Markerless 3D Motion Capture of Animals in the Wild Using Trajectory Optimisation

Presented by:
Daniel John Joska

Prepared for:
Dr Amir Patel
Dept. of Electrical and Electronics Engineering
University of Cape Town

Submitted to the Department of Electrical Engineering at the University of Cape Town in partial fulfilment of the academic requirements for a Bachelor of Science degree in Mechatronics

November 11, 2020

Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this report from the work(s) of other people has been attributed, and has been cited and referenced.
3. This report is my own work.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof.

Signature:.....
D. J. Joska

Date:.....

Acknowledgments

This report is the result of many hours of hard work performed not only by myself, but by many others too. Without the contributions of those who provided help, guidance, and support, however small it may have seemed at the time, this project would not have even come to light.

First, I would be remiss not to extend a resounding thank you to my supervisor, Dr Amir Patel. His passion for the field of engineering has pushed this work - and many others - to new heights.

I would like to extend a special thanks to Alexander Mathis for the expert guidance on the use of his code, and his recommendations for improvements to be made to the project.

I would like to thank Liam Clark for his patience and invaluable advice in the adaptation of his work. Without his initial hard work, this project would never have existed.

To my parents, without whom I would be nowhere, I extend my most heartfelt thanks. I may never be able to repay all you've done for me, but know that I will be forever grateful.

Abstract

In many fields of research, it is often desirable to estimate the 3D pose of a subject - human, animal, or otherwise - from a set of 2D pose estimations taken from multiple views of the subject. This is particularly applicable in robotics, where the design of biomimetic robots requires thorough analysis of animal movements in specific situations.

This paper describes the design, implementation, and testing of a software program for the 3D motion tracking of user-defined species in the wild using trajectory optimisation. The core functionality involves building a kinematic model from . The trajectory optimisation was implemented in Python, making extensive use of the Pyomo trajectory optimisation package.

The method for trajectory optimisation presented in this paper outperforms standard 3D triangulation aided by sparse bundle adjustment in a variety of both qualitative and quantitative measures. The addition of 2D pairwise predictions for the pose of the subject further improves performance, with the added measurements helping provide more reasonable 3D estimates for situations where a body part is highly occluded or in an unusual configuration.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background to the study

3D motion tracking of various species of animals has been relevant to many fields in recent decades. Particularly, research into biomimetic robots has been flourishing; Boston Dynamics has presented a variety of biomimetic robots over recent years, including Spot, a dog-inspired robotic quadruped, and Atlas, a humanoid robot [10]. Animal neuroscientists are often interested in the gait and overall trajectory of animal test subjects, as this gives insight into factors such as symptoms of neurological damage, decision making, and overall behaviour. For these any many other applications, it is essential that accurate 3D pose information is obtained for the exact species being studied. Often, training data for these species is extremely limited or non-existent in the public domain.

Usually, 3D motion tracking is achieved through the use of markers placed on joints of interest. However, this is not always a tenable solution. Many animals, regardless of their level of domestication or training, become distressed by the process of marker placement. Some animals exist in extremely limited numbers in captivity, and the capturing and studying of wild animals is often dangerous or simply undesirable.

A far more desirable solution consists of a non-intrusive markerless method of 2D pose estimation, the information from which is then used to reconstruct a 3D skeleton of the animal from multiple 2D views of the same pose. This method introduces a fair amount of uncertainty into both the 2D pose estimation and 3D reconstruction, and so further processing of the 3D points is required to obtain a sufficiently accurate 3D pose reconstruction.

1.2 Objectives of this study

1.2.1 Problems to be investigated

This study entails the development of a software package for the 3D motion tracking of user-defined species. Due to the virtually limitless spread of animal limb morphology (in cases where the animal even has limbs), obtaining an accurate 3D reconstruction of a particular animal's pose is extremely challenging, especially for uncommon species lacking in significant prior research. Solutions currently exist for the 2D pose estimation of user-defined species, such as the deep convolutional neural network-based DeepLabCut [1], but these are often rife with inaccuracies. Furthermore, currently existing 3D reconstruction techniques based on these results are fairly rudimentary. The main problem addressed in this study is how to produce consistent, accurate 3D reconstructions of user-defined species that are robust enough to compensate for large inaccuracies in the presented 2D data.

1.2.2 Purpose of the study

Current technology encourages the use of markers on animals; considering the state of existing algorithms for determining 3D pose, the accuracy gained from tracking markers is simply too great a benefit when compared with adapting an algorithm to track an individual species. This study aims to provide an effective set of tools for the 3D motion tracking of any conceivable species, given that the user has some basic knowledge of the anatomy of said species. In doing so, it will reduce our dependency on potentially intrusive, and oftentimes dangerous, techniques for data collection from live animals in the wild. It should be noted that for the remainder of this study the term "in the wild" will be used to refer to any scenario not within the confines of a laboratory (ie. with a static and controlled background).

1.3 Scope and Limitations

This project includes the design of a software package for the robust reconstruction of a 3D skeleton from a user-defined species given initial multi-view 2D pose estimates. The area of markerless 2D pose estimation is well-researched and continues to see innovation at a rapid pace. This study will focus on the less widely-researched task of providing accurate 3D data given inaccurate 2D pose predictions, as well as performing post-processing to refine already existing 2D pose estimates. The work in this project is also largely predicated on previous findings presented in an M.Sc dissertation entitled "Markerless

3D Motion Capture of Cheetahs in the Wild" [11] where similar techniques were used to reconstruct a cheetah's pose from multiple 2D views. This project will generalise these techniques to be applied to any user-defined species.

This report will also not cover any specifics or theory behind camera calibration, intrinsic or extrinsic. Although used by the sparse bundle adjustment presented later in this document, the exact derivation behind the calibration of cameras is outside of the scope of this project.

1.4 Plan of development

This report will detail the design process for the above described software program. First, a comprehensive literature review is presented wherein the theory surrounding the areas of 2D and 3D pose estimation, trajectory optimisation, and state estimation is summarised.

Following this, the report will describe the design process undertaken, as well as the implementation and validation of the results obtained. In this section, the general theory outlined in the literature review is developed for our specific purposes.

The report then presents a results section where the outcomes of the project are evaluated, both qualitatively and quantitatively. Following this section, a discussion section will draw meaning from the presented results and describe the impact of these results in relation to existing fields of research and the aforementioned goals of this project.

Finally, this report will summarise the project's outcomes in a conclusion section, and make recommendations for future related work.

Chapter 2

Literature Review

The following chapter evaluates existing literature in the field of motion capture, pose estimation, trajectory optimisation, and state estimation, and provides a summary of relevant theory.

2.1 2D Pose Estimation

Pose estimation is the process of estimating the configuration of a body, animal or otherwise, from a single 2D image. Although this project consists mostly of post-processing already estimated pose data, efficient pose estimation lies at the heart of this problem; without an effective method of obtaining an animal subject's pose from a 2D image, the initial triangulation of 2D data to a 3D state estimation is impossible. Pose estimation is a widely applicable tool and is therefore, thankfully, well-documented. Most research is understandably centred around human pose estimation, with offerings such as OpenPose providing accurate real-time 2D pose data for multiple human subjects appearing in a video [12]. The general form of this pose information is a simplified skeleton of the human subject formed from a handful of predefined joints, the positions of which have been estimated by some algorithm. An example of the output of such an algorithm is shown in Figure 2.1.

2.1.1 Convolutional Neural Networks

The vast majority of pose estimation algorithms make use of deep convolutional neural networks (CNNs). CNNs excel at computer vision-related tasks involving the processing of digital 2D images, oftentimes outclassing even human performance given enough training data. CNNs commonly yield impressive results in the area of pose estimation in the wild, advancing the state of the art in tasks such as head pose estimation [13], 3D hand

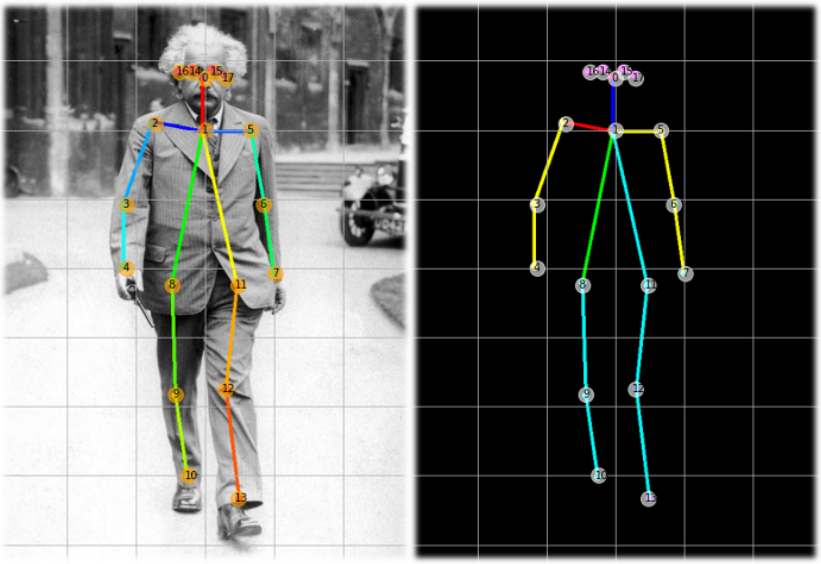

Figure 2.1: An example of the output data of a 2D pose estimation algorithm for a human subject

pose estimation [14], and - pertaining to this thesis - 2D pose estimation across species in the wild [15] [16] [1]. The general process for training a CNN to detect features for a novel species involves hand-labelling a large amount of training images containing the subject species and training the CNN to convergence on the aforementioned dataset. The network may then be evaluated (qualitatively, quantitatively, or both) and retrained if it does not produce satisfactory results. This study will largely make use of DeepLabCut [1] as a practical reference for 2D pose estimation across species.

Due to the nature of their architecture, CNNs are generally only as good as their training data. While their performance remains impressive and continues to improve in recent years, the fact that they require such a large increment in training data to see significant performance improvements is a hindrance in situations where images of a test subject are not plentiful. This is the case with many uncommon species, and the effect is compounded when motion capture of a completely novel test subject or situation is required. One widely implemented method for overcoming this hindrance is to pre-train the neural network in question on the extensive and publicly available ImageNet database [17]. ImageNet, and other similar datasets, provides a huge and diverse hierarchal database of images grouped by class. By pre-training on these datasets, CNNs are able to achieve far faster convergence times and more favourable outcomes in terms of performance. Additionally, large image datasets such as ImageNet are commonly used to benchmark neural networks and evaluate different techniques for feature extraction and overall performance gains [18] [19].

Given however the inherently analogue nature of pose estimation, even the most efficient

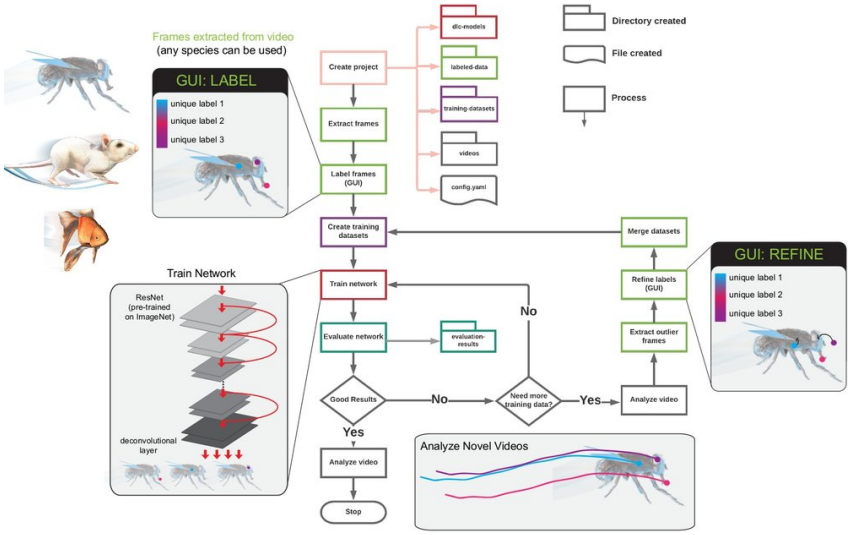

Figure 2.2: An overview of the training and evaluation cycle for DeepLabCut [1]

and precise neural networks are wrought with inaccuracies that impact a researcher's analysis of the test subject. The optimisation of 2D pose estimation is a neverending and labour-intensive process, limited greatly by compute power, problem complexity, and human error. The iterative nature of CNNs compounds this weakness. Shown in Figure 2.2 is an example of the process of refining and re-training a neural network until satisfactory results are obtained - in this case, the software package considered is the previously mentioned DeepLabCut [1], a relatively efficient offering in the pose estimation field. CNNs seldom make use of temporal information for inference outside of post-processing, and are therefore prone to errors a human would never make. One proposition this study aims to address is the effectiveness of highly constrained state estimation for smooth motion tracking of a subject over an entire section of its trajectory. By extension, achieving robust 3D motion tracking over a length of time may be used to further refine 2D pose estimation at particular instants in time.

2.1.2 Pairwise Prediction

Once a CNN is able to determine the location of a particular body part within an image with a reasonable degree of certainty, this may be exploited to derive probabilities of the presence of other related body parts near the first or "base" body part. The advantages of this idea are twofold:

1. The introduction of additional measurements for each body part may help refine the 2D pose estimations through regularisation, and by extension, may help improve the 3D reconstruction of these poses.

2. The probability of a certain body part to be found within a certain distance or

array of vectors from a base body part may be used to limit the search area for subsequent predictions of that body part. This leads to more computationally efficient and accurate pose predictions.

The human pose estimation network DeeperCut [20] made use of pairwise terms to predict vectors from each body part towards the expected location of the next body part. The study found the method to be suitably accurate to improve the performance of the network. Although pairwise predictions may be less accurate in more occluded and challenging poses, the expected distance between pairs of body parts is enough to provide a benefit for the overall pose estimation method.

The general method for collecting pairwise statistics during training of the CNN involves defining a tuple, $t^k_{cc'}$, containing the relative positions of every other body part c' for a base body part c across each of the training images of index k. This tuple is represented in the equation below.

$$t^k_{cc'} = (x_{c'} - x_k,\ y_{c'} - y_k) \tag{2.1}$$

The vectors obtained by the program during stats collection are visualised in Figure 2.3. During ordinary training of the CNN, three loss functions are combined in a linear fashion for efficiency: the cross-entropy loss used for the normal body part detectors, a location regression, and a pairwise regression. Typically, the pairwise regression will converge much faster than the other loss functions.

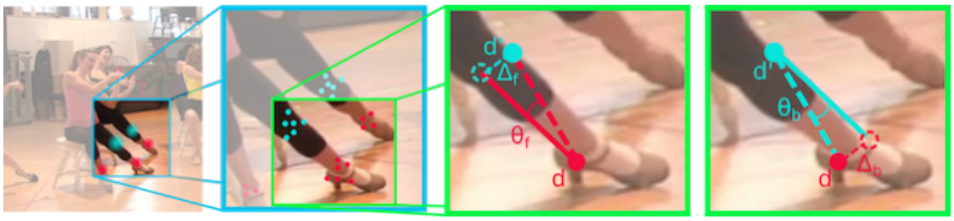

Figure 2.3: A visualisation of the vectors obtained for each body part during pairwise stats collection

The original DeeperCut paper [20] found not only that pairwise predictions improved performance significantly by 14.4% AP on the MPII Multi-person Val benchmark, but that the introduction of pairwise terms to limit search area for each consecutive body part drastically reduced the run-time of the program from 259220 seconds/frame to 1987 seconds/frame. It can be gleaned that including pairwise inference in a combined neural network trained for both typical body part detection and pairwise predicted body parts results in a superior 2D pose estimation model.

2.2 3D Pose Estimation

Much like 2D pose estimation, the vast majority of research into 3D pose estimation is centred around human subjects. However, the theoretical bases for this process are extendable into any conceivable species with some adaptation of the particular algorithm. Studies into human subjects will thus continue to inform our approach for animal subjects.

2.2.1 Triangulation

Triangulation is a method for estimating a 3D point - or series of points - from two or more separate corresponding 2D points captured from two or more separate views. Hartley and Sturm [21] present an optimal solution to the triangulation of a point from two corresponding views in the image planes. Generally, Gaussian noise is assumed in the measurement of the point in question as triangulation of a point where noise is not present at all is trivial. A visualisation of the process of triangulation is shown in Figure 2.4.

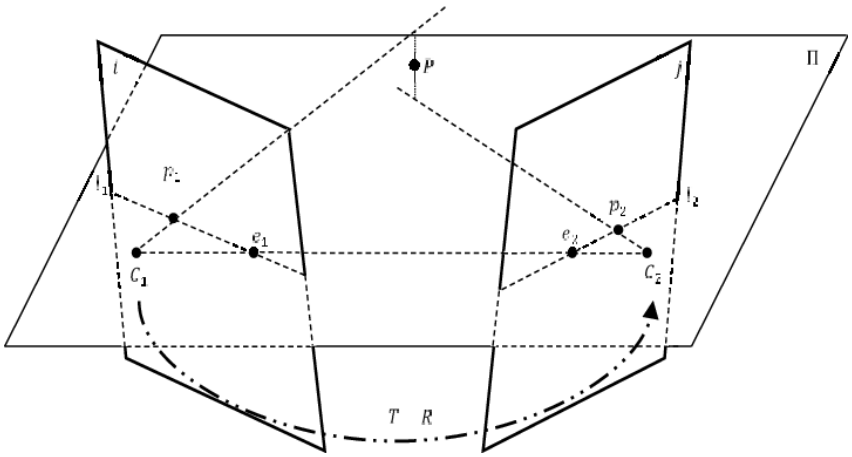

Figure 2.4: A visualisation of the triangulation of a point from two 2D views [2]

While triangulation is useful for the derivation of 3D pose given multiple 2D viewpoints, careful curating of the 2D data is required. Errors in the coordinates of the corresponding 2D points may be amplified in the triangulation process, resulting in 3D estimates with high variance. This is why triangulation is framed as an optimisation problem. With a Gaussian noise model, the optimisation may be constructed as a least-squares minimisation problem [21].

In order to perform an optimisation for a 3D triangulation problem, we of course require an error metric to minimise. This metric is generally chosen as the *reprojection error*. The reprojection error refers to a Euclidean distance measured from some given 2D ground truth point to the reprojected 3D point when transformed back to the image plane in

question. The ground truth point is usually defined quite simply as the initial 2D image plane point. This way, the best 3D estimate that fits all given 2D views is calculated. However, the reprojection error may also be compared with hand-labelled 2D ground truth data to obtain a metric for the accuracy of 3D reconstruction method.

2.2.2 Sparse Bundle Adjustment

Sparse Bundle Adjustment (SBA) is similar to 3D triangulation in that it refers to a process used to minimise the reprojection error of an estimated 3D point given two or more views in the image plane. SBA differs from basic triangulation in that additional parameters are considered for the optimisation, including camera calibration parameters (both intrinsic and extrinsic). Sparse bundle adjustment is therefore useful not only for a more effective triangulation, but for optimal camera calibration for use with other methods [22].

The formulation of a sparse bundle adjustment problem involves the use of a Jacobian matrix. The Jacobian matrix of a vector is a matrix comprised of the derivatives of each variable with respect to time as shown below.

$$\mathrm{J} = \begin{bmatrix} \frac{\delta f_1}{\delta x_1} & \cdots & \frac{\delta f_1}{\delta x_n} \\ \vdots & \ddots & \vdots \\ \frac{\delta f_m}{\delta x_1} & \cdots & \frac{\delta f_m}{\delta x_n} \end{bmatrix} \tag{2.2}$$

For a sparse bundle adjustment, the vector in question is composed of the set of camera parameters P_k for the k^{th} camera and the set of 3D points to be bundle adjusted as denoted by X_i (where $i = 1, 2, ...m$ for m points). This results in the parameter vector shown below.

$$\theta = [\mathrm{P}_1,\ \mathrm{P}_2,\ ...,\ \mathrm{P}_n,\ \mathrm{X}_1,\ \mathrm{X}_2,\ ...,\ \mathrm{X}_m] \tag{2.3}$$

Corresponding to each 3D point X_i are a set of 2D points, x_j^i, that describe the coordinates of the 3D point captured in the k^{th} image plane. The partial derivative of these 2D points must be found in order to calculate the Jacobian matrix for the vector.

The Jacobian matrix for the parameter vector is thus represented by the equation shown below.

$$\mathrm{J} = \frac{\delta \mathrm{x}}{\delta \theta} \tag{2.4}$$

However, only the corresponding camera parameters for each 2D point reprojected to the

relevant image plane may be none-zero. Furthermore, only the corresponding 3D point parameterised in θ may similarly be non-zero; other 3D points derived with respect to the 2D point in question simply yield a rate of change of 0. This fact contributes the word "sparse" to the concept of sparse bundle adjustment. Since the majority of the entries in the Jacobian matrix fall away to zero, we need only consider a small portion of non-zero rates of change stored in the matrix. This fact may be exploited by algorithms to compute sparse bundle adjustments problems far more quickly than problems involving densely populated matrices of the same size.

A visualisation of a sparse Jacobian matrix related to an SBA problem is shown below in Figure 2.5. This Jacobian matrix concerns a three-camera problem. As the number of cameras k increases, the percentage of populated entries in the Jacobian matrix reduces even further.

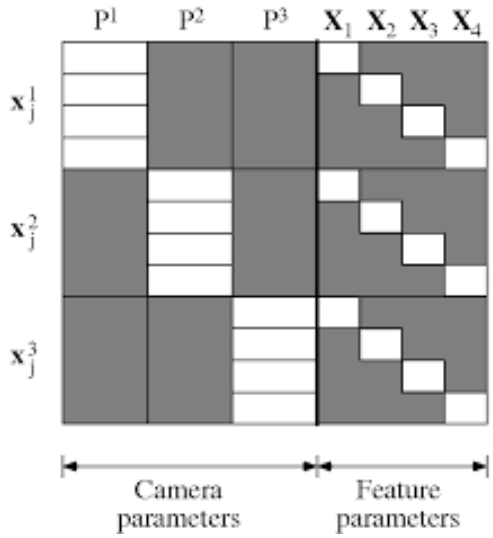

Figure 2.5: A visualisation of the Jacobian matrix for an example SBA problem. Grey zones indicate zero (or null) values, whereas bright zones denote non-zero partial derivatives

As seen above, the Jacobian matrix consists of two main sections: the camera parameters and feature parameters. The optimisation inherent in a sparse bundle adjustment problem minimises the error between these parameters, the relation between which is a simple reprojection function. This way, both the camera and feature parameters are made to agree as closely as possible for a particular 3D triangulation problem.

2.2.3 Neural Networks for 3D Pose Estimation

The problem of 3D pose estimation, especially when considered "in the wild" and without the advantage of a plain static background, is a challenging one. Much of this arises as a result of extremely limited training datasets available for public use. Again, when

one considers the problem of motion tracking "in the wild," this problem is compounded. Even concerning humans, the most widely studied subjects in the field of computer vision, there is a distinct lack of 3D training data. Given the high amount of training data needed to produce an accurate algorithm for the estimation of a 3D pose, many researchers have used a variety of methods either to augment existing training data artificially, or to produce more accurate 3D pose estimations from the data that exists.

Chen and Ramanan [3] obtained competitive 3D human pose estimation results from a single 2D image - eliminating the need for triangulation at all - using a technique involving "matching" poses from a library of predefined commonly occurring 3D poses to an image and using a typical CNN-based approach to refine the estimations with 2D pose estimation. The estimation of a 3D pose from a single 2D image is an extremely difficult problem even for machine learning-based approaches, and although this method is not comparable to human accuracy, it outperforms state-of-the-art pose estimation algorithms with relatively little training data.

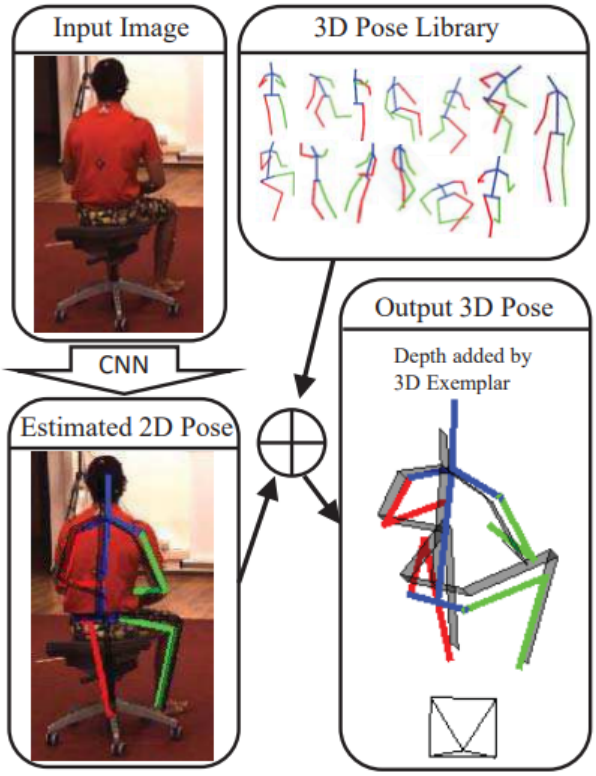

Figure 2.6: The matching method for 3D pose estimation from a single image [3]

Zhao et al [4] employed a deep neural network-based method for 3D pose estimation in the wild by combining 2D and 3D data labels into a single CNN. As opposed to other more common approaches for the estimation of 3D pose configurations from 2D labelled data, this method dispenses the need for multiple neural networks and instead aims to produce

a single network that has seen both 2D and 3D pose data. This way, the algorithm is able to minimise losses between 3D poses and 2D reprojections more easily and with greater efficacy. A summary of this technique is shown in Figure 2.7.

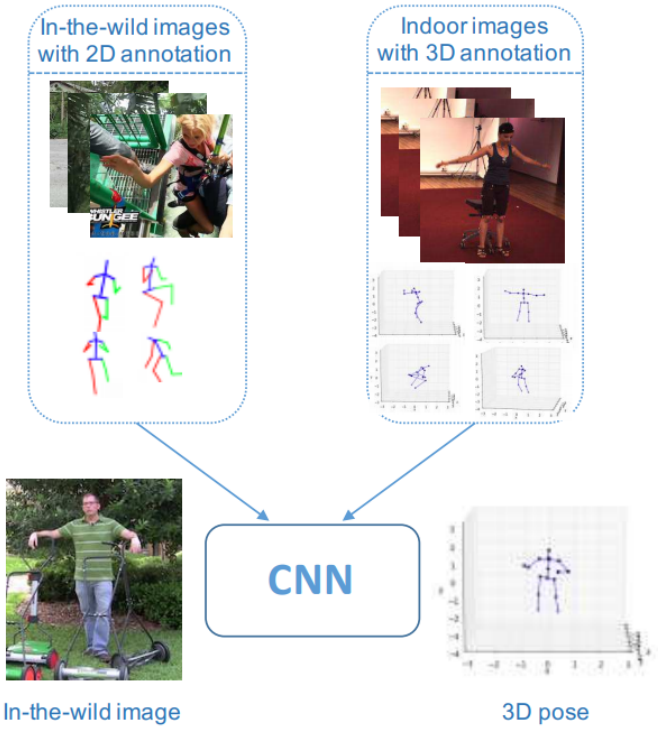

Figure 2.7: An overview of the process used by Zhao et al to combine 2D and 3D pose labels for 3D pose estimation [4]

With regard to non-human subjects, research has been undertaken using a single network trained to detect the 2D pose of a subject from multiple views and performing a stereo calibration of the 2D data. This method has the aforementioned disadvantage of high sensitivity to inaccuracies in the 2D data which - as is often the case with uncommon species or challenging computer vision problems - tends to arise as a result of a lack of training data.

Recent research involving the use of DeepLabCut for 3D markerless cheetah pose estimation yielded an easily repeatable process, given access to enough training data for the 2D neural network. Shown in Figure 2.8 is a diagram describing the general process of obtaining a 3D pose from a cheetah in the wild using a DeepLabCut-trained CNN for 2D pose detection [5].

The key problem with existing studies here is that a large amount of research into human subjects has been conducted with a wide variety of methods for enhancing performance, but advanced methods are rarely applied to a more general 3D pose estimation approach for a user-defined species. Ample training data exists for human subjects in both 2D and

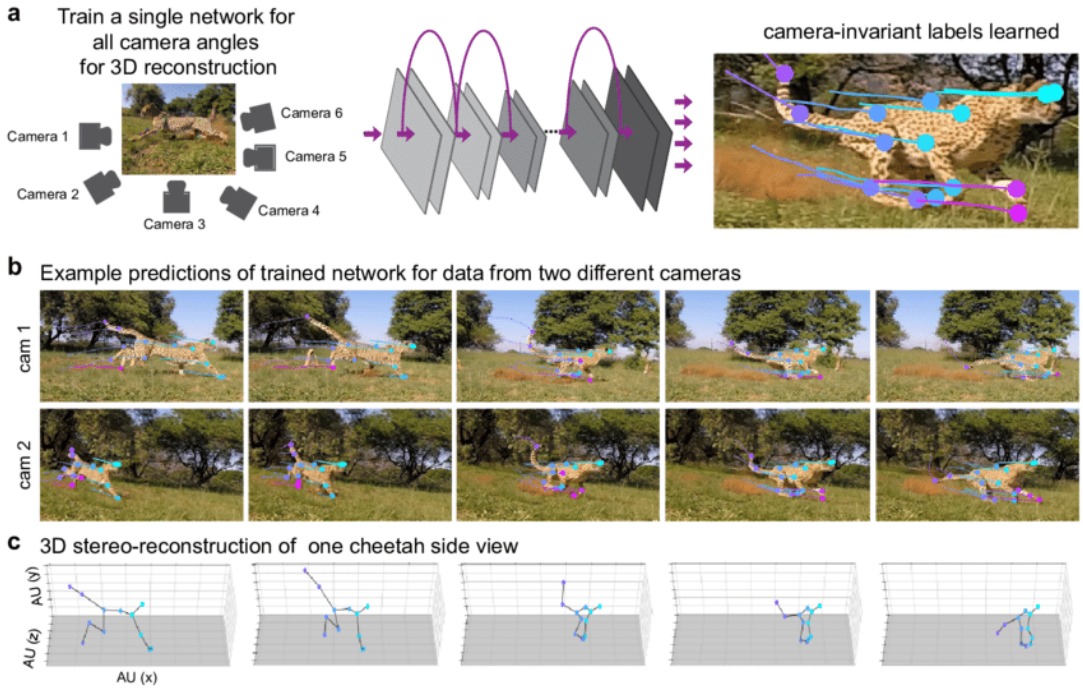

Figure 2.8: A visual overview of the process of performing a stereo reconstruction of a cheetah pose captured from multiple views using DeepLabCut [5]

3D form - why, then, is there so little research into the applications of more advanced methods for 3D pose estimation of uncommon species for which ample training data does not exist? The lack of adequate training data for species that are not so commonly studied as humans only serves to heighten the need for new, more robust approaches for 3D pose estimation.

2.3 State Estimation

State estimation is the process of calculating a set of parameters called *states* that describe some core aspects of a complex real-world system. These parameters must be calculated using some array of sensor data obtained from the real system. The main engineering problem that arises in state estimation is how to calculate sufficiently accurate states from sensor data that is often noisy or highly inaccurate.

To improve the accuracy of state estimations, the values yielded through any "prediction" calculations are often compared to the original sensor data to obtain an error. This error is then propagated back through the prediction loop to correct the resulting state estimations in a negative feedback loop. This is known as an "update" step. A simple block diagram illustrating an implementation of this process is shown in Figure 2.9.

There are many algorithms used for state estimation. One effective method that sees widespread use is the Kalman filter. In the original M.Sc thesis on cheetah motion

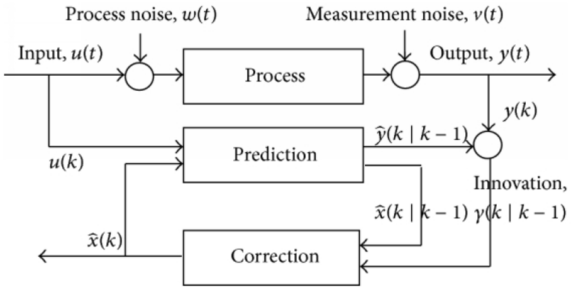

Figure 2.9: A simplified block diagram of a generally applicable state estimator for a process [6]

tracking [11], an extended Kalman filter (EKF) was used as a faster and less computationally intensive alternative for trajectory optimisation. It was found that the performance of the EKF, although very slightly worse overall than the full trajectory optimisation, provided competitive results at a fraction of the processing time. For applications demanding live interpretation of the results of a 3D pose estimation algorithm, this method may be best.

A Kalman filter consists of two main algorithm stages: the prediction step and the update step. In a discrete implementation, both steps are generally performed successively and repeatedly after a predefined time step has passed. An extended Kalman filter works similarly to an ordinary Kalman filter, but while both are discrete systems, the EKF is designed to estimate the states of a nonlinear system whereas the normal KF is designed around linear systems.

Shown in Figure 2.10 is a flowchart containing the general equations used in each step of the Kalman filtering algorithm. The exact mathematics behind the development of a Kalman filter are specific to the application, and this process is outside of the scope of this project. However, some lessons to be learned from the Kalman filtering algorithm are relevant to our investigation.

The main aspect of Kalman filtering that is relevant to our state estimation problem in general is the process model. The process model used in a Kalman filtering algorithm is generally derived from the state-space representation of the system [7]. This is a type of mathematical representation of a system in question that is generally given in the form showed in the equation below.

$$\dot{\mathrm{x}} = \mathbf{A}\mathrm{x} + \mathbf{B}\mathrm{u} + \mathrm{w} \tag{2.5}$$

Here, x is a vector containing the states of the system. Importantly, these states will be particular to the specific application of the algorithm as not all conceivable states associated with a system, however simple, are of interest. The vector u represents a control input and the matrices $\mathbf{A}$ and $\mathbf{B}$ represent the system dynamics and control

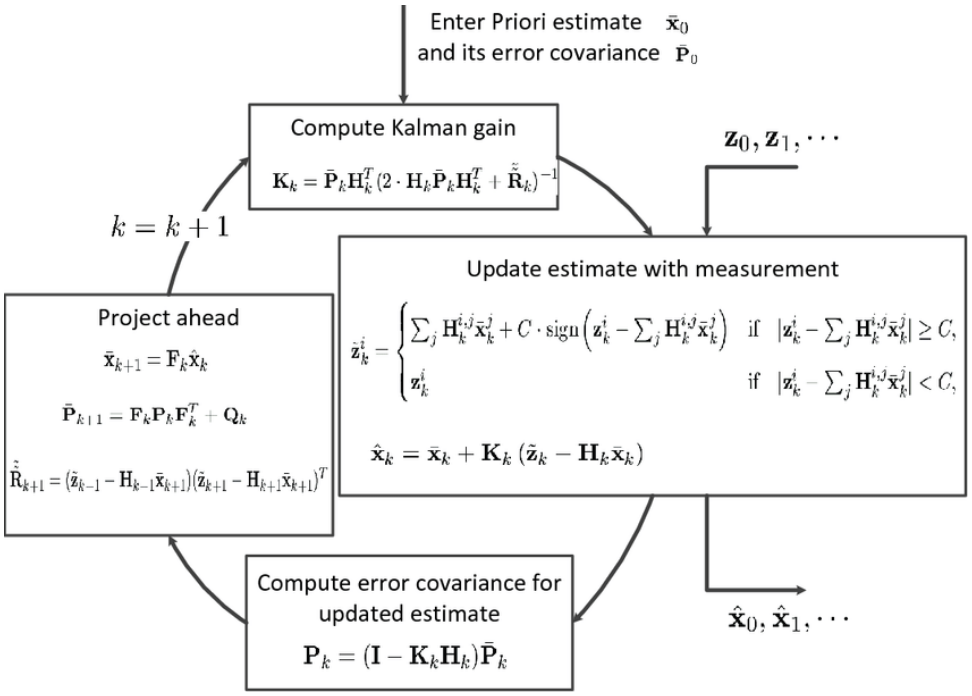

Figure 2.10: A diagram specifying a Kalman filtering algorithm including the prediction and update steps [7]

system model respectively.

Following the derivation of a state space model, it becomes necessary to obtain a vector w denoting the process noise. Process noise is present to some degree in all real systems, and efficiently dealing with this noise is crucial to the effective development of a state estimation algorithm.

The process noise at an instant k is generally assumed to be a function of random Gaussian noise distribution with a covariance matrix defined as $\mathbf{Q}_k$. Additionally, some measurement noise is present in the transformation from state space to measurement space. This noise may be denoted as the vector v where v_k is the measurement noise at an instant k. Analogous to the process noise, the measurement noise has a covariance matrix $\mathbf{P}_k$ specific to each insant k.

2.4 Trajectory Optimisation

Trajectory optimisation refers to the derivation of a trajectory for a point - or another more complex body - in a manner that minimises a chosen cost function. Constraints may also be imposed on the trajectory with a variety of intentions. These constraints must be met by the optimisation algorithm in order for the trajectory to be considered valid.

Trajectory optimisations are used extensively in the field of robotics. Often, it is necessary for a robot to calculate a route of movement from point A to point B that minimises power usage, risk, or other such factors. In these cases a cost function will most likely include contributions from several separate functions. A robot is also constrained both by its own morphology and by the layout of its surroundings - these are examples of constraint functions that must be imposed on the algorithm.

Zeng and Zhang [23] presented a method for developing a wireless communication system for UAVs that made use of trajectory optimisation to minimise energy usage. They not only provide a derivation for optimal trajectories that may be travelled by the drones to transmit and receive wireless signals, but propose a set of constraints on the general flight path that reduce energy usage through the limitation of wasteful high-velocity and high-acceleration maneuvers. This study is only one example of the many applications of trajectory optimisation within the field of mechatronic design.

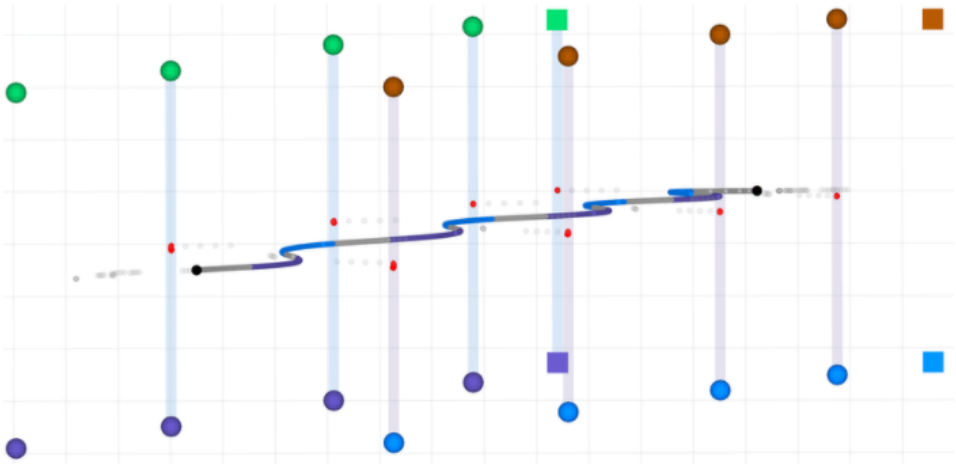

Figure 2.11: A plotted optimised trajectory for a quadrupedal robot [8]

Shown in Figure 2.11 is the output of a trajectory optimisation algorithm for a quadrupedal robot [8]. The design of legged robots typically makes extensive use of trajectory optimisation due to the complex nature of the control problem inherent in balancing and moving a legged robot in a manner that uses the least energy possible. Patel and Shield [9] found through the use of a high number of trajectory optimisations that the addition of a free stabilising limb to a bipedal robot greatly reduces stopping distance during high-speed maneuvers. For problems such as this, it is typical to compute thousands of trajectory optimisations with varying robot morphology (among other parameters) across iterations. Figure 2.12 shows a bipedal robot model with 7 degrees of freedom that was used in the aforementioned trajectory optimisations.

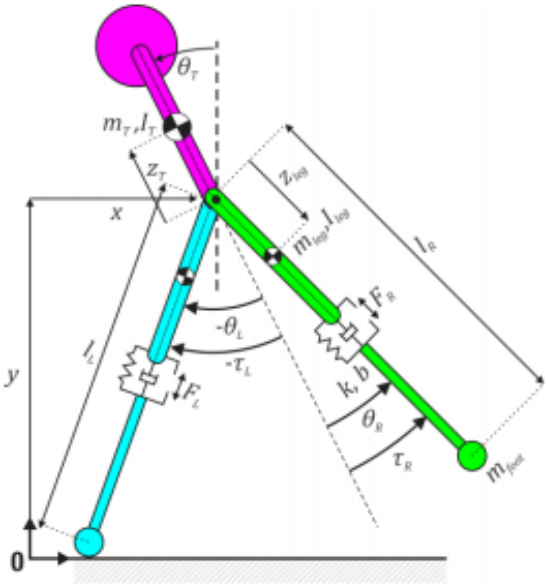

Figure 2.12: A bipedal robot model used to study methods of rapid deceleration in legged robots[9]

2.5 Summary

Upon close examination of the literature surrounding 3D pose estimation, 2D pose estimation, state estimation, and trajectory optimisation, an effective path forward for this project may be deduced. Although much of the literature is centred around human pose estimation, the insights gained from these studies are invaluable to informing our approach.

Firstly, we require as efficient a neural network as possible for the obtaining of 2D pose estimations for a user-defined species. For this task, DeepLabCut [1] seems the natural choice. With functionality for the 2D pose estimation of any conceivable user-defined species given enough training data, and a strong body of previous work performed using DeepLabCut-trained CNNs to track the 3D pose of cheetahs, the software package is well-suited to our needs.

Since DeepLabCut was predicated directly on the findings presented in DeeperCut [20], which was in turn an improvement on DeepCut, the methods used in the architecture of the original DeeperCut algorithm are easily extendable to DeepLabCut. The evidence for the performance increases seen from integrating pairwise pose predictions into a 2D pose estimation CNN is strong, in terms of both accuracy and computational efficiency. Since it may be assumed that the dataset for a user-defined species will be fairly small (at least, not nearly enough to provide adequate performance in 3D space), pairwise part predictions for the enhancement of the pose detection system are ideal for minimising inaccuracies.

The use of triangulation, and the more complete sparse bundle adjustment, is sub-

optimal for the computation of 3D pose parameters given 2D pose data for multiple views. No machine-labelled 2D pose data is perfect, and inaccuracies in the 2D data cause disproportionately large errors in the 3D triangulations due to the nature of the algorithm. Nevertheless, sparse bundle adjustment remains as an effective means to calculate the intrinsic and extrinsic calibrations of a set of cameras.

Although it is relatively computationally expensive, the most accurate and temporally consistent method for the transformation of 2D pose data to 3D pose estimates is a full trajectory optimisation. An effective and accurate state estimation algorithm is required for accurate results; however, trajectory optimisation remains a far more robust method to outliers than typical triangulation or SBA. This method assumes that the results are not needed in real-time. For a far quicker method with comparable accuracy, an extended Kalman filter (EKF) may be usde to smooth the 3D results as presented in the preceding M.Sc. thesis on the 3D motion tracking of cheetahs [11].

Chapter 3

Design

This section documents the design process undertaken for this project. It will include overviews of the preliminary design, implementation, and a description of the testing procedure for the program. Where possible, algorithms will be described via flowcharts. All code used to obtain these results is included in a GitHub repository linked in the appendices.

3.1 Graphical User Interface

The first problem with universal 3D motion tracking across species is the capturing of the subject morphology from the user. The code itself will build a mechanical model based on input vectors representing subject limbs and other body parts; however, it is impractical to demand the user to input these vectors directly from a terminal or other Text User Interface (TUI). It becomes necessary, then, to develop a simple graphical user interface (GUI) where the user is able to build a model of the subject piece-by-piece. This section will provide a short overview of the design of the GUI.

3.1.1 Skeleton Builder Panel Design

The first and most important aspect of the GUI is the aforementioned model building panel. This panel contains all components needed to create and visualise an animal "skeleton" that represents the general shape of the subject species. Note that this project assumes that the user already has data on the 2D pose of the animal for the motion in question from multiple camera views, and that the user has already performed either a simple intrinsic and extrinsic camera calibration or a sparse bundle adjustment. This panel is therefore used to plot already existing body part labels and link them to create a skeleton. The skeleton builder panel is shown in Figure 3.1

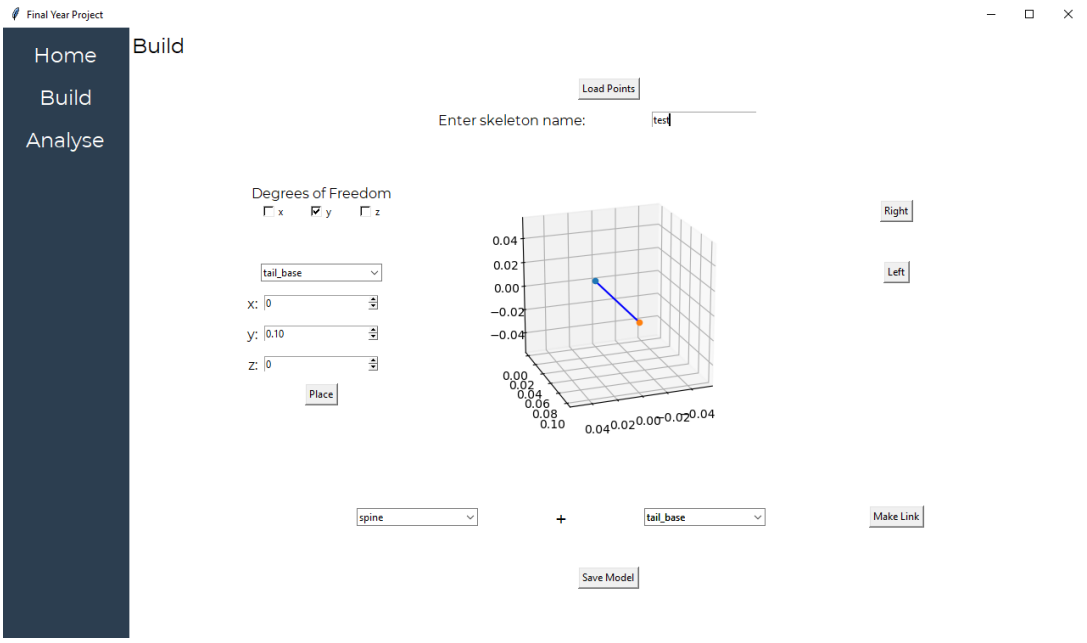

Figure 3.1: The skeleton builder GUI screen presented to the user for the capturing of subject morphology

The user is provided with a view-port in which the assembled skeleton may be seen. The GUI panel presents the user with the option to load predicted 2D poses from existing analysed 2D data. From this data, the comboboxes are populated with the bodypart labels. The user is then able to place a selected bodypart at the desired location specified by x, y, and z coordinates. The degrees of freedom for each joint (up to three for the x, y, and z axes) are specified by the checkboxes. Once placed, the user may define links between pairs of bodyparts through the two comboboxes at the bottom of the panel. Finally, once the skeleton is fully defined, the user may save the built skeleton. This writes all entered skeleton parameters to a .pickle file for later use by the program.

3.1.2 Implementation

The programming language selected for this project was Python. This was the natural choice for a few reasons:

1. Python is well-suited to machine learning- and data science-related tasks. Many of the most widely used packages in the field of data science are programmed primarily in Python.

2. The original M.Sc thesis on which this project is based was programmed in Python. This will streamline the process of adapting the code to be applicable to various species.

3. This project will make use of DeepLabCut [1] as its main 2D pose estimation component. DeepLabCut was developed in Python and, once again, the process will be much more streamlined if we are to stay consistent.

For the handling of the GUI, Python's native package TKinter was used. TKinter was chosen as it is extremely well-established (being over two decades old) and is cross-platform. TKinter provides a wide array of simple and effective functions for the handling of GUI components. Although fairly rudimentary, TKinter provides ample functionality for the purposes of this project.

Shown in Figure 3.2 is a basic overview of the process undertaken for the capturing of a user-defined skeleton. The user is able to move or replace unsatisfactory body parts simply by placing the same label at the desired location. The process is designed to be quick and efficient. With use of the GUI, the capturing of mechanical information from the user for the species to be analysed is far quicker than it would be if the user is required to enter skeleton information purely in text form. Depending on skeleton complexity, the process typically takes 5 to 10 minutes.

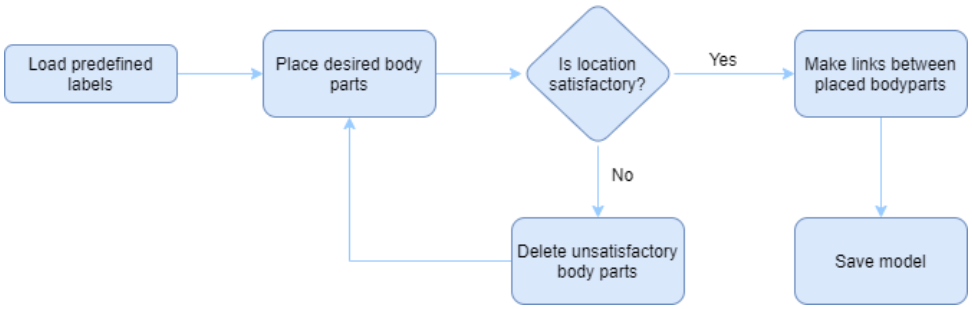

Figure 3.2: Flow chart showing the basic process involved in capturing a user-defined skeleton

3.2 Building Mechanical Models

The next challenge inherent in performing a trajectory optimisation for a user-defined skeleton is the algorithmic development of a kinematic model representative of the subject species. This kinematic model will be used to constrain the trajectory optimisation to within physically allowable poses. For this task, we require the application of several core mechanical design processes.

3.2.1 Mathematical Problem Description

The development of a kinematic model first requires the conceptualisation of a set of generalised coordinates to define the pose of the subject at any given point in time. These

generalised coordinates should be applicable to any conceivable user-defined skeleton, and must be effectively utilised by subsequent trajectory optimisation processes to obtain satisfactory results.

The absolute position of the subject may be simply defined as a point in 3D space. For simplicity, this point will correspond to one of the body parts placed by the user. This "base body part" will then be used further to calculate the relative positions of all remaining body parts.

$$p = [x \quad y \quad z]^T \tag{3.1}$$

Note that the transpose of a matrix is denoted by $\mathbf{M}^T$. This results in a matrix where the rows and columns have been switched when compared with the original matrix.

Once we have defined the global position of the subject as above, the remaining pose parameters consist solely of rotation transformations. Since the skeleton of the subject is assumed to be rigid, we need not further define the lengths of each limb; instead, we need only consider the angles through which the limb in question has been rotated. To help define the remaining pose parameters, we will make use of generalised rotation matrices describing rotation about the x, y, and z axes as shown below.

$$\mathbf{R}_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} \tag{3.2}$$

$$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{3.3}$$

$$\mathbf{R}_z(\psi) = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

Each of the above matrix transformations is applied when a given joint has been assigned a degree of freedom about the corresponding axis. Note that the transpose of one of these rotation matrices is equivalent to its inverse.

To complete the generalised coordinate system to be used by the program, we require expressions for the 3D pose of each labelled body part in terms of the base body part and rotations. These are to be built progressively as one recedes further from the base body part, with each part being defined by its previous linked part. A vector is drawn

from the base body part to the next linked body part as described by the user-defined skeleton. If the base body part position is denoted by p_{prev}, and the next body part in the link is denoted by p_{next}, the vector describing the position of the linked part in terms of the base part is as shown below.

$$v_{link} = p_{next} - p_{prev} = [x_n - x_p \; y_n - y_p \; z_n - z_p]^T = [x_l \;\; y_l \;\; z_l]^T \tag{3.5}$$

Similarly, the rotation matrix governing the degrees of freedom of the next part in a link, $\mathbf{R}_{next}$, is shown below. This assumes three degrees of freedom (about the x, y, and z axes) but for joints with fewer DOFs the corresponding fundamental rotation matrices may be omitted.

$$\mathbf{R}_{next} = \mathbf{R}_z(\psi_{next})\mathbf{R}_x(\phi_{next})\mathbf{R}_y(\theta_{next})\mathbf{R}_{prev} \tag{3.6}$$

Finally, the position of the next part in a link is given by the equation shown below.

$$p_{next} = p_{prev} + \mathbf{R}_{next} * (v_{link})^T \tag{3.7}$$

3.2.2 Algorithmic Implementation

The mechanical models built by the program make use of the Python library Pyomo [24] [25]. Pyomo is a mechanical modelling package designed to build trajectory optimisation problems using a handful of relevant functions to add constraints, build objective functions, and store parameters in a model object. The particulars of how Pyomo was used to structure the trajectory optimisation problem will be discussed in the following section.

Shown below in Figure 3.3 is a flow chart summarising the steps taken by the algorithm when building a Pyomo model from a user-defined skeleton. The part positions, links between the parts, and DOFs are stored in a dictionary that is built during the skeleton saving stage. From this dictionary, the mechanical model is built using generalised coordinates as described above. Near the end of the process, the symbolic pose functions are lambdified using SciPy. This is used to improve processing speed when dealing with complex equations such as those produced by the algorithm. Once lambdified, the pose model is passed on to be used by the trajectory optimisation program.

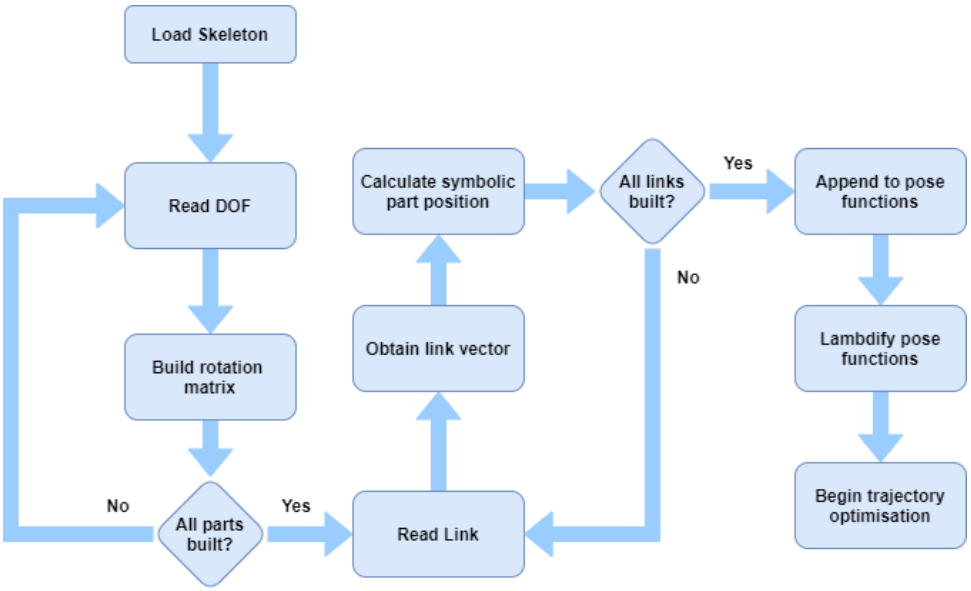

Figure 3.3: Flow chart showing how a mechanical model is built from a user-defined skeleton

3.3 Trajectory Optimisation

As mentioned in the previous section, the Python package Pyomo was used to build the trajectory optimisation problem. Before the problem may be solved, constraints must be imposed on it and a cost function must be developed. This section describes these processes in detail.

3.3.1 Model Constraints

The first constraint that must be imposed on the problem is the limiting of the subject pose such that it does not violate the constructed mechanical model (or, more simply, the geometry of the user-defined skeleton). This would eliminate physical impossibilities in the subject's pose that a human would intuitively discard.

A constant acceleration model was also applied to exclude temporal impossibilities such as the unnaturally high speed required to reach certain poses over a trajectory. To achieve this, an implicit Euler integration was performed on the discrete states representing position, velocity, and acceleration throughout the trajectory. These integrations were formulated as shown below. Note that acceleration noise is accounted for through the constant approximation $n_{acc} = 0.1$

$$x_i = x_{i-1} + \Delta t \dot{x}_i \tag{3.8}$$

$$\dot{x}_i = \dot{x}_{i-1} + \Delta t \ddot{x}_i \tag{3.9}$$

$$\ddot{x}_i = \ddot{x}_{i-1} \pm 0.1 \tag{3.10}$$

Additionally, all angles were constrained to less than 180 deg or π radians. This constraint was added to limit unnatural positions in the subject's pose where the subject's anatomy would not allow such flexibility.

3.3.2 Measurement Constraints

The second constraint necessary to the problem is the measurement constraint. This refers to the objective of the optimisation to minimise the error between neural-network labelled 2D pose estimations and the projected 3D points. There are two stages inherent to the formulation of this constraint:

1. Obtain a set of 3D point locations corresponding to each of the joints of the subject at any given point in time, which must be calculated from the generalised coordinates
2. Reproject these 3D points to each 2D image plane so that they may be compared with the neural network's estimates and the error may be minimised

This approach is far more computationally efficient than obtaining the 2D image plane coordinates directly from the pose parameters at each point in time, as discovered in the original cheetah pose estimation thesis [cite]. This constraint then requires the use of two functions. The first obtains the 2D point coordinates s_i at some instant i from the pose parameters x_i such that the relationship shown below is obtained.

$$\mathrm{s}_i = f(\mathrm{x}_i) \tag{3.11}$$

The second function obtains the set of 2D point coordinates $y_{i,c}$ at a point in time i for camera c according to a reprojection function particular to the camera parameters h_c. The expected measurement error is included as a variable parameter $v_{i,c}$. The formulation of this function is shown below.

$$\mathrm{y}_{i,c} = h_c(\mathrm{s}_{i,c}) + \mathrm{v}_{i,c} \tag{3.12}$$

3.3.3 Pairwise Refinement

To expand upon the results presented in the original cheetah pose estimation paper, the trajectory optimisation estimates were refined by the addition *pairwise predictions*. In

place of simply applying neural network-placed estimates to the measurement constraint portion of the algorithm, additional 2D point estimates were obtained through pairwise statistics collection. These additional estimates were added to the trajectory optimisation through the same method as the ordinary estimates described above.

The creators of DeepLabCut [1] provided an alpha version of the algorithm that included functionality for the collection of pairwise stats from training data. These stats are then able to be used for the obtaining of pairwise predictions on images in question. This functionality was drawn from the technique used in the original DeeperCut paper [20]. Examples of the outputs of pairwise stats collection and pairwise prediction will be shown in later chapters.

3.3.4 Cost Function

The original M.Sc. thesis [11] found that a redescending cost function was best at eliminating outliers from the data. The cost function may be defined piecewise through the use of three threshold parameters: a, b and c. Each of these three parameters were chosen to reduce the effect of outliers as much as possible depending on the type of measurement to be considered: ordinary neural network labels or pairwise predictions. This resulted in two separate cost functions. Both cost functions were formulated as seen below.

$$C(e) = \begin{cases} \frac{e^2}{2} & \text{for } |e| < a \\ a|e| - \frac{a^2}{2} & \text{for } a \leq |e| < b \\ ab - \frac{a^2}{2} + \frac{a(c-b)^2}{2}(1 - (\frac{c-|e|}{c-b})^2) & \text{for } b \leq |e| < c \\ ab - \frac{a^2}{2} + \frac{a(c-b)^2}{2} & \text{for } c \leq |e| \end{cases} \tag{3.13}$$

To account for the larger standard deviation inherent in the pairwise point predictions, values of $a = 3$, $b = 5$, and $c = 15$ were chosen for the pairwise measurement cost function. In accordance with the findings of the original paper, an identical cost function with $a = 3$, $b = 10$, and $c = 20$ was chosen for the neural network labels. Both cost functions are shown in Figure 3.4.

3.3.5 Implementation

As mentioned, the trajectory optimisation problem was built in Pyomo. IPOPT [26] was used to solve the trajectory optimisation problem. IPOPT is an interior point optimiser that implements several basic yet effective linear solvers by default. For all trajectory optimisations performed in this paper, the default MUMPS linear solver was used.

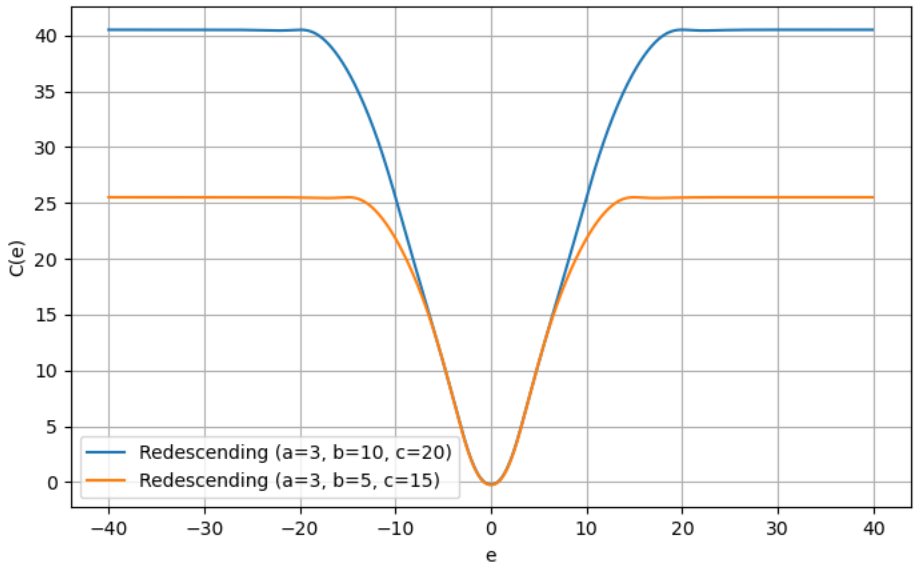

Figure 3.4: Graph showing the redescending cost functions used for both the pairwise predictions and neural network estimations

With the MUMPS linear solver, the vast majority of trajectory optimisations built by this program will take under 5 minutes to converge. Generally, the problem reaches an optimal solution within a tolerance of 0.1 in under 500 iterations. Depending on the machine used, the solutions may converge in as little as 2 minutes. On a Ryzen 5 1600 AF CPU with a base clock of 3.2 GHz and 6 cores/12 threads, even the longer optimisations take under 10 minutes to solve.

If quicker convergences are required for a particular application of these optimisations, other linear solvers such as MA57 or MA86 may be more effective due to their greater use of parallelisation.

3.4 Validation

Validation of this project's outputs was performed in two phases: qualitative analysis and quantitative analysis. Although quantitative analysis alone may seem the natural choice for optimisation problems such as these, the nature of this project's objective calls for a human eye to verify certain results. Specifically, the trajectory optimisation is targeted in large part at minimising unnatural and physically impossible results that a reasonable human would quickly eliminate from their results if given the opportunity. The visual impressions of the program's output are therefore important for validation purposes.

3.4.1 Dataset

The majority of the validation for this project will be performed with a cheetah subject, and will make use of the same dataset used for the original M.Sc. thesis that focused on cheetahs. There are a few main reasons for this:

1. The cheetah is an animal with a fairly complex and flexible morphology. Being a highly agile quadruped with a tail often used during high-speed maneuvers, the animal provides a suitable challenge for our algorithm.

2. If we are able to approach the performance obtained in the original paper focusing on cheetahs with a skeleton built by our program, we can be satisfied that we have built an efficient method.

3. We have access to a large dataset presented in the exact format needed for our program. This dataset has been used widely by the UCT Mechatronics Lab for various experiments.

4. Theoretically, the exact animal subject chosen does not matter. If we are able to build a suitable skeleton ourselves and fit it to an optimal trajectory, it is a simple matter to extrapolate this process to any conceivable species so long as a suitable skeleton is constructed.

The dataset consists of 8582 labelled images of cheetahs. The cheetah was captured from 6 different view angles by 6 different cameras. An example of a cheetah pose viewed from each of the 6 cameras is shown in Figure 3.5.

An accurate CNN was required to obtain 2D pose estimates to within a reasonable degree of accuracy for use by the program. This project made use of the network trained for the original M.Sc. thesis. The network was built using DeepLabCut and trained on 8522 hand-labelled images of cheetahs. A training fraction of 0.95 was used; this means that a proportion of 0.95 of the training images (8096) was used for training and a further 0.05 proportion of the images (the remaining 426 images) was used for testing data. The performance of the network is summarised in Table 3.1.

Set	RMSE	St. Dev
Training	11	59.88
Testing	15.5	68.82

Table 3.1: Training and testing statistics (in pixels) for the cheetah pose estimation CNN used in this project

The cheetah pose viewed in Figure 3.5 is an example of one of the ground truth data poses used to evaluate the trajectory optimisation. To obtain ground truth data, selected

poses were hand-labelled from each of the 6 camera views. Given that these 2D labels are placed by a human, we assume the error in these frames to be negligible. Thus, a basic triangulation was performed on each of the ground truth poses to obtain ground truth 3D data. This data was used to evaluate the performance of the program.

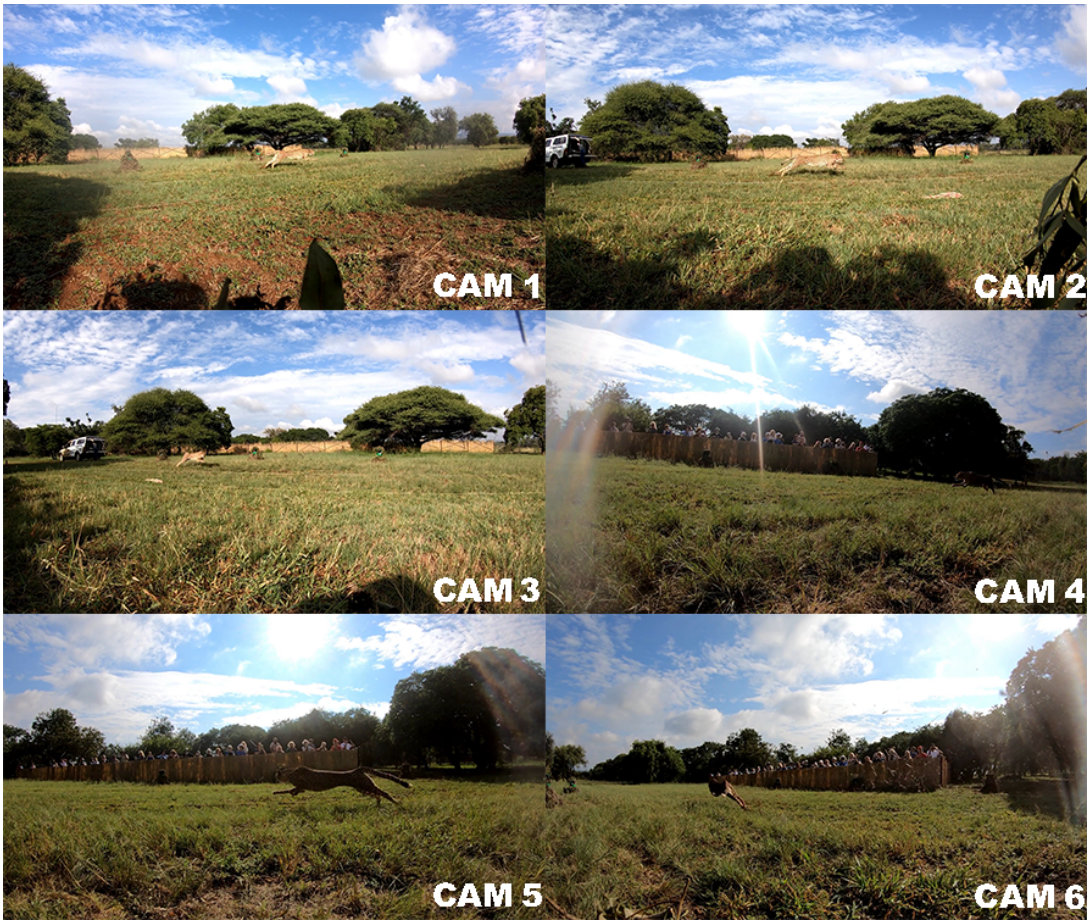

Figure 3.5: An example of a cheetah pose included in the ground truth dataset viewed from all 6 camera angles.

3.4.2 Testing Procedure

To demonstrate the effectiveness of the trajectory optimisation, we compared the results with ground truth data obtained via the method described above. To provide a comparison with existing methods and to show the performance gains of our program relative to them, we performed a triangulation on the neural network labels to obtain 3D pose estimations throughout the trajectory.

Two main methods were used to evaluate performance quantitatively. Both involve the use of the Root Mean Square Error (RMSE) metric. The RMSE is obtained simply by obtaining the difference between a predicted point and a ground truth equivalent point and applying the formula shown below.

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N}(x_{predict} - x_{actual})^2}{N}} \tag{3.14}$$

The first main method used to evaluate performance was the 3D RMSE. This simply involves finding the RMSE between the predicted 3D points and the triangulated 3D ground truth data points. The second main method examines the performance of each algorithm when reprojected back to each of the 2D image planes. The 3D pose estimates are reprojected back to 2D via a reprojection function specific to each camera, where they are then compared with the 2D ground truth labels for each point. This method provides a more complete comparison of the performance, and may be examined separately for each camera.

Additionally, an error metric known as the Mean Absolute Error (MAE) was used for situations where the RMSE may have been skewed disproportionately by outliers. The MAE as a metric provides an advantage over the RMSE for assessing neural network performance as it is more robust to outliers [27]. The MAE, as its name suggests, is calculated as shown below.

$$\text{MAE} = \frac{\sum_{i=1}^{N} |(x_{predict} - x_{actual})|}{N} \tag{3.15}$$

Chapter 4

Results

This chapter will present the results obtained during the course of this project using the aforementioned testing procedure. The chapter is split into two main sections: qualitative and quantitative results. The qualitative results section will provide diagrams and images summarising the visual outputs of the program, while the qualitative results section will provide numerical summaries of the program's performance.

4.1 Qualitative Results

It is important to evaluate the visual outputs of the trajectory optimisation algorithm as well as those of the more rudimentary triangulation algorithm that many existing 3D pose estimation methods use. This section will present our findings in this category. The results will discussed in the following chapter.

4.1.1 3D Motion Tracking Comparison

Reconstructed 3D cheetah skeletons for the ground truth data, basic triangulation, and trajectory optimisation output are shown below in Figure 4.1. The data represents the cheetah at one instant in time viewed by 6 cameras, with all 6 views being labelled and triangulated for ground truth data. It should be noted that, even though real skeletons are rigid, the reconstructed ground truth skeleton is slightly more flexible than the user-defined skeleton. This is in large part due to variance in the human labels for each cheetah image; a human tends to label where they perceive the joint to be at the angle from which they are viewing the subject. The ground truth is therefore prone to slight inaccuracies that lead to temporal inconsistencies. However, these inconsistencies are unlikely to affect the overall quality of the analysis when compared to the drastic mislabels presented by the sparse bundle adjustment, for example.

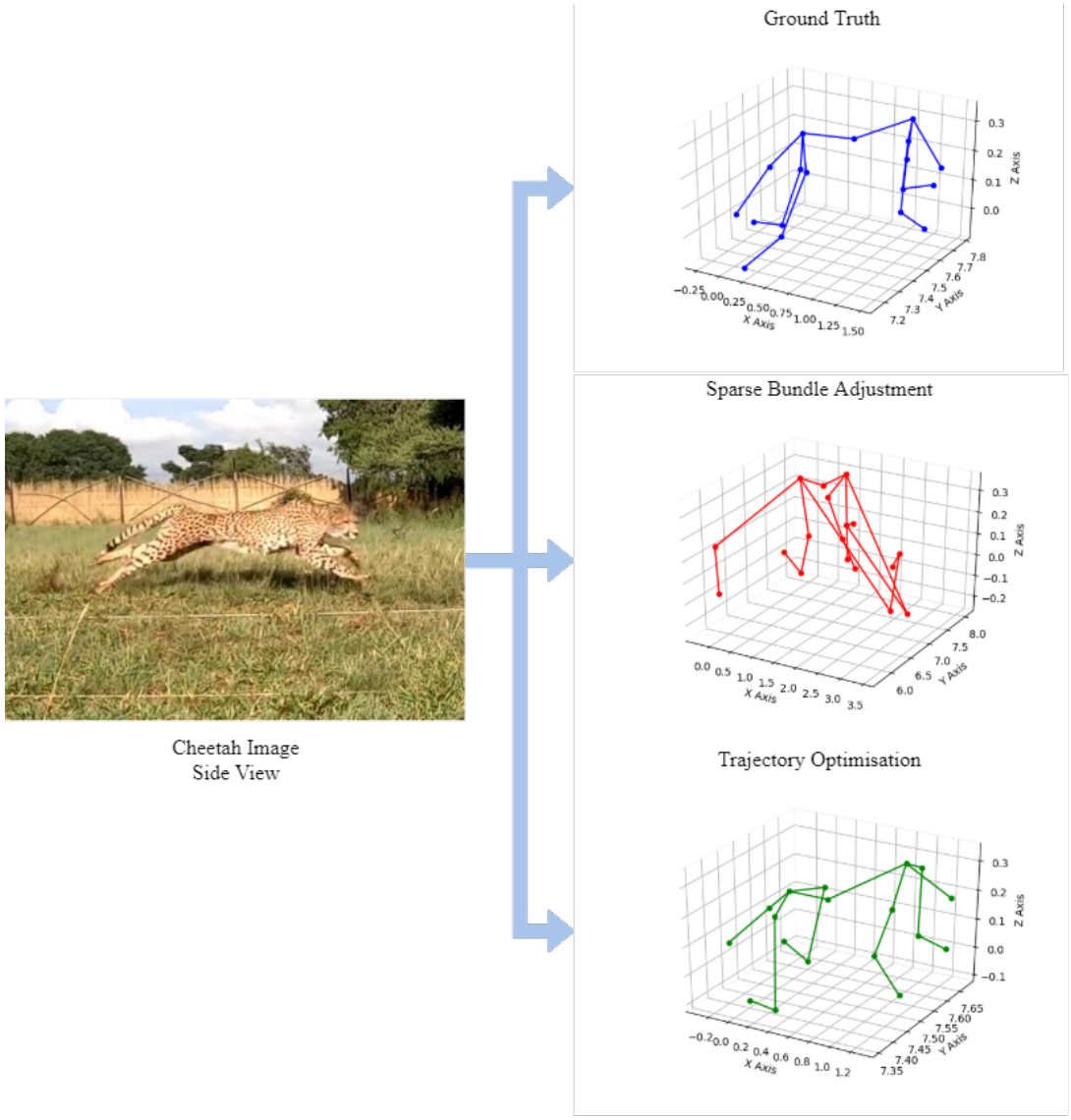

Figure 4.1: A comparison of the trajectory optimisation vs ground truth pose and SBA from a side view

To visualise the cheetah over an extended period of time rather than a single instant, a point cloud was obtained for each of the three sets of data displayed above. Once again, these sets are the ground truth points, SBA, and trajectory optimisation output. The data was viewed from both the side and top to provide a complete picture of the temporal consistency of each of the three sets of results. The plotted 3D point clouds are shown in Figure 4.2.

4.1.2 2D Reprojections

To provide a visual comparison of the accuracy of the different methods when seen from the viewpoint of each of the cameras, the 3D pose estimations from each method were reprojected back to each of the 6 image planes to yield reprojected 2D subject pose

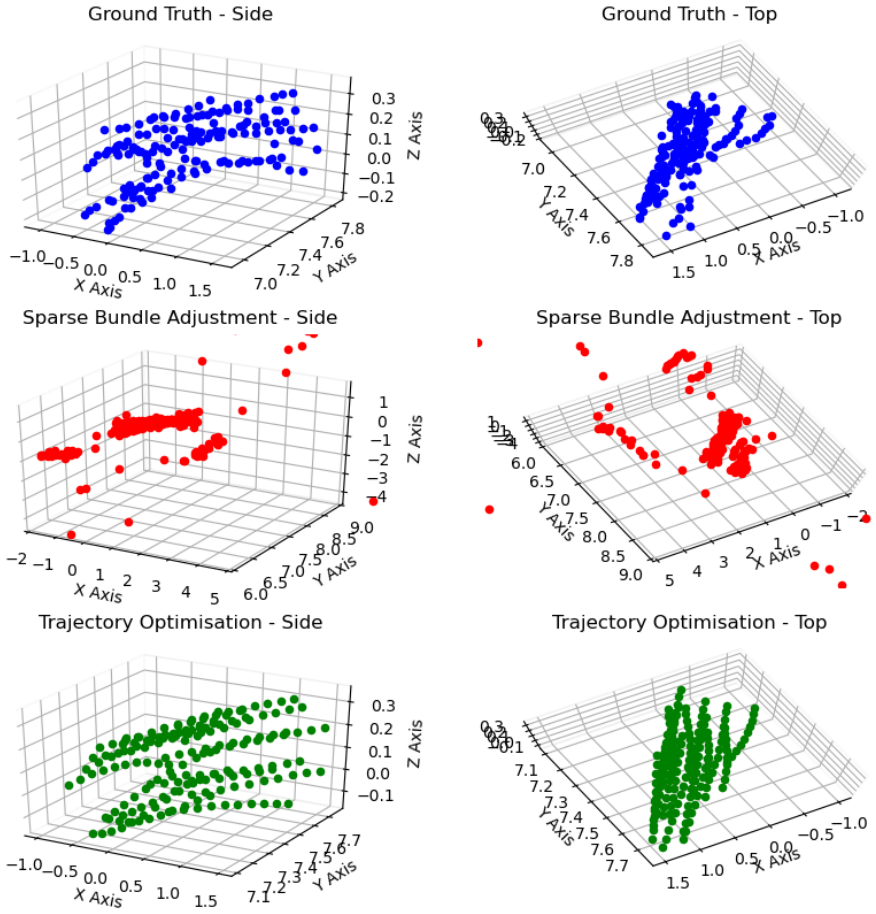

Figure 4.2: Side and top views of the point clouds plotted for the ground truth, SBA, and trajectory optimisation data

estimations. The results of these reprojections for the first 3 camera views of one of the ground truth data poses are shown below in Figure 4.3. As before, SBA is shown in red, ground truth in blue, and trajectory optimisation in green.

4.1.3 Pairwise Predictions

Pairwise predictions were collected and stored for each image in the training and testing datasets ($n = 8582$ images). The pairwise loss statistic on this relatively large dataset converged quickly - relative gains slowed to zero after 50000 iterations which took around 2 hours of real time. To visualise the vector predictions from a base body part to a closely associated body part, the predictions from each ankle of the cheetah to each paw of the cheetah for two frames are shown in Figure 4.4 and Figure 4.5

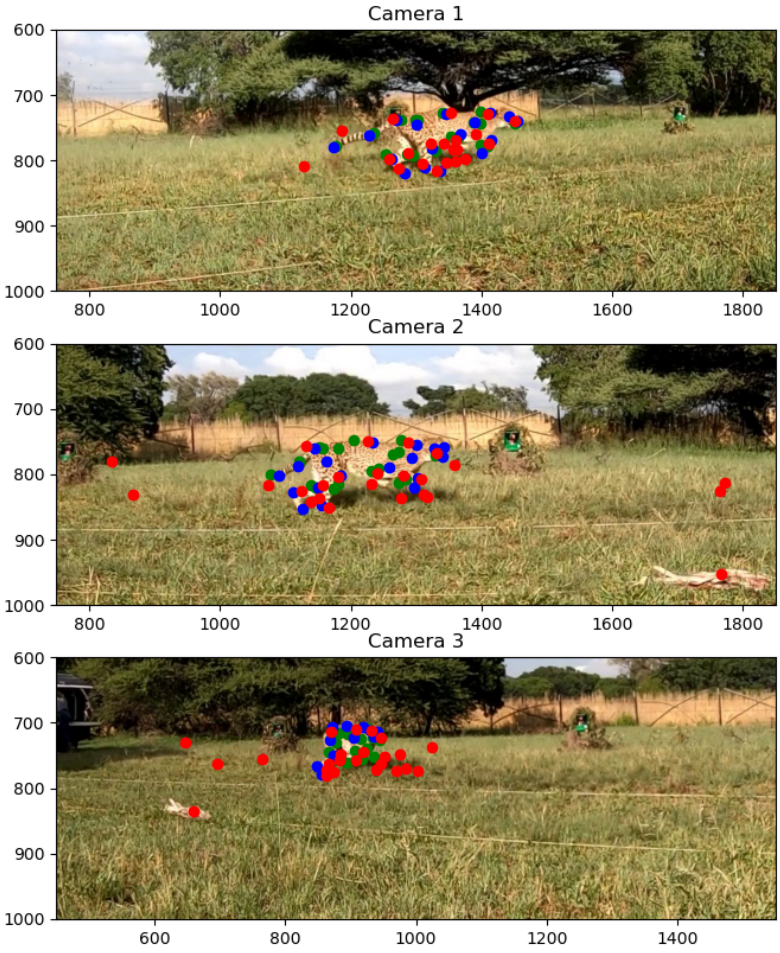

Figure 4.3: Reprojected 2D pose estimations for the SBA and trajectory optimisation for the first 3 camera views

The pairwise predictions for each body part from its closest neighbouring body part were used as extra measurements for the trajectory optimisation. Example 3D reconstructions for the trajectory optimisations including the pairwise measurements and those not including them are shown in Figure 4.6 and Figure 4.7.

4.2 Quantitative Results

This section will provide quantitative measures for the performance of the algorithm. Numerical estimations of the accuracy of the algorithm are important to consider as visual results alone do not provide the specificity that numbers do. As such, this section will provide several numerical measures of performance using the above described methods.

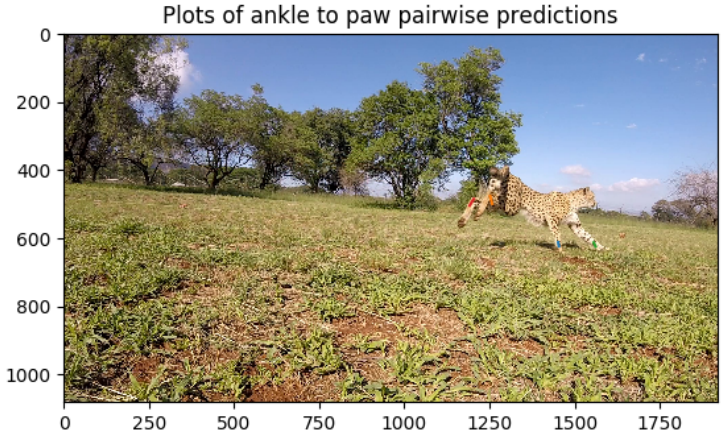

Figure 4.4: Plotted vector predictions for pairwise inference of each paw from its relative ankle (camera 1)

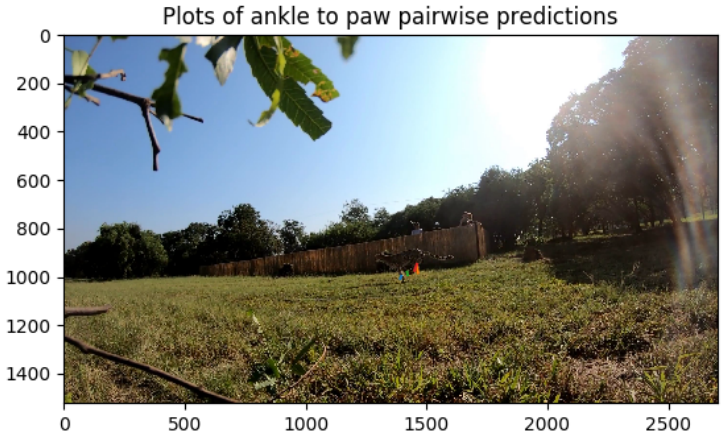

Figure 4.5: Plotted vector predictions for pairwise inference of each paw from its relative ankle (camera 6)

4.2.1 3D Pose Estimation

The 3D pose estimation error in metres was obtained by simply calculating the mean of the x, y, and z differences when compared with the ground truth data. A histogram of the 3D pose error in metres for the sparse bundle adjustment is shown below in Figure 4.8.

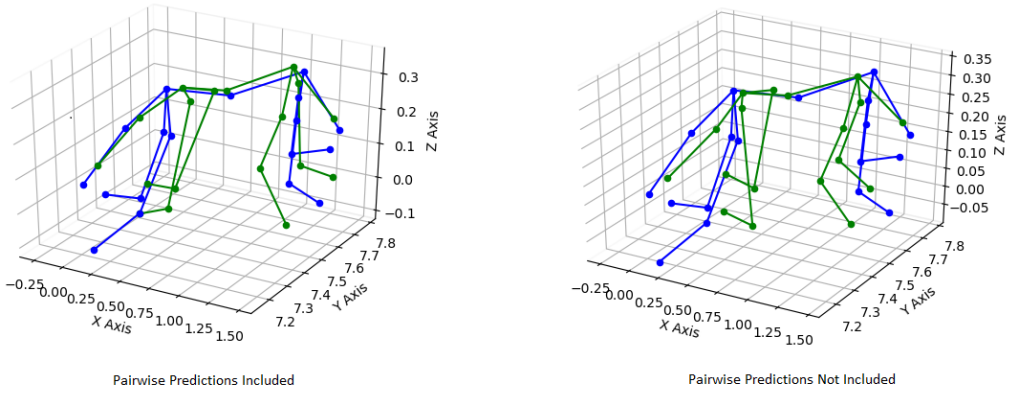

Figure 4.6: Reconstructed 3D skeletons (green) vs. ground truth (blue) for results obtained with and without pairwise predictions

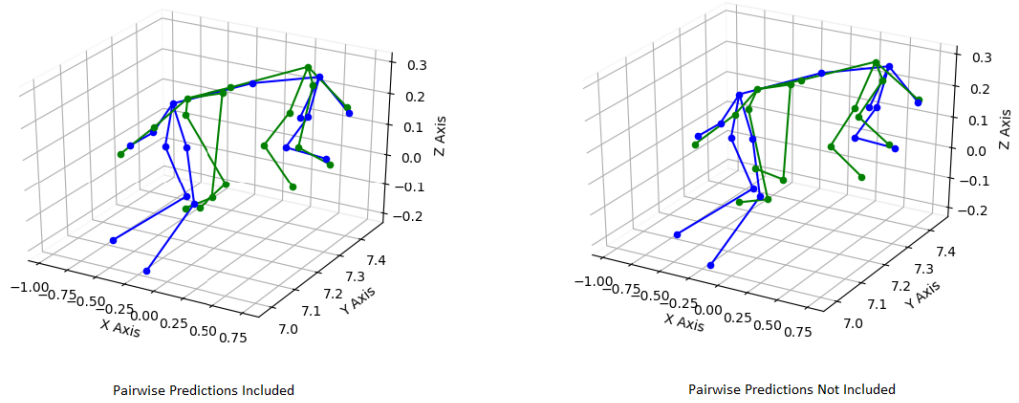

Figure 4.7: Reconstructed 3D skeletons (green) vs. ground truth (blue) for a second set of results obtained with and without pairwise predictions

Additionally, a histogram of the 3D pose error for the trajectory optimisation is shown below in Figure 4.9. Note the higher grouping around an error of 0 metres and the lower amount of outliers.

To directly compare the two methods, the RMSE and standard deviation of both the SBA and trajectory optimisation are shown in Table 4.1. As expected, the SBA method returns both a higher RMSE and standard deviation when compared with the trajectory optimisation.

Method	RMSE	St. Dev
SBA	0.612	0.611
Traj. Opt.	0.213	0.155

Table 4.1: RMSE and standard deviation in metres for each of the 3D pose estimation methods

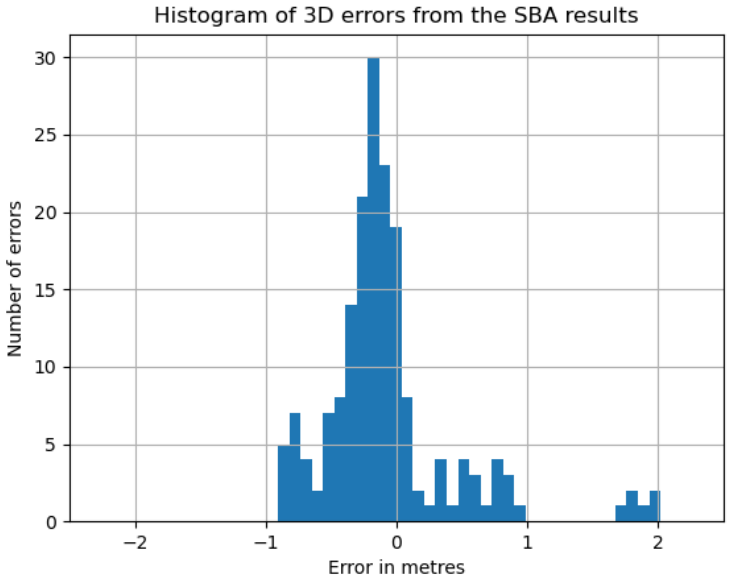

Figure 4.8: A histogram of the 3D pose error in metres for the SBA method ($n = 180$ points)

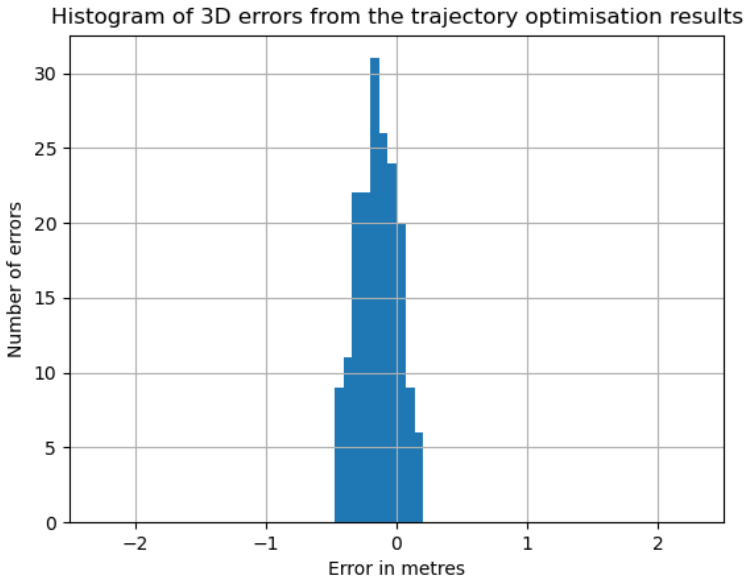

Figure 4.9: A histogram of the 3D pose error in metres for the trajectory optimisation ($n = 180$ points)

4.2.2 2D Reprojections

In addition to a 3D error metric measured in metres, it is pertinent to consider the error of each method in pixels when reprojected to the image plane. To this end, a histogram of the 2D reprojection errors for camera 3 for the SBA method is shown in Figure 4.10.

To compare the SBA's performance with that of the trajectory optimisation, a histogram

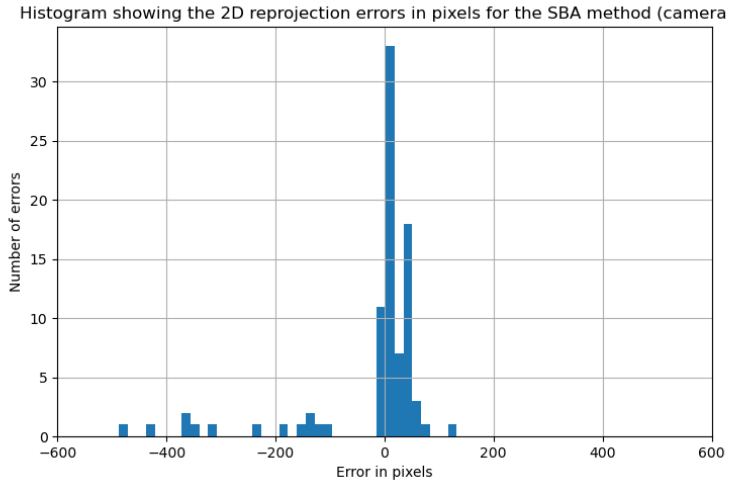

Figure 4.10: A histogram of the 2D reprojection error in pixels for the SBA ($n = 180$ points)

of the reprojection errors for the same camera (camera 3) is shown in Figure 4.11. Note the distinct lack of significant outliers and the higher precision, with most errors grouping closely around the 0 pixel error mark.

We may also infer the performance of each method in relation to each camera angle by calculating the standard deviation and RMSE for each method when reprojected to each image plane. These findings are summarised in Table 4.2

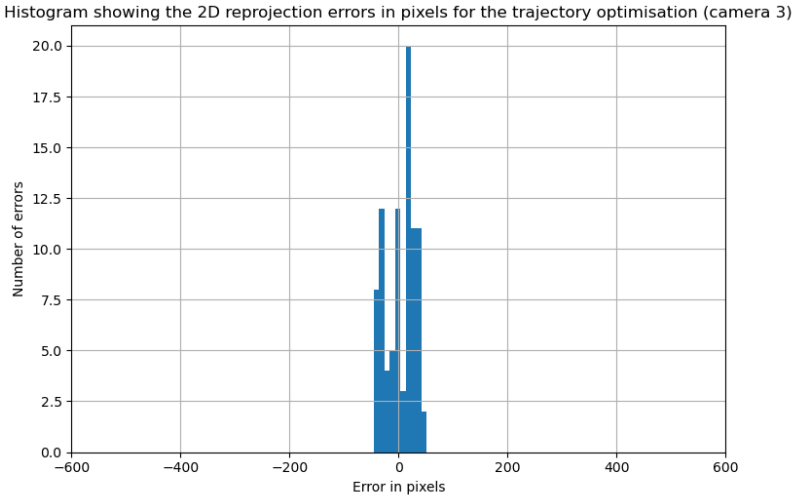

Figure 4.11: A histogram of the 2D reprojection error in pixels for the trajectory optimisation ($n = 180$ points)

Method	Metric	CAM1	CAM2	CAM3	CAM4	CAM5	CAM6
SBA	RMSE	97.89	145.31	167.15	141.23	291.21	146.82
	Std. Dev.	97.04	145.31	166.93	125.17	285.86	114.55
Traj. Opt.	RMSE	62.84	72.03	26.43	77.01	181.31	56.91
	Std. Dev	62.75	71.82	26.13	76.99	178.12	34.49

Table 4.2: RMSE and standard deviation in pixels for the 2D reprojection to each image plane for the SBA and trajectory optimisation

4.2.3 Pairwise Predictions

To provide an idea of the accuracy for typical pairwise predictions from their closest neighbour, histograms for example pairs are shown below. Figure 4.12 shows the errors of the tail tip predicted from the middle of the tail, and Figure 4.13 shows the errors of the program in predicting the right rear ankle from the right rear knee.

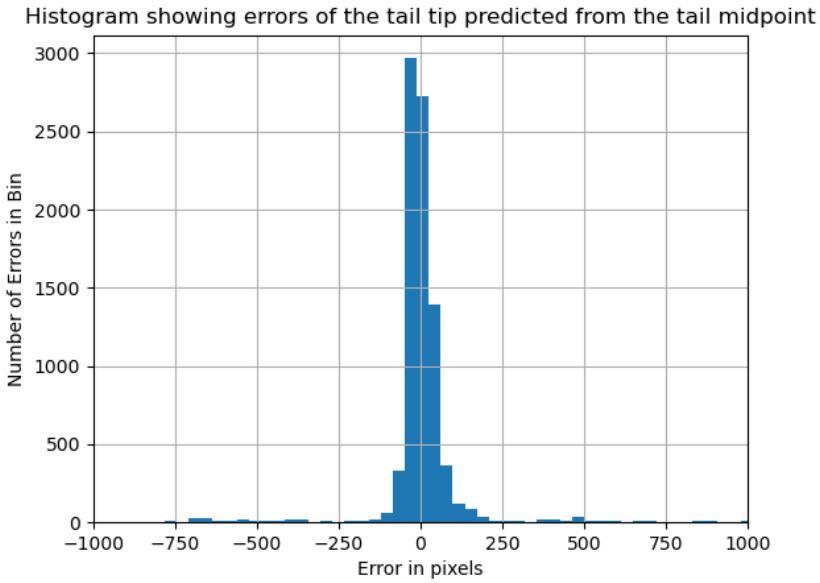

Figure 4.12: A histogram of the errors in pixels of the tail tip predicted from the tail midpoint ($n = 8582$ images)

Additionally, the Mean Absolute Error (MAE) and standard deviations of the tail tip predicted from the tail midpoint and the right rear ankle predicted from the knee are shown in Table 4.3. These errors were measured in pixels for $n = 8582$ 2D training and testing images.

To compare the performance of the trajectory optimisation without pairwise predictions numerically with that of the trajectory optimisation including pairwise predictions, the 3D errors in metres were computed across $n = 90$ ground truth points. The results of this computation are illustrated in the histograms shown in Figure 4.14 and Figure 4.15.

For further numerical analysis of the comparison between including the pairwise points

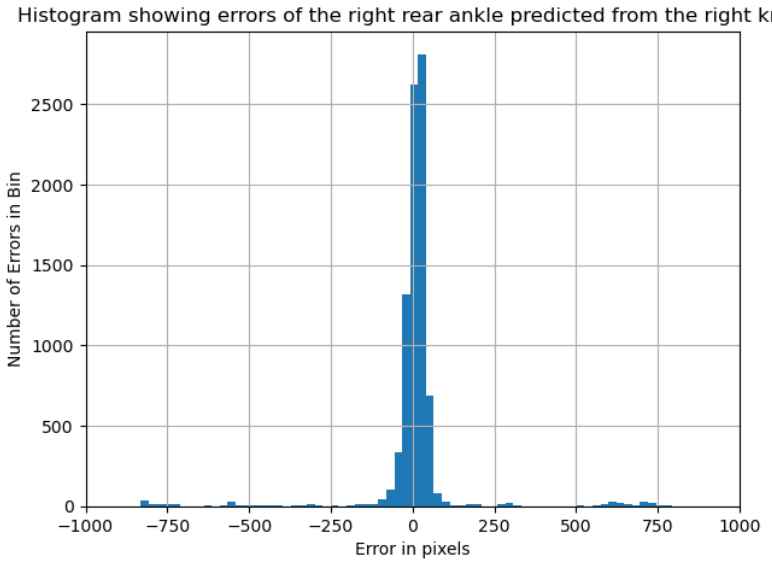

Figure 4.13: A histogram of the errors in pixels of the right ankle predicted from the right knee ($n = 8582$ images)

Pair	MAE	Std. Dev.
Tail mid to tip	56.97	145.90
Right knee to ankle	51.28	138.18

Table 4.3: MAE and standard deviation in pixels for two pairs of body parts with pairwise prediction

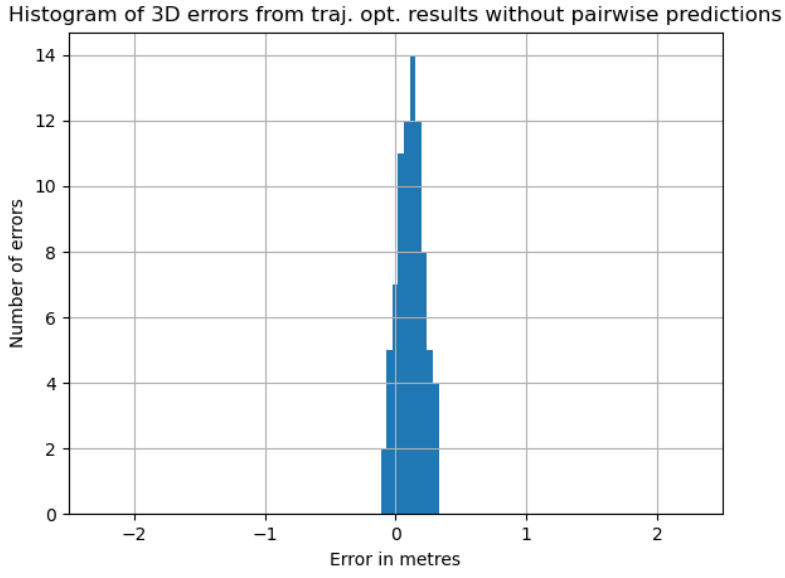

Figure 4.14: A histogram of the 3D errors in metres of the traj. opt. performed without pairwise predictions ($n = 90$ points)

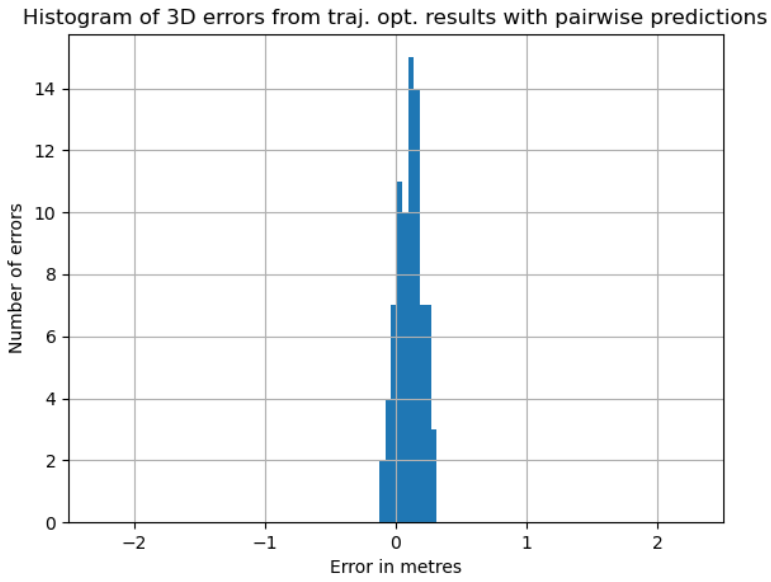

Figure 4.15: A histogram of the 3D errors in metres of the traj. opt. performed with pairwise predictions ($n = 90$ points)

and not including them, a table summarising the 3D RMSE in metres and standard deviations of both sets of results is shown in Table 4.4.

Pairwise	RMSE	Std. Dev.
Yes	0.1457	0.0960
No	0.1549	0.1037

Table 4.4: RMSE and standard deviation in metres for the results obtained with and without pairwise predicted points ($n = 90$ points)

4.2.4 State Tracking

In further analysis of the temporal consistency of the trajectory optimisation, it is useful to view the estimated states over time. The states should appear smooth from frame to frame, and should not display any sudden "jumps" characteristic of outliers. A graph of the x, y, and z positions over time is shown in Figure 4.16. Note the smoothness of each of the three pose parameters, and the relatively constant trajectory of the x and y coordinates as the cheetah moves. The z parameter shows some slight dipping at regular intervals; this is due to the cheetah's "bounding" run style at high speeds.

The remaining states comprise of angles determining the configuration of each rigid link of the skeleton. To examine the state tracking of the cheetah's tail throughout a 100 frame trajectory, the angles ϕ, θ, ψ denoting the configuration of the middle of the tail are plotted in Figure 4.17.

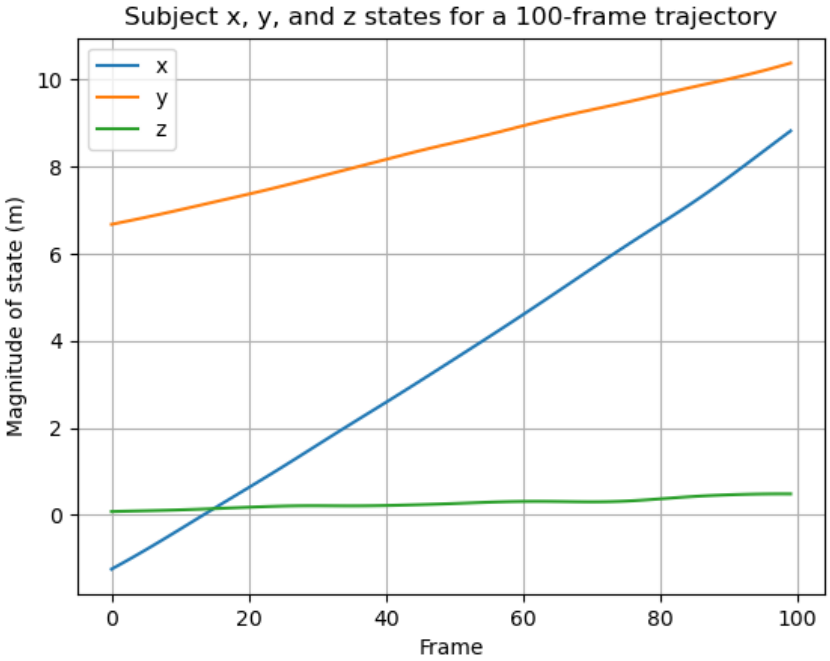

Figure 4.16: X, y, and z estimated states plotted for each frame in an analysed trajectory ($n = 100$ frames)

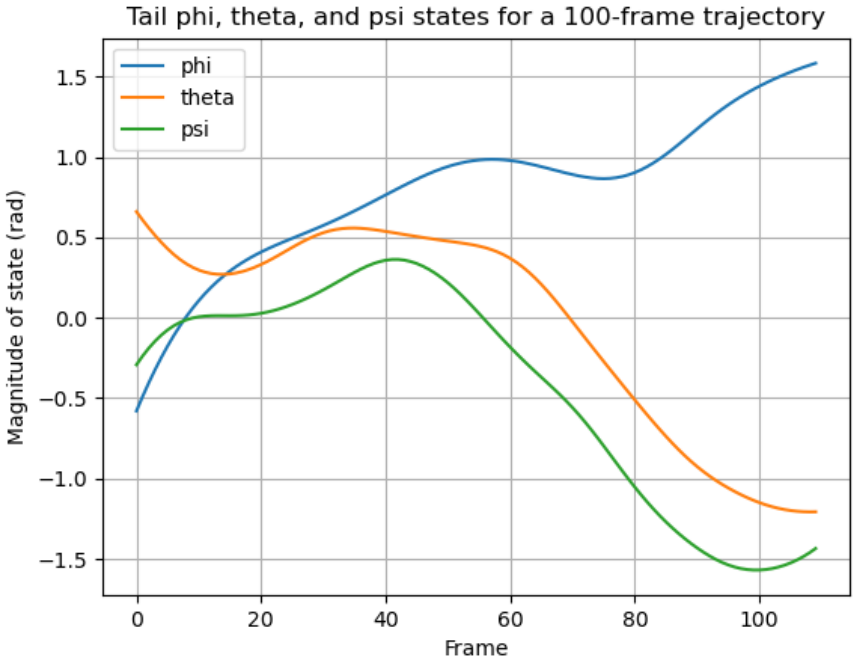

Figure 4.17: Mid-tail phi, theta, and psi estimated states plotted for each frame in an analysed trajectory ($n = 100$ frames)

Chapter 5

Discussion

This section will summarise the findings presented in the previous results section and discuss the relevance of these results in terms of the original problem statement and the current state of research in the field.

5.1 Trajectory Optimisation

5.1.1 Pose Accuracy

The application of the trajectory optimisation for a user-defined skeleton was successful, with drastic improvements in accuracy on a basic sparse bundle adjustment performed on an identical set of 2D pose information. It was found that even with highly curated 2D pose estimations, from images seen by the CNN during training, the inaccuracies present in the machine labels were enough to skew the triangulated 3D reconstructions significantly. The use of the highly constrained trajectory optimisation eliminated many of these outliers. Visually, the results obtained from the trajectory optimisation are far smoother than those obtained from an SBA.

When reprojected back to the image planes, the high precision of the trajectory optimisation algorithm in comparison with the SBA is clear. Virtually no outliers exist in the 2D reprojections of the trajectory optimisation results, with the vast majority of the reprojected labels remaining within the bounds of the cheetah's anatomy. The SBA, by contrast, shows a handful of reprojected points in each frame that are a large distance away from the body of the cheetah. The labels that are accurate remain comparable in accuracy to the trajectory optimisation; however, the SBA's overall performance is lacking. Note that there is a distinction between precision and accuracy; precision refers to a tight grouping of results with low variance, whilst accuracy refers to the average performance of a set of results regardless of their distribution. It is possible for a set of results to be precise but

not accurate, or accurate with low precision.

The SBA is an example of a method that produces low precision, high variance results. While they may not be as inaccurate on average as one may expect, the high variance present in the results leads to a visually confusing reconstruction. The trajectory optimisation leads to results that have both high accuracy and high precision. This may be attributed to the use of temporal information inherent in the trajectory optimisation; while the SBA simply considers a single frame at a time, the trajectory optimisation has the advantage of extra information from the past and future inputs.

5.1.2 Run-Time

The trajectory optimisation algorithm is not intended for "live" or particularly fast processing speeds. The optimisations typically take around $2-5$ minutes to converge for a moderately complex cheetah skeleton for around 100 frames. This algorithm is intended for post-processing performed on 2D pose data some time after the original data has been collected. Thus, it is best suited for close analyses of animal subjects for applications such as research papers, the design of biomimetic robots, the analysis of gait for neurological assessment, and other such purposes. Situations where fast data analysis is required, such as live trajectory optimisations performed by an autonomous robot in motion, may find better results with a system such as an EKF.

5.2 Pairwise Predictions

The introduction of pairwise predictions into the trajectory optimisation algorithm was relatively successful. While performance gains are difficult to see on many of the body parts, the effect becomes more obvious when the extremities are considered. Body parts that are typically more occluded or in more unpredictable positions display the largest performance gains. In these cases, the addition of extra measurements to minimise the effect of errors in the primary set of measurements was an effective method at improving overall accuracy and precision.

The improvements on performance may not seem too impressive when one examines the numbers either. Performance gains of around $0.01m$ are seen when the RMSE is considered, with a comparable decrease in the standard deviation. This number is deceptively meaningful, however; these performance gains are noticeable upon a visual inspection, and they were obtained using the pairwise predictions from only one other body part. More complex skeletons with higher numbers of body parts to track will become both increasingly difficult to estimate by the trajectory optimisation alone, and

increasingly plentiful in regard to close body parts from which to obtain pairwise predictions. These small performance increases may pave the way to larger and more noticeable performance gains for other species.

5.3 Potential Applications

This project yielded promising results for the applications of 3D motion tracking in the wild. Since a GUI was developed for the input of the user's desired skeleton, this method need not only be applied to animals. If a suitable DeepLabCut network (or other software solution for the 2D pose estimation of user-defined species) is trained, the user is theoretically able to perform 3D reconstructions using trajectory optimisations on any subject. The user may desire, for example, to track the motion of an aircraft for which they have multiple recorded videos from differing angles. The temporal smoothing effects of the trajectory optimisation algorithm are easily extendable to robots, projectiles, vehicles, and any other configuration of object for which the user has some cursory geometric knowledge.

Chapter 6

Conclusions

This project has presented a method for the performing of a trajectory optimisation on a user-defined species. A GUI for user input of a predefined skeleton was developed, the output of which is then able to be fitted to a set of 2D pose estimations corresponding to two or more camera views of the subject. The design process for this algorithm, as well as a description of its implementation and a series of validation procedures, is presented in this document.

It was found that the trajectory optimisation provided significant performance improvements over conventional 3D reconstruction techniques. These include sparse bundle adjustments run on each specific scene to optimise the intrinsic and extrinsic camera parameters as well as the 3D projections for a given set of 2D pose data. For each set of points evaluated, the trajectory optimisation generated more temporally consistent results and vastly greater visual impressions than the SBA.

This project also incorporated pairwise predictions into the trajectory optimisation algorithm and saw further increases in performance. These gains are primarily useful (and effective) for extremities that have particularly high variance or constant occlusion throughout a motion.

Chapter 7

Recommendations

7.1 Further Testing of the Trajectory Optimisation

Due to the limited amount of data concerning uncommon species, this project was only evaluated on cheetah subjects. For future researchers, it may be of use to evaluate the program's performance on a variety of different species in different conditions. While the situation evaluated in this project is a relatively complex pose estimation problem, the wider applicability of this project has yet to be specifically evaluated.

7.2 Expansion of Pairwise Functionality

This project made fairly straightforward use of the pairwise predictions obtained from the CNN, with only one closest neighbour body part being used to predict a given body part. Future researchers may find it intriguing to explore the concept of pairwise predictions further. One area that may be improved upon in this regard is the algorithm run-time; the human pose estimation project DeeperCut [20] presented astronomical speed-ups in the inference times of their network when pairwise inference was used to limit the search radius for neighbouring body parts. Potentially, these speed ups may be used to develop a live (or semi-live) version of this algorithm, where trajectory optimisations may be performed in real time on subjects in the wild.

7.3 Parallelisation of the Algorithm

To provide a further boost in processing speed for the trajectory optimisation, the exploration of more efficient linear solvers such as MA86 may be of interest. The higher rates of data parallelisation of these solvers may increase both speed and overall

performance of the trajectory optimisation. Once again, higher speeds may be utilised for live versions of this algorithm.

7.4 Automated Skeleton Building

The process of building a trajectory optimisation to be solved may be further automated through the development of a means to obtain basic subject skeletons in an algorithmic manner. This would involve the application of advanced computer vision principles as well as a highly efficient algorithm to match human accuracy when defining skeleton parameters. Due to the robust nature of the trajectory optimisation procedure developed in this project, small amounts of variance in skeleton parameters - although unnatural - may be relatively insignificant to the final results.

7.5 Optimising the Optimisation

Recent research into various avenues of meta-learning [28] [29] [30] [31] may inform the process of optimising the optimisations inherent in this program. There are various parameters that may be optimised to improve performance significantly when compared to hand-tailored alternatives:

1. The cost function may be made more robust to outliers through gradient descent or intelligent algorithms
2. The trajectory optimisation may be provided with a better initialisation learned through an optimisation
3. The tolerance, constraints, and other hyperparameters of the trajectory optimisation may be iteratively tweaked to provide the best results

Optimised trajectory optimisations may execute far faster, reach significantly better final results, and require fewer computational resources than hand-designed ones.

Bibliography

[1] A. Mathis, P. Mamidanna, K. M. Cury, T. Abe, V. N. Murthy, M. W. Mathis, and M. Bethge, "Deeplabcut: markerless pose estimation of user-defined body parts with deep learning," *Nature neuroscience*, vol. 21, no. 9, pp. 1281–1289, 2018.

[2] P. Rahimian and J. Kearney, "Optimal camera placement for motion capture systems in the presence of dynamic occlusion," pp. 129–138, 11 2015.

[3] C.-H. Chen and D. Ramanan, "3d human pose estimation = 2d pose estimation + matching," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[4] X. Zhou, Q. Huang, X. Sun, X. Xue, and Y. Wei, "Towards 3d human pose estimation in the wild: A weakly-supervised approach," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[5] T. Nath, A. Mathis, A. C. Chen, A. Patel, M. Bethge, and M. W. Mathis, "Using deeplabcut for 3d markerless pose estimation across species and behaviors," *Nature protocols*, vol. 14, no. 7, pp. 2152–2176, 2019.

[6] S. V. and M. Vaidyan, "Ann approach for state estimation of hybrid systems and its experimental validation," *Mathematical Problems in Engineering*, vol. 2015, pp. 1–13, 03 2015.

[7] X. Sun, J. Duan, X. Li, and X. Wang, "State estimation under non-gaussian levy noise: A modified kalman filtering method," *Banach Center Publications*, vol. 105, 03 2013.

[8] A. Winkler, F. Farshidian, D. Pardo, M. Neunert, and J. Buchli, "Fast trajectory optimization for legged robots using vertex-based zmp constraints," *IEEE Robotics and Automation Letters*, vol. PP, 05 2017.

[9] S. Shield and A. Patel, "Balancing stability and maneuverability during rapid gait termination in fast biped robots," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4523–4530, IEEE, 2017.

[10] "Boston dynamics robots," 2020.

[11] L. J. Clark, "Markerless 3d motion capture of cheetahs in the wild," 2020.

[12] Z. Cao, G. Hidalgo, T. Simon, S. Wei, and Y. Sheikh, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," *CoRR*, vol. abs/1812.08008, 2018.

[13] M. Patacchiola and A. Cangelosi, "Head pose estimation in the wild using convolutional neural networks and adaptive gradient methods," *Pattern Recognition*, vol. 71, pp. 132 – 143, 2017.

[14] L. Ge, H. Liang, J. Yuan, and D. Thalmann, "3d convolutional neural networks for efficient and robust hand pose estimation from single depth images," July 2017.

[15] J. M. Graving, D. Chae, H. Naik, L. Li, B. Koger, B. R. Costelloe, and I. D. Couzin, "Deepposekit, a software toolkit for fast and robust animal pose estimation using deep learning," *Elife*, vol. 8, p. e47994, 2019.

[16] T. D. Pereira, D. E. Aldarondo, L. Willmore, M. Kislin, S. S.-H. Wang, M. Murthy, and J. W. Shaevitz, "Fast animal pose estimation using deep neural networks," *Nature methods*, vol. 16, no. 1, pp. 117–125, 2019.

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

[18] 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.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[20] E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, and B. Schiele, "Deepercut: A deeper, stronger, and faster multi-person pose estimation model," in *European Conference on Computer Vision*, pp. 34–50, Springer, 2016.

[21] R. I. Hartley and P. Sturm, "Triangulation," vol. 68, p. 146–157, Nov. 1997.

[22] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment—a modern synthesis," in *International workshop on vision algorithms*, pp. 298–372, Springer, 1999.

[23] Y. Zeng and R. Zhang, "Energy-efficient uav communication with trajectory optimization," *IEEE Transactions on Wireless Communications*, vol. 16, no. 6, pp. 3747–3760, 2017.

[24] W. E. Hart, J.-P. Watson, and D. L. Woodruff, "Pyomo: modeling and solving mathematical programs in python," *Mathematical Programming Computation*, vol. 3, no. 3, pp. 219–260, 2011.

[25] W. E. Hart, C. D. Laird, J.-P. Watson, D. L. Woodruff, G. A. Hackebeil, B. L. Nicholson, and J. D. Siirola, *Pyomo–optimization modeling in python*, vol. 67. Springer Science & Business Media, second ed., 2017.

[26] A. Wchter and L. T. Biegler, "On the implementation of an interior point filter line-search algorithm for large-scale nonlinear programming,"

[27] C. J. Willmott and K. Matsuura, "Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance," *Climate research*, vol. 30, no. 1, pp. 79–82, 2005.

[28] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas, "Learning to learn by gradient descent by gradient descent," in *Advances in neural information processing systems*, pp. 3981–3989, 2016.

[29] S. Thrun and L. Pratt, *Learning to learn*. Springer Science & Business Media, 2012.

[30] K. Li and J. Malik, "Learning to optimize," *arXiv preprint arXiv:1606.01885*, 2016.

[31] H. Sun, X. Chen, Q. Shi, M. Hong, X. Fu, and N. D. Sidiropoulos, "Learning to optimize: Training deep neural networks for interference management," *IEEE Transactions on Signal Processing*, vol. 66, no. 20, pp. 5438–5453, 2018.

Appendix A

Additional Files and Schematics

A.1 GitHub Repository

The GitHub repository containing all code used to obtain the results presented in this report is available at https://github.com/DJoska/FYP.

Appendix B

Addenda

B.1 Ethics Forms

ETHICS APPLICATION FORM

Please Note:
Any person planning to undertake research in the Faculty of Engineering and the Built Environment (EBE) at the University of Cape Town is required to complete this form **before** collecting or analysing data. The objective of submitting this application *prior* to embarking on research is to ensure that the highest ethical standards in research, conducted under the auspices of the EBE Faculty, are met. Please ensure that you have read, and understood the **EBE Ethics in Research Handbook** (available from the UCT EBE, Research Ethics website) prior to completing this application form: http://www.ebe.uct.ac.za/ebe/research/ethics1

APPLICANT'S DETAILS		
Name of principal researcher, student or external applicant		Daniel Joska
Department		Electrical Engineering
Preferred email address of applicant:		JSKDAN001@myuct.ac.za
If Student	Your Degree: e.g., MSc, PhD, etc.	B.Sc (Eng)
	Credit Value of Research: e.g., 60/120/180/360 etc.	40
	Name of Supervisor (if supervised):	Dr Amir Patel
If this is a researchcontract, indicate the source of funding/sponsorship		N/A
Project Title		Markerless Motion Capture in the Wild using Optimisation

I hereby undertake to carry out my research in such a way that:

- there is no apparent legal objection to the nature or the method of research; and
- the research will not compromise staff or students or the other responsibilities of the University;
- the stated objective will be achieved, and the findings will have a high degree of validity;
- limitations and alternative interpretations will be considered;
- the findings could be subject to peer review and publicly available; and
- I will comply with the conventions of copyright and avoid any practice that would constitute plagiarism.

APPLICATION BY	Full name	Signature	Date
Principal Researcher/ Student/External applicant	Daniel Joska		16/08/2020
SUPPORTED BY	Full name	Signature	Date
Supervisor (where applicable)	Amir Patel		17/08/2020

APPROVED BY	Full name	Signature	Date
HOD (or delegated nominee) Final authority for all applicants who have answered NO to all questions in Section 1; and for all Undergraduate research (Including Honours).	A/Prof F Nicolls pp J Buxey	 Dept. Manager: Elec Eng Authorised to sign obo HOD	28.8.2020
Chair: Faculty EIR Committee For applicants other than undergraduate students who have answered YES to any of the questions in Section 1.			

18/08/2020

Research Proposal

Daniel Joska
JSKDAN001

Marker-less Motion Capture of Animals in the Wild Using Optimisation

---

1 Introduction

The analysis of specific filmed motions of various species is necessary to many forms of research, including research into animal psychology, biomechanics, and bio-inspired robotics. While software packages exist for popular research subjects such as humans, there is a current lack of effective solutions for the three-dimensional motion capture of user-defined species. Recently, work by the UCT Mechatronics Lab has yielded impressive results for the analysis of cheetah motions using a combination of sparse bundle adjustment (SBA) and trajectory optimisation techniques for data smoothing. This research is predicated upon the previously mentioned results for cheetahs and will extend this methodology across user-defined species.

2 Problem Statement

Most open-source offerings in the category of 3D motion capture make use of triangulation. This involves the processing of 2D estimations of animal joint positions to estimate a 3D position of the animal. This is often an inaccurate technique as slight errors in 2D labels may lead to large mistakes in the 3D pose estimation – with user-defined species, this problem is compounded as highly accurate pose estimation networks are scarce or non-existent.

3 Objectives

Based on the problem statement above, this project aims:

- To provide an open-source software package for the 3D motion capture of user-defined species from user-supplied data
- To make use of previously established sparse bundle adjustment techniques and trajectory optimisation to smooth data and produce reasonable motion reconstructions
- To provide useful and meaningful output data, including (possibly) in-built post-processing and analysis functions

4 Methodology

This project will derive many of its functions from already existing open-source code. The bulk of this project's work will be in generalising existing code for specific species to any user-defined input species. In achieving this, the project will make use of public image datasets depicting a variety of species in motion in order to evaluate its success. The procurement of such datasets will be ethical, and the practices used to generate the datasets by third parties in the first place will be ethically evaluated and, where necessary in the report, described to the reader. Due to the abundance of useful toolboxes, the project shall be programmed in Python. The project will begin by adapting existing code to extend its functionality to generic species. The results will be qualitatively evaluated on public image datasets, and subsequently numerically evaluated. The project will then provide additional post-processing functions which may be of use to future researchers and evaluate the effectiveness of such functions. Finally, the open-source code will be published to GitHub.