, "Online monitoring of swimmer training using a 3d accelerometer-identifying swimming and swimming style," in *Proceedings of the 2nd International Congress on Sports Sciences*

Research and Technology Support, pp. 111–115, SCITEPRESS–Science and Technology Publications, 2014.

[281] Y. Ohgi, M. Yasumura, H. Ichikawa, and C. Miyaji, "Analysis of stroke technique using acceleration sensor ic in freestyle swimming," *Eng. Sport*, vol. 250, pp. 503–511, 2000.

[282] Y. Ohgi, "Microcomputer-based acceleration sensor device for sports biomechanics-stroke evaluation by using swimmer's wrist acceleration," in *SENSORS, 2002 IEEE*, vol. 1, pp. 699–704, IEEE, 2002.

[283] Y. Ohgi, H. Ichikawa, M. Homma, and C. Miyaji, "Stroke phase discrimination in breaststroke swimming using a tri-axial acceleration sensor device," *Sports Engineering*, vol. 6, no. 2, pp. 113–123, 2003.

[284] M. Bachlin, K. Forster, J. Schumm, D. Breu, J. Germann, and G. Troster, "An automatic parameter extraction method for the 7× 50m stroke efficiency test," in *2008 Third International Conference on Pervasive Computing and Applications*, vol. 1, pp. 442–447, IEEE, 2008.

[285] S. E. Slawson, L. M. Justham, A. West, P. Conway, M. Caine, and R. Harrison, "Accelerometer profile recognition of swimming strokes (p17)," in *The engineering of sport 7*, pp. 81–87, Springer, 2009.

[286] M. Bächlin and G. Tröster, "Pervasive computing in swimming: A model describing acceleration data of body worn sensors in crawl swimming," in *2009 Joint Conferences on Pervasive Computing (JCPC)*, pp. 293–298, IEEE, 2009.

[287] B. Khoo, B. K. Lee, S. A. Senanayake, and B. D. Wilson, "System for determining within-stroke variations of speed in swimming (swiss)," in *2009 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pp. 1927–1932, IEEE, 2009.

[288] S. Slawson, P. Conway, L. Justham, T. Le Sage, and A. West, "Dynamic signature for tumble turn performance in swimming," *Procedia Engineering*, vol. 2, no. 2, pp. 3391–3396, 2010.

[289] B. Wright, M. G. Hinman, and J. M. Stager, "Accelerometry as a means of quantifying training distance and speed in competitive swimmers," in *Proceedings of the XIth International Symposium for Biomechanics & Medicine in Swimming, Oslo, Norway*, pp. 16–19, 2010.

[290] S. Daukantas, V. Marozas, A. Lukosevicius, D. Jegelevicius, and D. Kybartas, "Video and inertial sensors based estimation of kinematical parameters in swimming sport," in *Proceedings of the 6th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems*, vol. 1, pp. 408–411, IEEE, 2011.

[291] J. Lee, R. Leadbetter, Y. Ohgi, D. Thiel, B. Burkett, and D. A. James, "Quantifying and assessing biomechanical differences in swim turn using wearable sensors," *Sports Technology*, vol. 4, no. 3-4, pp. 128–133, 2011.

[292] A. Stamm, D. V. Thiel, B. Burkett, and D. A. James, "Towards determining absolute velocity of freestyle swimming using 3-axis accelerometers," *Procedia Engineering*, vol. 13, pp. 120–125, 2011.

[293] S. Slawson, L. Justham, P. Conway, T. Le-Sage, and A. West, "Characterizing the swimming tumble turn using acceleration data," *Proceedings of the Institution of Mechanical Engineers, Part P: Journal of Sports Engineering and Technology*, vol. 226, no. 1, pp. 3–15, 2012.

[294] F. Dadashi, G. Millet, and K. Aminian, "Gaussian process framework for pervasive estimation of swimming velocity with body-worn imu," *Electronics Letters*, vol. 49, no. 1, pp. 44–45, 2013.

[295] F. Dadashi, A. Arami, F. Crettenand, G. P. Millet, J. Komar, L. Seifert, and K. Aminian, "A hidden markov model of the breaststroke swimming temporal phases using wearable inertial measurement units," in *2013 IEEE international conference on body sensor networks*, pp. 1–6, Ieee, 2013.

[296] A. Stamm, D. A. James, B. B. Burkett, R. M. Hagem, and D. V. Thiel, "Determining maximum push-off velocity in swimming using accelerometers," *Procedia Engineering*, vol. 60, pp. 201–207, 2013.

[297] F. Dadashi, G. P. Millet, and K. Aminian, "A bayesian approach for pervasive estimation of breaststroke velocity using a wearable imu," *Pervasive and Mobile Computing*, vol. 19, pp. 37–46, 2015.

[298] L. Seifert, M. L'hermette, J. Komar, D. Orth, F. Mell, P. Merriaux, P. Grenet, Y. Caritu, R. Hérault, V. Dovgalecs, *et al.*, "Pattern recognition in cyclic and discrete skills performance from inertial measurement units," *Procedia Engineering*, vol. 72, pp. 196–201, 2014.

[299] C. W. Phillips, A. I. Forrester, D. A. Hudson, and S. R. Turnock, "Comparison of kinematic acquisition methods for musculoskeletal analysis of underwater flykick," *Procedia engineering*, vol. 72, pp. 56–61, 2014.

[300] J.-T. Zhang, A. C. Novak, B. Brouwer, and Q. Li, "Concurrent validation of xsens mvn measurement of lower limb joint angular kinematics," *Physiological measurement*, vol. 34, no. 8, p. N63, 2013.

[301] Y. Abdel-Aziz and H. Karara, "Direct linear transformation into object space coordinates in close-range photogrammetry, in proc. symp. close-range photogrammetry," *Urbana-Champaign*, pp. 1–18, 1971.

[302] C. Mills, M. Lomax, B. Ayres, and J. Scurr, "The movement of the trunk and breast during front crawl and breaststroke swimming," *Journal of sports sciences*, vol. 33, no. 4, pp. 427–436, 2015.

[303] R. H. Sanders, M. M. Fairweather, A. Alcock, and C. B. McCabe, "An approach to identifying the effect of technique asymmetries on body alignment in swimming exemplified by a case study of a breaststroke swimmer," *Journal of sports science & medicine*, vol. 14, no. 2, p. 304, 2015.

[304] G. R. Bernardina, P. Cerveri, R. M. Barros, J. C. Marins, and A. P. Silvatti, "Action sport cameras as an instrument to perform a 3d underwater motion analysis," *PloS one*, vol. 11, no. 8, p. e0160490, 2016.

[305] Y. Hayashi, M. Homma, and Z. Luo, "Optimal timing of dolphin kick during breaststroke underwater swimming movement," in *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 249–254, IEEE, 2015.

[306] P. Figueiredo, T. M. Barbosa, J. P. Vilas-Boas, and R. J. Fernandes, "Energy cost and body centre of mass' 3d intracycle velocity variation in swimming," *European journal of applied physiology*, vol. 112, no. 9, pp. 3319–3326, 2012.

[307] F. Puel, J. Morlier, M. Avalos, M. Mesnard, M. Cid, and P. Hellard, "3d kinematic and dynamic analysis of the front crawl tumble turn in elite male swimmers," *Journal of biomechanics*, vol. 45, no. 3, pp. 510–515, 2012.

[308] C. B. McCabe and R. H. Sanders, "Kinematic differences between front crawl sprint and distance swimmers at a distance pace," *Journal of sports sciences*, vol. 30, no. 6, pp. 601–608, 2012.

[309] V. Gourgoulis, A. Boli, N. Aggeloussis, A. Toubekis, P. Antoniou, P. Kasimatis, N. Vezos, M. Michalopoulou, A. Kambas, and G. Mavromatis, "The effect of leg kick on sprint front crawl swimming," *Journal of sports sciences*, vol. 32, no. 3, pp. 278–289, 2014.

[310] R. Bartlett, M. Bussey, and N. Flyger, "Movement variability cannot be determined reliably from no-marker conditions," *Journal of Biomechanics*, vol. 39, no. 16, pp. 3076–3079, 2006.

[311] C. J. Payton and R. M. Bartlett, *Biomechanical evaluation of movement in sport and exercise: the British Association of Sport and Exercise Sciences guide*. Routledge, 2017.

[312] R. E. Bahamonde and R. R. Stevens, "Comparison of two methods of manual digitization on accuracy and time of completion," in *ISBS-Conference Proceedings Archive*, 2006.

[313] S. G. Psycharakis and R. H. Sanders, "Shoulder and hip roll changes during 200-m front crawl swimming.," *Medicine and Science in Sports and Exercise*, vol. 40, no. 12, pp. 2129–2136, 2008.

[314] S. Ceccon, E. Ceseracciu, Z. Sawacha, G. Gatta, M. Cortesi, C. Cobelli, and S. Fantozzi, "Motion analysis of front crawl swimming applying cast technique by means of automatic tracking," *Journal of sports sciences*, vol. 31, no. 3, pp. 276–287, 2013.

[315] B. C. Elliott, J. A. Alderson, and E. R. Denver, "System and modelling errors in motion analysis: Implications for the measurement of the elbow angle in cricket bowling," *Journal of Biomechanics*, vol. 40, no. 12, pp. 2679–2685, 2007.

[316] R. H. Sanders, T. Gonjo, and C. B. McCabe, "Reliability of three-dimensional angular kinematics and kinetics of swimming derived from digitized video," *Journal of sports science & medicine*, vol. 15, no. 1, p. 158, 2016.

[317] A. J. Callaway, J. E. Cobb, and I. Jones, "A comparison of video and accelerometer based approaches applied to performance monitoring in swimming," *International Journal of Sports Science & Coaching*, vol. 4, no. 1, pp. 139–153, 2009.

[318] C. Zerpa, C. Lees, P. Patel, E. Pryzsucha, and P. Patel, "The use of microsoft kinect for human movement analysis," *International journal of sports science*, vol. 5, no. 4, pp. 120–127, 2015.

[319] H. E. Kloefkorn, T. R. Pettengill, S. M. Turner, K. A. Streeter, E. J. Gonzalez-Rothi, D. D. Fuller, and K. D. Allen, "Automated gait analysis through hues and areas (agatha): a method to characterize the spatiotemporal pattern of rat gait," *Annals of biomedical engineering*, vol. 45, no. 3, pp. 711–725, 2017.

[320] P. Fankhauser, M. Bloesch, D. Rodriguez, R. Kaestner, M. Hutter, and R. Siegwart, "Kinect v2 for mobile robot navigation: Evaluation and modeling," in *2015 International Conference on Advanced Robotics (ICAR)*, pp. 388–394, IEEE, 2015.

[321] M. H. Reyna, J. A. Pimentel, G. T. Toledo, and M. H. López, "A 3d playful framework for learning the components of a wind turbine, using kinect," *International Journal of Education and Development using ICT*, vol. 14, no. 2, 2018.

[322] L. Fisher, *Obstacle detection with Kinect V2 on a ground robot.* PhD thesis, University of Alaska Fairbanks, 2018.

[323] A. Schmitz, M. Ye, G. Boggess, R. Shapiro, R. Yang, and B. Noehren, "The measurement of in vivo joint angles during a squat using a single camera markerless motion capture system as compared to a marker based system," *Gait & posture*, vol. 41, no. 2, pp. 694–698, 2015.

[324] M. Gabel, R. Gilad-Bachrach, E. Renshaw, and A. Schuster, "Full body gait analysis with kinect," in *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 1964–1967, IEEE, 2012.

[325] E. Auvinet, F. Multon, C.-E. Aubin, J. Meunier, and M. Raison, "Detection of gait cycles in treadmill walking using a kinect," *Gait & posture*, vol. 41, no. 2, pp. 722–725, 2015.

[326] J. Preis, M. Kessel, M. Werner, and C. Linnhoff-Popien, "Gait recognition with kinect," in *1st international workshop on kinect in pervasive computing*, pp. 1–4, New Castle, UK, 2012.

[327] B. F. Mentiplay, L. G. Perraton, K. J. Bower, Y.-H. Pua, R. McGaw, S. Heywood, and R. A. Clark, "Gait assessment using the microsoft xbox one kinect: Concurrent validity and inter-day reliability of spatiotemporal and kinematic variables," *Journal of biomechanics*, vol. 48, no. 10, pp. 2166–2170, 2015.

[328] B. Sun, X. Liu, X. Wu, and H. Wang, "Human gait modeling and gait analysis based on kinect," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3173–3178, IEEE, 2014.

[329] E. E. Stone and M. Skubic, "Unobtrusive, continuous, in-home gait measurement using the microsoft kinect," *IEEE Transactions on Biomedical Engineering*, vol. 60, no. 10, pp. 2925–2932, 2013.

[330] S. Sridhar, A. Markussen, A. Oulasvirta, C. Theobalt, and S. Boring, "Watchsense: On- and above-skin input sensing through a wearable depth sensor," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pp. 3891–3902, ACM, 2017.

[331] S. Pasinetti, M. M. Hassan, J. Eberhardt, M. Lancini, F. Docchio, and G. Sansoni, "Performance analysis of the pmd camboard picoflexx time-of-flight camera for markerless motion capture applications," *IEEE Transactions on Instrumentation and Measurement*, 2019.

[332] F. L. Siena, B. Byrom, P. Watts, and P. Breedon, "Utilising the intel realsense camera for measuring health outcomes in clinical research," *Journal of medical systems*, vol. 42, no. 3, p. 53, 2018.

[333] R. Das and K. S. Kumar, "Gerosim: A simulation framework for gesture driven robotic arm control using intel realsense," in *2016 IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES)*, pp. 1–5, IEEE, 2016.

[334] E. Knippenberg, J. Verbrugghe, I. Lamers, S. Palmaers, A. Timmermans, and A. Spooren, "Markerless motion capture systems as training device in neurological rehabilitation: a systematic review of their use, application, target population and efficacy," *Journal of neuroengineering and rehabilitation*, vol. 14, no. 1, p. 61, 2017.

[335] H. Rhodin, N. Robertini, D. Casas, C. Richardt, H.-P. Seidel, and C. Theobalt, "General automatic human shape and motion capture using volumetric contour cues," in *European conference on computer vision*, pp. 509–526, Springer, 2016.

[336] Q. Wang, G. Kurillo, F. Ofli, and R. Bajcsy, "Evaluation of pose tracking accuracy in the first and second generations of microsoft kinect," in *2015 international conference on healthcare informatics*, pp. 380–389, IEEE, 2015.

[337] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *CVPR 2011*, pp. 1297–1304, Ieee, 2011.

[338] Z. Zhang, "Microsoft kinect sensor and its effect," *IEEE multimedia*, vol. 19, no. 2, pp. 4–10, 2012.

[339] "Continuous-wave modulation." `http://shannon.cm.nctu.edu.tw/comtheory/chap2.pdf`. Accessed: 2019-11-10.

[340] S. Choppin and J. Wheat, "The potential of the microsoft kinect in sports analysis and biomechanics," *Sports Technology*, vol. 6, no. 2, pp. 78–85, 2013.

[341] A. Anwer, S. S. A. Ali, A. Khan, and F. Mériaudeau, "Underwater 3-d scene reconstruction using kinect v2 based on physical models for refraction and time of flight correction," *IEEE Access*, vol. 5, pp. 15960–15970, 2017.