WildPose: Long-Range 3D Motion Tracking of Cheetahs in the Wild Using Multi-Sensor Fusion

---

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 Master of Science degree in Mechatronics

August 27, 2023

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.

Signed by candidate

Signature:.....

D. J. Joska

Date: 31/08/2023

Acknowledgments

I would like to extend my heartfelt thanks to all who supported me - financially or otherwise - during the writing of this thesis.

I greatly appreciate my parents and all they have done for me; the list is too long to recount. I am grateful for their undying support.

I extend my thanks to my supervisor, Prof Amir Patel, for his sage guidance and advice throughout my years at UCT. His passion and drive inspire me to this day to persevere and reach higher.

I sincerely thank all those who were involved in the funding of my studies - including the National Research Foundation, Google Research, UCT's internal scholarship board, and the Manuel and Luby Washkansky fund. Without their support, this research would not have been possible.

I appreciate all the assistance from those involved at Anne Van Dyk Cheetah Centre, Cheetah Outreach, and Kgalagadi National Park - every trip to these locations yielded valuable data as well as the insight that made this project possible.

Lastly, I thank all who take the time to read this thesis. I hope that the research contained herein proves insightful and enjoyable to every reader.

Abstract

In many fields of research, it is often desirable to estimate the 3D pose of a subject - human, animal, or otherwise. Methods for obtaining accurate 3D pose data of a subject are broad in their applications; they inform the design of bio-mimetic robots, they aid greatly in bio-mechanical research, and they are used commonly in the study of animal neuroscience. Currently, robust methods for long-range tracking of subjects in the wild are few and far between, given the rarity of specific training data as well as the generally challenging nature of the associated computer vision problems.

This thesis describes the design, implementation, and testing of both a hardware and software component to a method for the 3D motion capture of cheetahs in the wild, dubbed WildPose. The method makes use of multi-sensor fusion including lidar, RGB and IMU sensors to compensate for situations where pure vision-based techniques perform inadequately. To increase robustness, the software design makes use of previously successful trajectory optimisation techniques to yield accurate pose data in adverse conditions that would otherwise be extremely difficult to obtain. The method is extendable to other species with minimal variations.

We demonstrate the method's efficacy through experimental validation on challenging cheetah locomotion datasets collected in the wild, presenting both qualitative and quantitative analyses for varied movements, environments, and lighting conditions. Through the shown effectiveness of these techniques in our specific use case, we aim to prove that the methods used perform admirably even in difficult reconstruction problems. Thereby, we present our findings on cheetahs as a promising blueprint for the 3D motion capture of other species.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

In various fields of research and industry, it is necessary to gather 3D pose data of a subject throughout a given sequence of motion. The applications of this pose data are vast and depend on the specific subject. Bio-inspired robotics systems such as Boston Dynamics' robotic dog Spot use information gleaned from animal pose data to dictate control schemes, trajectories, and overall mechanical design. Various entertainment products such as movies and video games make use of motion capturing technologies to reproduce both human and animal motion in animated form. Pose data in 3D space provides a wealth of invaluable information about the subject's underlying motion.

One area of 3D pose estimation that has large room for improvement is the motion capture of animals in the wild. Outside of the confines of a laboratory, many difficulties in obtaining accurate pose data arise. These include but are not limited to: frequent total and partial occlusion of body parts rendering traditional vision-based techniques ineffective, challenging dynamic backgrounds, long distances between the researcher and the subject, and the inability to place markers on the subject animal,

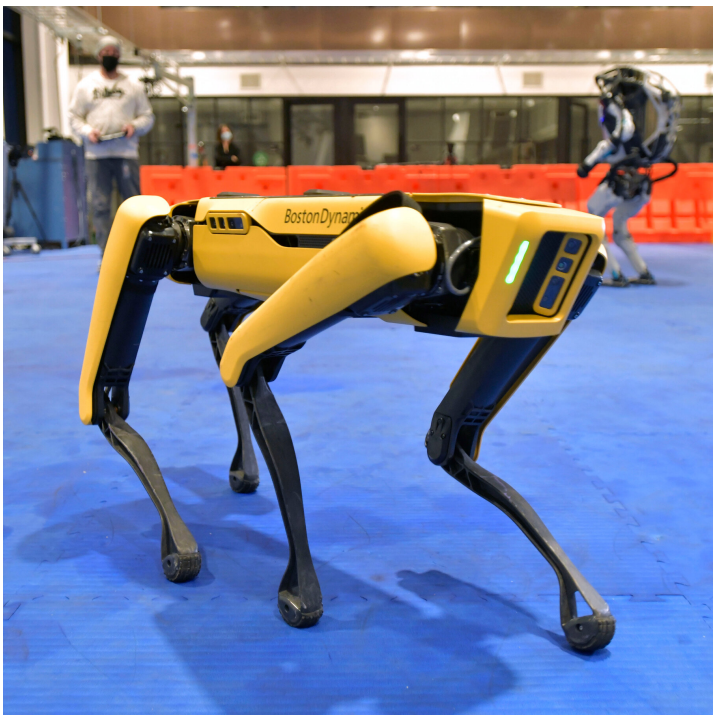

Figure 1.1: An image of Spot, a dog-inspired quadrupedal robot developed by Boston Dynamicsc [1]

eliminating the use of GPS and similar technologies. Previous research has made great progress by utilising stereo camera setups paired with model fitting on top of the acquired data [10] [11]. However, this technique largely relies on static cameras at close distances, vastly reducing the total pool of research data that may be obtained over a given stretch of time in the wild.

Increases in the effectiveness and reductions in the cost of motion capture methods for subjects in the wild would significantly improve the range and quantity of pose data for difficult species.

1.2 Problem Statement

Given the aforementioned gap in motion capture technologies, we may further specify the problem we aim to solve with this study. The main

issue as it pertains to this work is the relative lack of robust motion capture techniques to capture and reconstruct animal locomotion in a wild setting. Existing methods require either contact with the animal as well as compliance in order to place markers, or elaborate and static camera trap setups to surround the subject with traditional RGB cameras for later stereo reconstruction. When neither of these solutions are possible, it becomes far more difficult to create and analyse a suitably sized dataset.

Within the scope of this project, we will focus on the cheetah as a research subject. The main reasons for the selection of cheetahs as a focal point are as follows:

- The cheetah is a particularly interesting research subject and many aspects of its locomotion, such as the use of its tail during high speed manoeuvres, are not well understood within existing literature.
- Cheetahs lie at a crucial intersection of the problems with modern motion capture techniques: they move extremely fast, are usually only observable in the wild at long distances, and tend to be occluded for large portions of their motion due to dust and foliage.
- Existing work has provided a solid basis for the study of cheetah pose reconstruction, yet there still remains much room for improvement.

Given this set of reasons the work presented henceforth will revolve around the study of cheetahs. However, there is no reason the broader techniques outlined here could not be extended to other species. Once these methods are proven on a difficult research problem such as the reconstruction of highly occluded cheetah motion at long distances, their efficacy on simpler problems becomes even more promising. It is our hope that with the designs and methodologies laid out in this work, future studies conducted on uncommon species in the wild will be able to build off of our success.

1.3 Objectives

There are several main objectives of this project. Broadly, these may be defined as follows:

1. The design and implementation of a low-cost data capture rig able to be deployed in the wild and capable of capturing rich data of a subject from long distances.
2. The building of a dataset of cheetah locomotion in the wild, involving the deploying of the data capture rig and recording any further iterations in its design.
3. The design and implementation of a set of software techniques to reconstruct accurate 3D pose data of cheetahs from the dataset.

1.4 Review of Existing Solutions

Before an approach to solving our particular research problem is selected, it is pertinent to review similar studies and examine what tools and technologies were used therein. In doing so, we may extract the parts of these studies relevant to our research and create a refined plan of development.

A 2017 study had success tracking the pose of the tails of captive cheetahs using an attached GPS device, an IMU, and a camera affixed to the animal [12]. The method made use of an Extended Kalman Smoother (EKS) to aid the fusion of sensor data, resulting in an accurate pose over the tracked trajectory. Whilst robust, these methods are only useful for cheetahs bred in captivity, as they must be docile enough to handle while the devices are attached. The method does not carry over to the motion capture of wild cheetahs.

Another study presented a more general animal motion capture technique

making use of DeepLabCut - a toolbox for 2D pose estimation - along with a stereo camera setup [6]. The article touches on the use of a six-camera setup for the motion tracking of cheetahs, requiring no devices be attached to the subject at all. However, the proposed camera setup is static, and surrounds a lure track used to guide the subject along a predefined path for optimal camera coverage. This, too, is not sufficient for our specific problem, as we will be unable to guide the wild cheetahs through a stereo camera setup reliably. The range at which this method is able to capture pose accurately is also unsatisfactory.

Given this brief overview of attempted solutions for similar problems, it is clear that our situation will require a novel approach - or at least, a novel fusion of existing motion capture techniques. A more thorough analysis on the literature relevant to our study is presented in the following section.

1.5 Scope and Limitations

As mentioned, this project will limit the scope of subjects studied to cheetahs alone. Strategies for expanding these methodologies to other species and for facilitating the creation of a general adaptive solution to animal motion capture will be suggested, but not implemented or evaluated within the scope of this work.

This project will not provide in-depth details on the design and implementation of the 2D convolutional neural networks used to provide 2D keypoints for footage in the dataset. Such descriptions are outlined in previous works and will not be verbosely expanded upon here. Based on successes in previous research [10] [11], we will make use of a DeepLabCut [3] neural network for 2D keypoint estimation.

The bulk of the dataset used in this work was obtained during two expeditions undertaken by the author and co-workers. It does not include footage of cheetahs or other animals from geographical locations outside South Africa or from differing climates and ecosystems. The dataset may

be expanded in future work.

1.6 Plan of Development

This study will follow a general structure as outlined below. These main blocks of work are referred to within multiple sections of this document. It should be noted that this plan of development follows the sequence that the real world work took; however, this document may not.

1.6.1 Project Milestones

Data Capture Rig Design

The first milestone of this project is the hardware and software design of the WildPose data collection rig. Broadly, the rig will make use of multi-sensor fusion to gather synchronised raw data of the subject from multiple different sensors, including lidar, camera, and IMU devices.

Data Gathering and Dataset Creation

The next major undertaking of the project entails field work. Once the data capture rig has been designed and built, it will then be used during several expeditions to capture data from cheetah locomotion. These expeditions will take place in various locations. The ultimate aim of this phase is to build a sizeable dataset containing synchronised multi-sensor data captured during interesting periods of motion of the subject cheetah. Generally we will aim to capture data during high-speed manoeuvres where the cheetah is either rapidly changing direction or running linearly at a high velocity.

Pose Reconstruction Software Design

The final milestone is the use of the aforementioned dataset to reconstruct 3D pose data of the cheetah during segments of locomotion of particular interest. This will build upon previous post-processing methods predicated upon a stereo camera setup, incorporating the additional sensor data of WildPose to achieve superior reconstruction accuracy.

1.6.2 Document Structure

The structure of this document loosely follows the timeline of the above described milestones; however, there are minor changes to the overall flow. A diagram depicting a high-level structural outline of the chapters of this thesis is shown below in Figure 1.2.

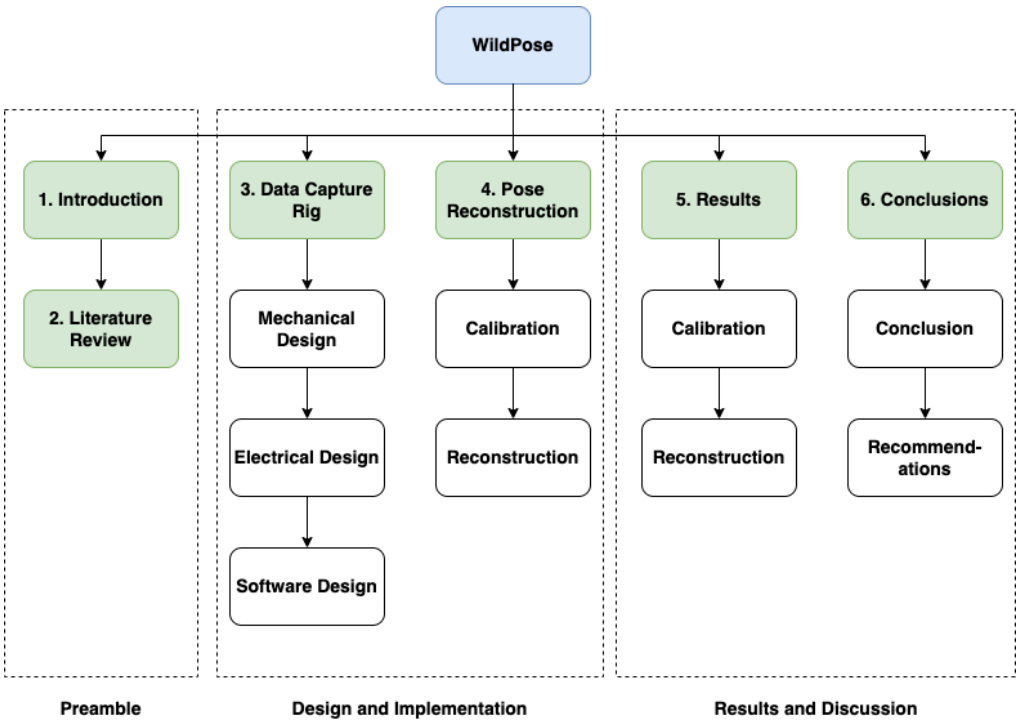

Figure 1.2: A diagram depicting the structure of this document

Chapter 2

Literature Review

The following chapter evaluates existing literature in the field of pose estimation, trajectory optimisation, and multi-sensor fusion, and provides a summary of relevant theory.

2.1 Multi-Sensor Fusion

Multi-sensor fusion refers to the usage of a combination of various forms of sensors to gather and analyse data. This technique is generally used to reduce the variance inherent in the use of a single sensor or type of sensor. By combining a diverse set of data obtained from multiple sources - each with different limitations, resolution, and range - we may compensate for any weaknesses and create a more robust and rich set of information.

2.1.1 RGB Cameras

Cameras are sensors that capture visual data by converting light reflected off 3D objects to a 2D image through various mechanisms. Modern cameras commonly achieve this through an array of complementary metal oxide

semiconductor (CMOS) sensors, which respond to light by accumulating an electrical charge that may be measured, transported and recorded [13]. Through this method, modern digital cameras are able to convert the analogue 3D world around them to a digital 2D image that may be analysed and processed as needed. Although images lack explicit depth measurements, they still provide a wealth of information about the captured scene and remain a powerful and widely studied method for pose estimation in the current body of research [14] [15] [16] [17].

In the literature, cameras are approximated by a model known as the **pinhole camera**. This refers to the idea that light passes through a pinhole - also known as the aperture of the camera - and is projected onto a plane behind the aperture where it is captured as an image [18]. Figure 2.1 below further illustrates this concept of a pinhole camera model.

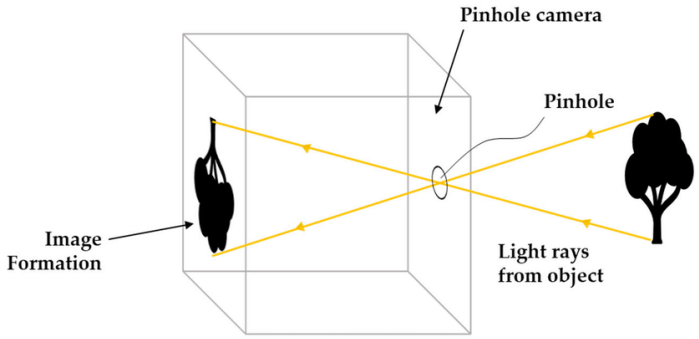

Figure 2.1: A simplified diagram illustrating a pinhole camera model.

This model is extremely useful in the field of computer vision as it allows for the development of mathematical models to convert between points in the image frame and the world frame. A pinhole camera model represents an idealised version of a camera. In reality, cameras tend to deviate from the exact geometry represented by this model. However, modern computer vision techniques compensate for this deviation using amendments to the ideal pinhole camera model known as the **projective model** and the **distortion model**. Through these extensions it is theoretically possible to model the vast majority of currently existing cameras in a relatively accurate manner.

The Projective Model

The projective camera model [19] assumes an ideal pinhole camera model as a base, and describes the transformation of a 3D real-world point into a point on the image plane. Figure 2.2 illustrates the assumptions that form the basis of the projective camera model.

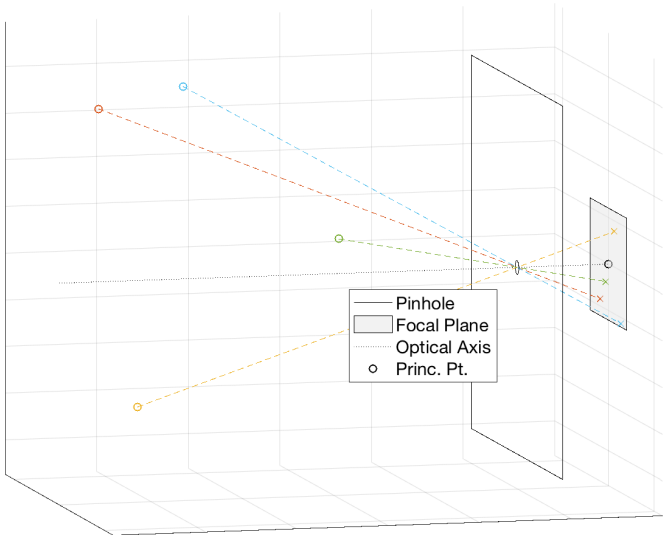

Figure 2.2: A diagram illustrating the basic principles of a projective camera model.

In essence, this model provides us with a transformation matrix F such that:

$$\mathbf{x_i} = \mathbf{F}\ (\mathbf{X_i}) \tag{2.1}$$

Here, $\mathrm{x_i}$ denotes the image point with index i, described in pixel coordinates as:

$$\mathbf{x_i} = [\ x_i,\ y_i\ ]^T \tag{2.2}$$

The 3D world point $\mathrm{X_i}$ is likewise simply defined as:

$$\mathbf{X_i} = [\ X_i,\ Y_i,\ Z_i\ ]^T \tag{2.3}$$

To apply these transforms, we require three main components:

1. The intrinsic camera parameters describing a pinhole camera model
2. The extrinsic camera parameters, which describe the camera's position in the 3D world frame
3. The camera distortion parameters to describe the camera's deviation from an ideal pinhole camera model

Once all three of these components have been determined mathematically to a satisfactory degree of accuracy, we may then apply an overall transformation to convert between world points and image points in either direction.

Intrinsic Camera Parameters

A pinhole camera model is parameterised through an intrinsic camera matrix, composed of three main elements: focal length, principle point, and skew. For most cameras, the skew parameter is assumed to be zero. Skew tends only to be introduced in uncommon situations such as photographing another photograph, and so most approaches focus solely on calculating focal length and principle points [20].

Focal length, as its name suggests, describes the distance from the pinhole to the focal plane. In the intrinsic camera matrix, focal length is separated into two components: f_x and f_y. Usually, f_x and f_y are equal or their difference is negligible, resulting in square pixel shapes. In the case where they are not, a scaling factor α is introduced to account for the difference:

$$\mathrm{f_x} = \alpha \cdot \mathrm{f_y} \tag{2.4}$$

However, in the vast majority of cases, $\alpha = 1$.

The principle point is a point in pixel coordinates where the optical axis intersects with the focal plane. Generally, this is near the centre of the image, meaning $\mathrm{p_x} = w/2$ and $\mathrm{p_y} = h/2$.

Overall, these parameters are combined in the camera intrinsic matrix $\mathbf{K}$ as follows:

$$\mathbf{K} = \begin{bmatrix} \mathrm{f_x} & 0 & \mathrm{p_x} \\ 0 & \mathrm{f_y} & \mathrm{p_y} \\ 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

The calculation of this intrinsic camera matrix is the goal of intrinsic camera calibration. In some cases, intrinsic and extrinsic calibrations are performed simultaneously, such as in a sparse bundle adjustment [21]; however, since the camera intrinsic camera matrix is theoretically constant given that a camera's focus and zoom do not change, the intrinsic matrix is often obtained first and remains fixed throughout future calculations.

Extrinsic Camera Parameters

The extrinsics of a camera refer to that camera's position in the world coordinate system [22]. Whereas the camera intrinsics denote a transformation between an image point and a world point in the camera frame (i.e. relative to the camera's position), the camera extrinsics describe the difference the difference between the camera frame and the world frame. The camera extrinsics can be represented in two ways: via an axes transformation, or via a point coordinate transformation. Both transformations include two parts: a rotation vector and a translation vector.

An axes transformation is analogous to the camera's pose in the world frame. A rotation and translation vector are used to represent the

transformation of a point from the camera frame to the world frame. For these purposes, the origin of the camera's coordinate system is considered to be the location of the aperture.

A point coordinate transformation is the inverse of the axes transformation. As such, the point coordinate transform converts a point from the world frame to the camera frame, whereas the axes transformation does the reverse. This transformation can be used interchangeably with the axes transformation given that the equation is rearranged to account for the differences; thus, the decision of whether to use a point transform or an axes transform is left up to whichever convention a researcher prefers, as long as future mathematical analysis remains consistent with the choice.

Distortion Modelling

In reality, cameras do not fully lie within the parameters assumed with a pinhole camera model. Distortion models parameterise the differences between an ideal pinhole camera and the actual real-world camera [23]. Typically, the source of distortion that has the largest effect on pinhole camera models is inaccuracy in the lens, denoted **lens distortion**.

There are two main types of lens distortion: radial distortion and tangential distortion. During a camera calibration process, both are calculated and represented using a five-element vector that will be outlined in more detail later.

Radial distortion occurs due to differences in the way light is refracted nearer to the centre of a lens and nearer to the edge. Positive and negative radial distortion is possible; Figure 2.3 illustrates the visual differences between the two.

Mathematically, radial distortion may be represented through the use of three **distortion coefficients**: k_1, k_2, and k_3. These affect the x and y coordinates of a point in the image plane as follows:

Figure 2.3: An image showing the effects of radial distortion - from left to right: zero, negative, and positive.

$$x_{distort} = x(1 \ + \ k_1 r^2 \ + \ k_2 r^4 \ + \ k_3 r^6) \tag{2.6}$$

$$y_{distort} = y(1 \ + \ k_1 r^2 \ + \ k_2 r^4 \ + \ k_3 r^6) \tag{2.7}$$

Where:

$$r = \sqrt{x^2 \ + \ y^2} \tag{2.8}$$

Tangential distortion occurs when a lens is not perfectly parallel with the focal plane. As intuition suggests, this can cause an image to appear slanted or tilted. Figure 2.4 depicts this effect on a straight grid for reference.

On the mathematical side, tangential distortion is typically represented by

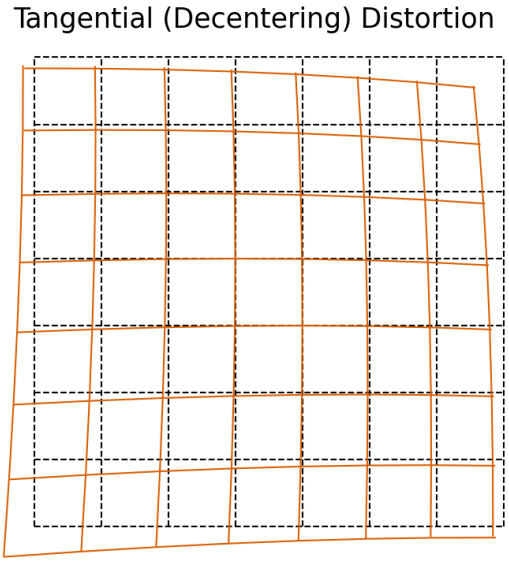

Figure 2.4: The effect of tangential distortion on an image.

two coefficients p_1 and p_2. The distorted coordinate system is described by:

$$x_{distort} = x + [2p_1xy + p_2(r^2 + 2x^2)] \tag{2.9}$$

$$y_{distort} = y + [p_1(r^2 + 2y^2) + 2p_2xy] \tag{2.10}$$

Certain calibration methods make use of more coefficients to represent radial distortion, tangential distortion, and another distortion parameter called **thin-prism distortion**. Although this is not necessary, in this case the full equation becomes:

$$x_{distort} = x\frac{(1 \ + \ k_1r^2 \ + \ k_2r^4 \ + \ k_3r^6)}{(1 \ + \ k_4r^2 \ + \ k_5r^4 \ + \ k_6r^6)} + 2p_1xy + p_2(r^2 + 2x^2) + s_1r^2 + s_2r^4 \tag{2.11}$$

$$y_{distort} = y\frac{(1 \ + \ k_1r^2 \ + \ k_2r^4 \ + \ k_3r^6)}{(1 \ + \ k_4r^2 \ + \ k_5r^4 \ + \ k_6r^6)} + 2p_2xy + p_1(r^2 + 2y^2) + s_3r^2 + s_4r^4 \tag{2.12}$$

2.1.2 Lidar

Lidar is a shorthand for Light Detection and Ranging. Lidar functions similarly to radar; the main difference, as the acronym suggests, is that lidar makes use of light whilst radar makes use of radio waves.

In essence, a lidar device uses time-of-flight calculations to map a 3D space through the emission of near-infrared wavelengths of light [24]. Time-of-flight refers to a measurement of the time taken for a projectile or signal to travel back and forth between two points. In the case of a lidar system, pulses of light are emitted by a laser emitter within the device. This light is then reflected off of an object and returns to the device to be received by a light sensor, typically some form of pinhole camera. The time taken for the pulse to return to the sensor is measured and converted to a distance using the constant speed of light (299 792 458 m/s). Figure 2.5 shows a diagram of the principle behind a lidar system.

This process is repeated thousands of times during a single lidar scan. The shape in which light pulses are emitted from the lidar is known as the **scanning pattern**, and various scanning patterns are used across the spectrum of lidar devices. Additionally, the amount of times light is emitted and received during a single scan varies. This rate determines the resolution of the lidar and is often referred to as the **point cloud density**.

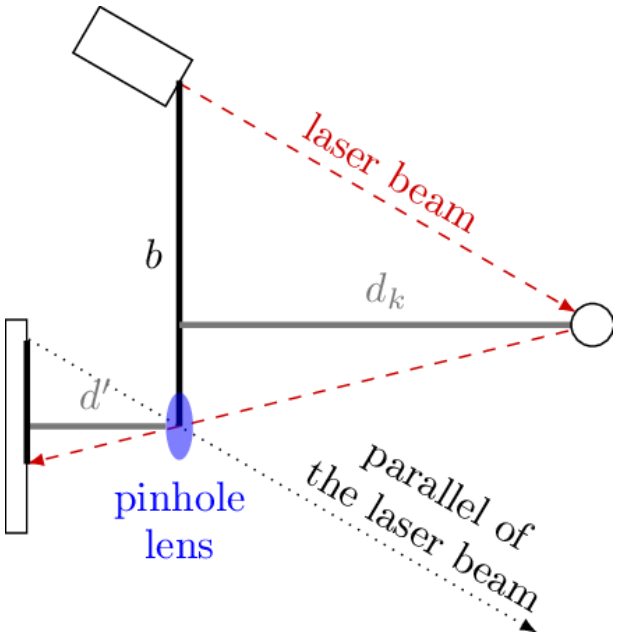

Figure 2.5: A simplified diagram showing the basic process behind a lidar distance measurement.

The mechanism behind lidar generally takes one of two forms: scanning (or mechanical) lidar, and solid state lidar. Scanning lidar uses a single laser that mechanically rotates 360 degrees. The device then uses time-of-flight cameras to determine the distance of each real-world obstacle the laser beam encountered during its period of travel. These devices tend to be far more mechanically complex than solid state lidars, and thus are more prone to failure as well as being typically more expensive.

Solid state lidars, on the other hand, use a single static laser to "flash" laser light at their surroundings. The resultant returning laser light is then captured by an array of time-of-flight cameras to determine distance. These lidars have no moving parts and tend to be cheaper; however, they usually do not capture information at a full 360 degree range. These lidars generally may record data in ranges of up to 270 degrees.

The data recorded by a lidar device is usually visualised in a **point cloud**. Unsurprisingly, this takes on the appearance of a cloud of points in 3D space. Aside from the expected x, y, and z readings for each points in the point cloud, most lidars also record the **reflectivity** of the object scanned.

By measuring minute changes in the wavelength of the light that has returned to the sensor, a numerical representation of the level of reflectivity for that point may be obtained. These readings are extremely useful for differentiating between objects in a point cloud, and can even allow for the inference of colours; a dark object will be viewed as having a lower reflectivity, whilst a light coloured object will display a higher reflectivity. Figure 2.6 shows a visualisation of a typical point cloud recorded by a lidar device atop a self-driving car. Usually, the colour of the points in a graphical representation of a point cloud corresponds to the reflectivity of each point, with brighter colours denoting higher reflectivity.

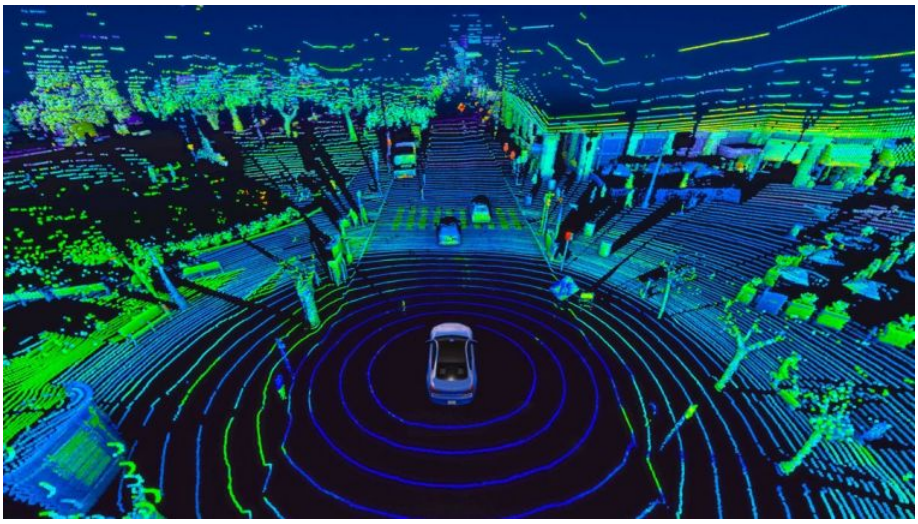

Figure 2.6: A point cloud visualising a lidar scan of a self-driving car's surroundings.

Lidar is widely used for mapping applications, such as long-range topographical mapping of natural structures, ocean floor mapping, and environment mapping for self-driving car algorithms [25] [26] [27] [28]. However, the usefulness of lidar is not limited to the mapping of static structures. The 3D data delivered by lidar scans may be useful for any application that requires information about a subject in 3D space; its high accuracy and the rich information is provides are both assets in the pursuit of 3D pose estimation.

2.1.3 Inertial Measurement Units

Inertial Measurement Units (IMUs) are devices that measure the net force and acceleration of the body to which they are attached. Generally, IMUs are comprised of two main types of sensors to provide a complete picture of the body's motion:

- Gyroscopes, which measure angular velocity
- Accelerometers, which measure proper acceleration in the inertial frame

Through the combination of these sensors, IMUs are able to record the trajectory of an object in terms of its inertial translation and rotation [29]. This data may be used to transform between the coordinate system of a moving platform and the world coordinate system.

Accelerometers exist in many different forms, but the most common type is the MEMS (micro electromechanical system) accelerometer. MEMS accelerometers consist of a mass and spring system where the mass may only translate in a certain direction known as the **sensitivity axis**. When the MEMS accelerometer experiences proper acceleration (acceleration relative to free fall, or non-gravitational acceleration), the mass will move along the sensitivity axis. This displacement will be recorded by the device, and will correspond to the proper acceleration felt by the device at the point in time. Figure 2.7 further illustrates this system and its inner mechanics.

Gyroscopes are sensors that measure an object's angular velocity in the inertial frame. Again, MEMS gyroscopes tend to be the most commonly used systems, although quartz gyroscopes are occasionally used in industry and consumer markets. On the mechanical level, MEMS gyroscopes consist of a spring and mass system where the rotational velocity of an object is measured by reading the displacement of two masses around a rotational axis. Figure 2.8 illustrates this effect.

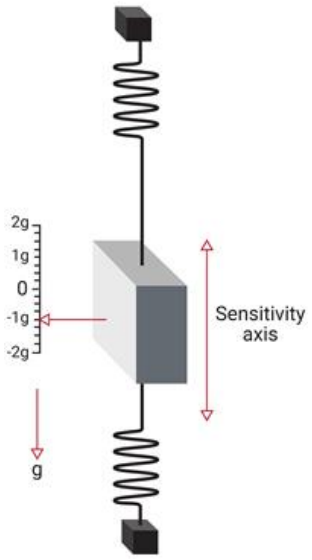

Figure 2.7: A diagram illustrating the inner mechanism of an accelerometer.

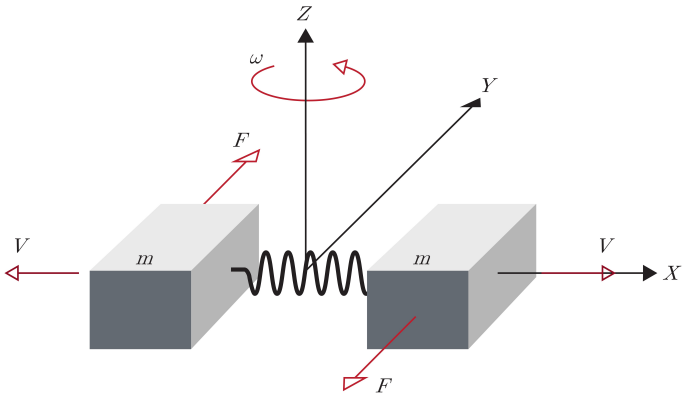

Figure 2.8: A diagram illustrating the inner mechanism of a gyroscope.

2.2 Pose Estimation

Pose estimation is the process of estimating the configuration of a body, animal or otherwise, from raw data obtained from one or more sensors. 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 [30]. 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 2.9.

Figure 2.9: An example of the output data of a 2D pose estimation algorithm for a human subject [2].

2.2.1 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 [31], 3D hand pose estimation [32], and - pertaining to this thesis - 2D pose estimation across species in the wild [33] [34] [3]. 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 [3] 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 [35]. 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 [36] [37].

Given however the inherently analogue nature of pose estimation, even the most efficient 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 2.10 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 [3], 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.

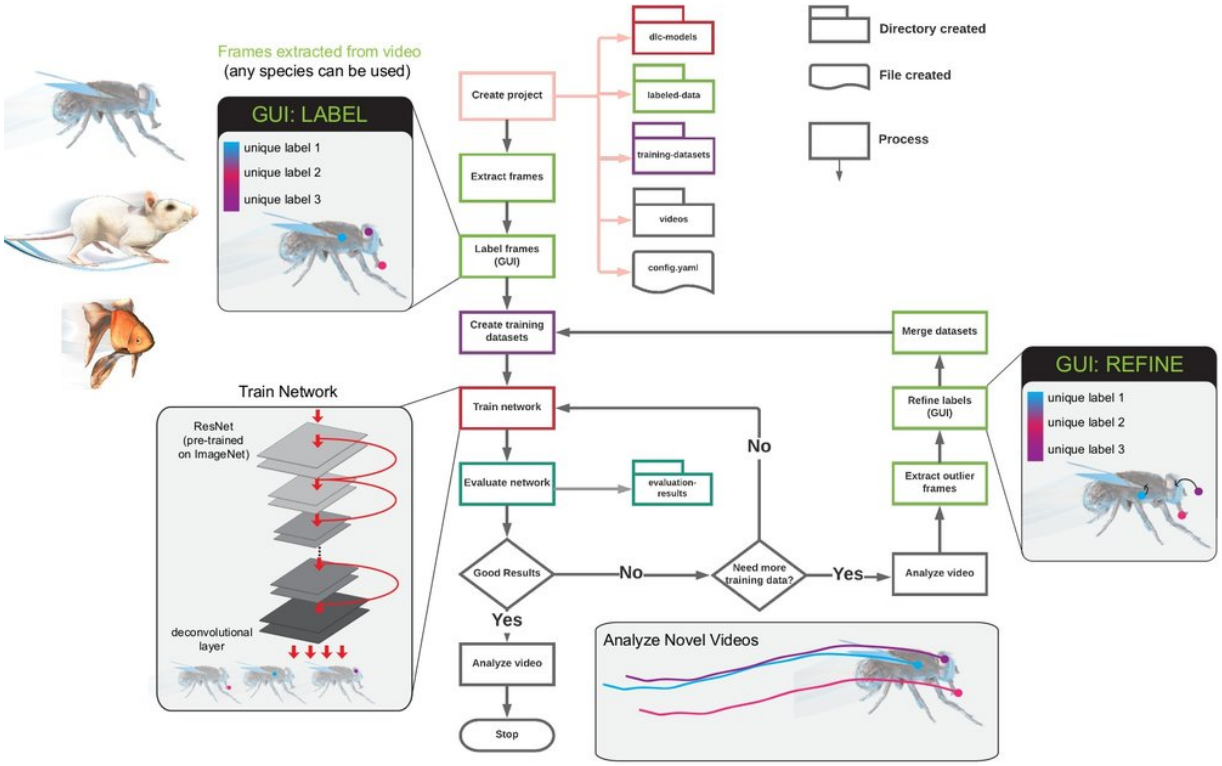

Figure 2.10: An overview of the training and evaluation cycle for DeepLabCut [3].

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.

2.2.2 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 [4] 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.

Zhao et al [5] 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 2.12.

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 2.13 is a diagram describing the general process of obtaining a 3D pose from a cheetah in the

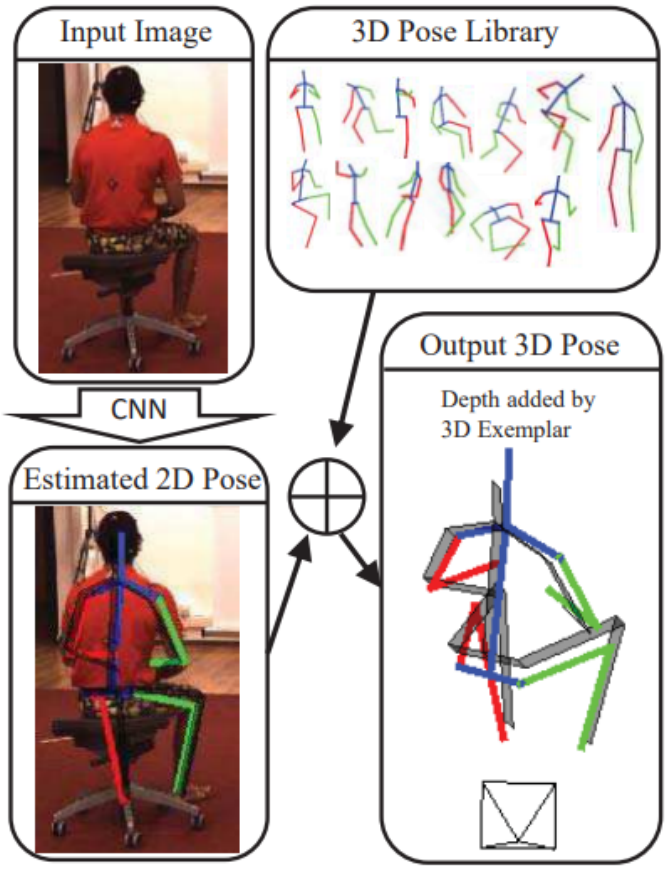

Figure 2.11: The matching method for 3D pose estimation from a single image [4]

wild using a DeepLabCut-trained CNN for 2D pose detection [6].

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 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.

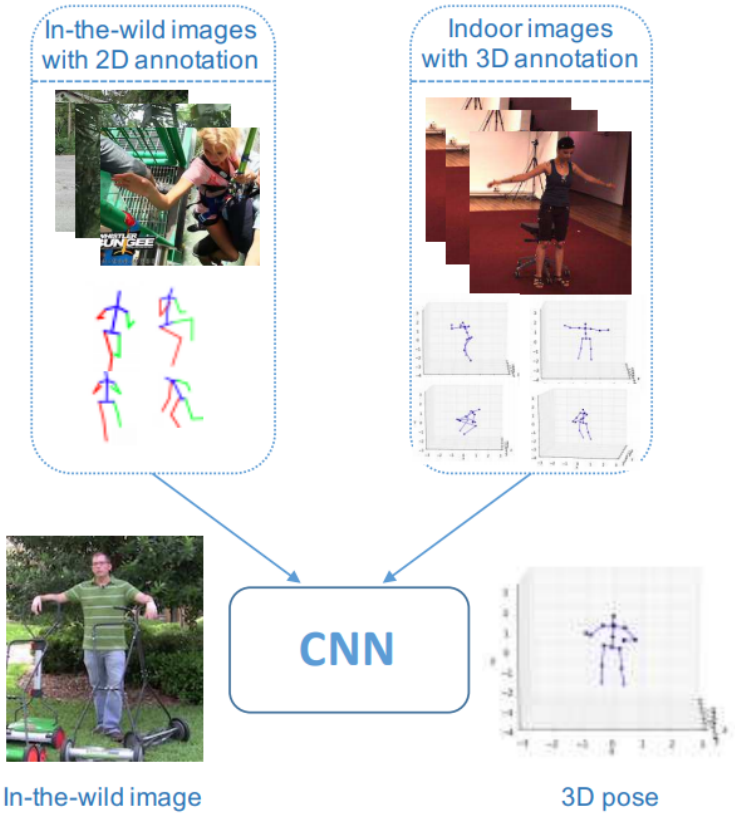

Figure 2.12: An overview of the process used by Zhao et al to combine 2D and 3D pose labels for 3D pose estimation [5]

2.2.3 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 [38] 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 2.14.

While triangulation is useful for the derivation of 3D pose given multiple 2D viewpoints, careful curating of the 2D data is required. Errors in

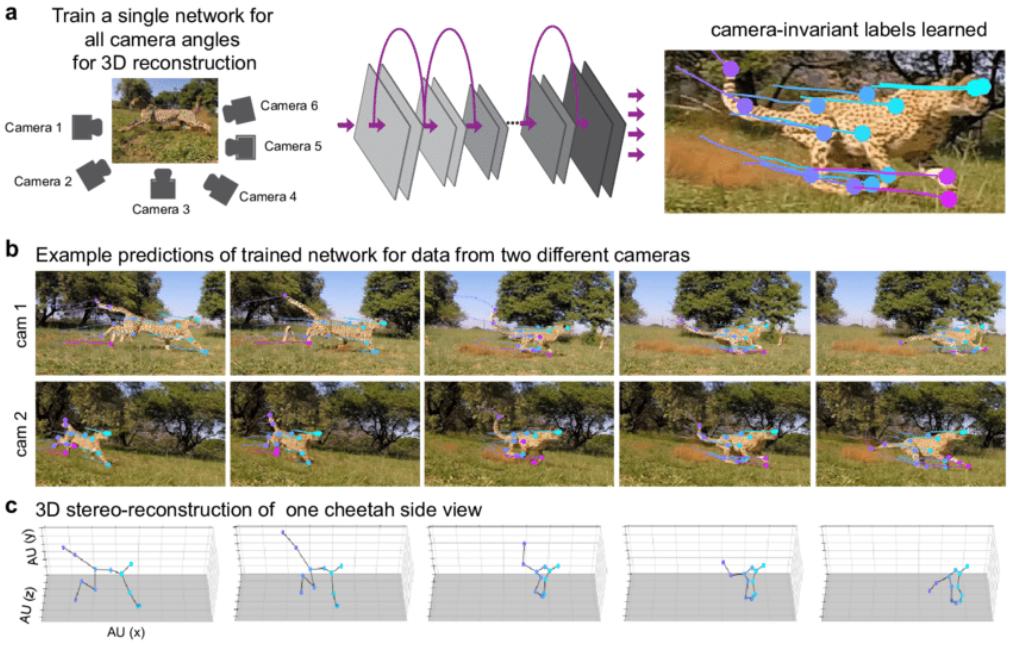

Figure 2.13: A visual overview of the process of performing a stereo reconstruction of a cheetah pose captured from multiple views using DeepLabCut [6]

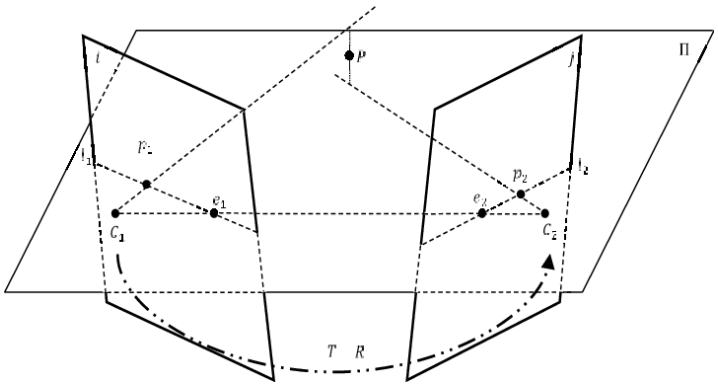

Figure 2.14: A visualisation of the triangulation of a point from two 2D views [7].

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 [38].

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.

2.2.4 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 [39].

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 first -order partial derivatives of a vector-valued function with respect to the vector, as shown below.

$$\mathbf{J} = \begin{bmatrix} \frac{\delta f_1}{\delta x_1} & \cdots & \frac{\delta f_1}{\delta x_n} \\ \vdots & \ddots & \vdots \\ \frac{\delta f_m}{\delta x_1} & \cdots & \frac{\delta f_m}{\delta x_n} \end{bmatrix} \tag{2.13}$$

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.

$$\theta = [\mathbf{P}_1,\ \mathbf{P}_2,\ ...,\ \mathbf{P}_n,\ \mathbf{X}_1,\ \mathbf{X}_2,\ ...,\ \mathbf{X}_m] \tag{2.14}$$

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.

$$\mathbf{J} = \frac{\delta \mathrm{x}}{\delta \theta} \tag{2.15}$$

However, only the corresponding camera parameters for each 2D point reprojected to the relevant image plane may be non-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 partial derivative 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 partial derivatives 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 2.15. 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.

As seen above, the Jacobian matrix consists of two main sections: the

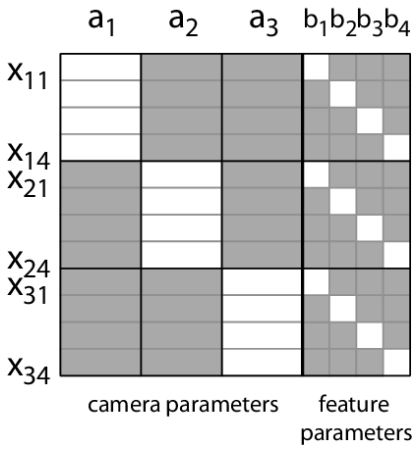

Figure 2.15: A visualisation of the Jacobian matrix for an example SBA problem. Grey zones indicate zero (or null) values, whereas bright zones denote non-zero partial derivatives.

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.

2.2.5 Odometry

Odometry refers to the estimation of a sensor's position in the world frame given some amount of data in the sensor frame. The two main forms of odometry researched and discussed here are visual odometry and lidar odometry.

Visual Odometry

Visual odometry is the most widely studied and applied version of odometry. This process involves estimating a camera's extrinsic matrix, or pose, based only on visual information recorded by the camera. Generally, this is achieved through feature matching in subsequent frames. Features

may be extracted from an image space using multiple methods; however, the two most common methods of visual feature extraction for odometry are SIFT (Scale Invariant Feature Transform) and, as an extension to this method, SURF (Speeded Up Robust Features).

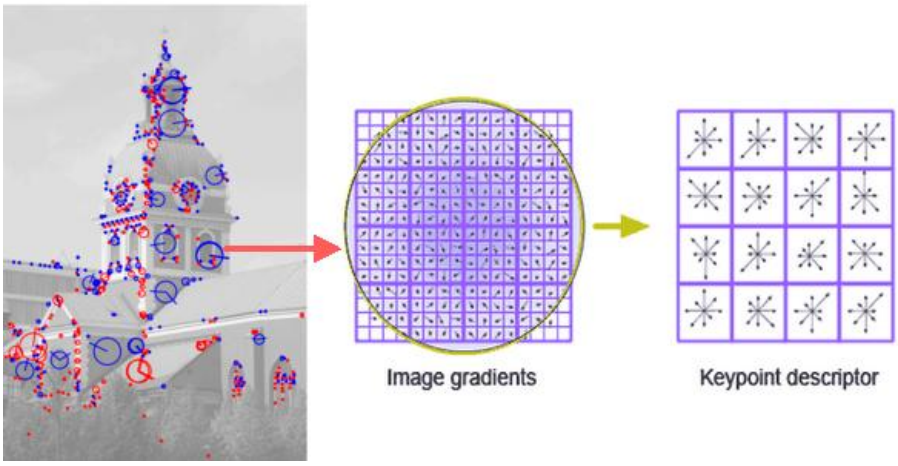

Figure 2.16: A visual depiction of the extraction of SIFT features from an image using image gradients.

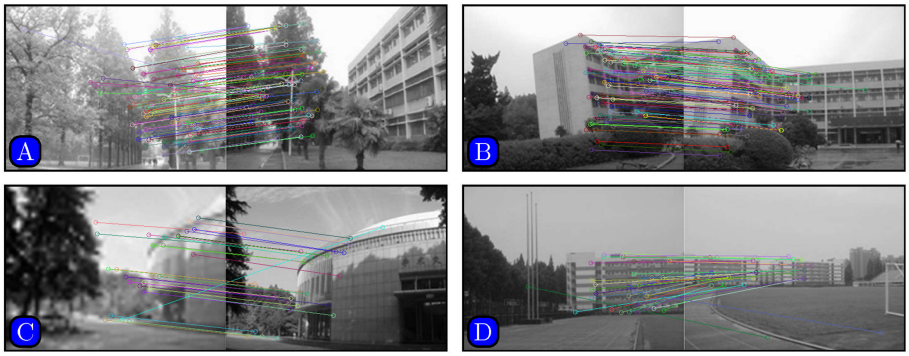

Figure 2.17: A diagram showing the process of SURF feature matching between frames.

Generally, SURF has been found to perform slightly better and run slightly faster than SIFT feature extraction [40] for monocular visual odometry. However, both methods are useful for situations where no additional pose data is provided regarding a camera's position in the world frame. The use of only visual data to reconstruct camera pose along a trajectory is an invaluable tool for many research applications.

2.3 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 [41] 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 manoeuvres. This study is only one example of the many applications of trajectory optimisation within the field of mechatronic design.

Shown in Figure 2.18 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

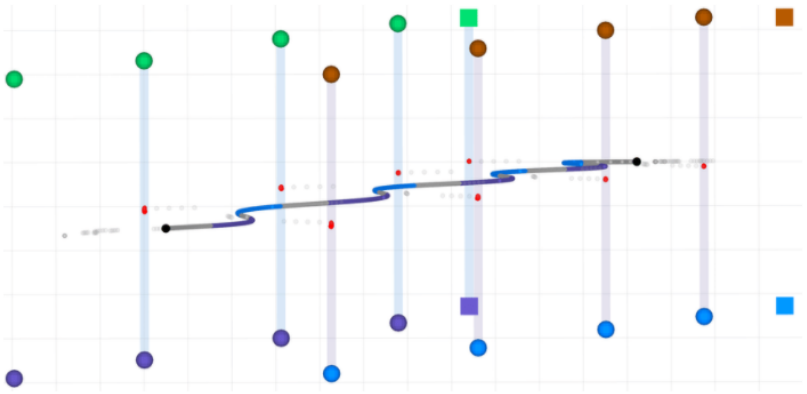

Figure 2.18: A plotted optimised trajectory for a quadrupedal robot [8].

robot morphology (among other parameters) across iterations. Figure 2.19 shows a bipedal robot model with seven degrees of freedom that was used in the aforementioned trajectory optimisations.

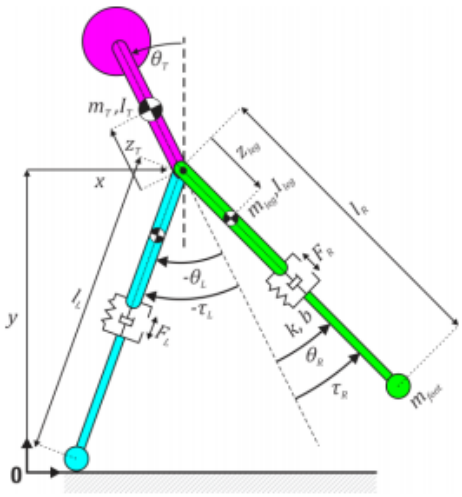

Figure 2.19: A bipedal robot model used to study methods of rapid deceleration in legged robots [9].

2.4 Summary

The literature and theory outlined above serves as a foundation for the development of our method for data capture and pose estimation. The

main insights gained from the above are summarised here.

First, it is clear that, given the opportunity to design both the method of acquisition of data as well as the downstream reconstruction methods, a multi-sensor approach seems natural. Further, the combination of sensors conducive to robust 3D pose estimation in the world frame may be narrowed down to lidar, RGB, and IMU devices. Conventional RGB cameras provide rich visual information from which to estimate a 2D pose, while lidar provides high-accuracy depth information as well as valuable insight into the reflectivity of the subject that RGB does not cover. Finally, IMU data provides a robust method for the conversion of 3D pose data from the rig frame to the world frame, although the lack of an IMU may be compensated for by vision- and lidar-based odometry methods.

Second, a trajectory optimisation-based approach will assist in the calculation of 3D skeletal pose from the recorded data. Trajectory optimisation is a well tested method in applications such as these and is able to make great use of temporal information for outlier rejection and the further increase of pose accuracy. The advantages of trajectory optimisation are numerous: we are able to define a kinematic model for the subject in order to restrict pose to the naturally possible, we may introduce physical constraints for the elimination of physically implausible trajectories, and we are likely to obtain a much smoother end result by fitting a trajectory optimisation based model on top of a naive vision- and depth-based estimate.

The above theory will inform the mechanical, electronic, and software design of the WildPose data capture rig as well as the software design of the reconstruction algorithm outlined in sections to follow.

Chapter 3

Data Capture Rig

This chapter will outline the mechanical, electronic, and software design of the WildPose data capture rig. It will describe in detail the rationale behind design changes made between iterations, and provide schematics for full replication of the device.

3.1 Overall System Design

The are two main categories of user that will interact with the data capture rig: the researcher and the operator. The researcher refers to one who aims to analyse data captured by the rig, as well as one who is involved with maintaining and designing the rig. This user requires a high amount of technical knowledge about the design and functionality of both the rig and the data itself. The operator refers to the user who will operate the rig in the field and record raw data for later analysis. This user need not possess extensive technical knowledge about the rig itself, but need only follow basic operating procedure during the recording process. Camera operation is the primary skill and task of this user. These two user roles may overlap as a single person may perform the tasks of both the researcher and operator. However, it is important that the design of the rig

accommodate for separate roles and provide the necessary functionality for specific users without overwhelming complexity and scope. The high-level use case requirements of the rig are summarised in the Use-Case diagram shown in Figure 3.1.

Figure 3.1: A Use-Case diagram showing high-level user requirements of the WildPose rig.

Once we have outlined the division in requirements between the researcher and operator, we may begin specifying overall user requirements for the WildPose rig. These will be further refined by the system-level functional requirements and the technical specifications for each major subsystem: the mechanical, electronic, and software subsystems.

3.2 System-Level Requirements

The first step in outlining a set of useful design specifications for the rig is the deciding of a set of high-level system user requirements. These are broad requirements that describe general minimum functionality that the final design of the rig will need to fulfil.

3.2.1 User Requirements

There are several conventions one may follow in defining user requirements, as well as in subsequently refining them. Here, it was chosen to outline a set of base system user requirements - where each has a unique identifier, a description, and a list of further refined requirements - in a simple tabular format.

The user requirements are outlined in Tables 3.1 to 3.6.

UR 1	**Operation Time**
Description	The data capture rig should be able to record continuously for a long enough time for the researcher to capture desired motion without need for recharge.
Rationale	During field operations, it is common that the user will need to keep the rig running and recording for extended periods of time in order to ensure that the desired motion is captured.

Table 3.1: A table outlining system-level user requirement 1.

UR 2	**Data Storage**
Description	The data capture rig will need to be able to record and store a liberal amount of footage and data to its internal storage.
Rationale	During a field expedition, it may arise that a researcher will be out in the field for full days at a time with no access to external storage to offload data.

Table 3.2: A table outlining system-level user requirement 2.

UR 3	**Capture Range**
Description	The data capture rig will need to be able to record usable research data of a subject at far distances.
Rationale	In many situations, a researcher will be unable to assume a close vantage point from the desired subject. Thus, high zoom capabilities are required.

Table 3.3: A table outlining system-level user requirement 3.

UR 4	**Dynamic Zoom and Focus**
Description	The data capture rig will need to be able to zoom and refocus mid-data stream without loss of data or extrinsic calibration accuracy.
Rationale	It may occur that a subject moving at high speeds will translate closer to or further from the camera mid-movement. Accurate data in this case is still required.

Table 3.4: A table outlining system-level user requirement 4.

UR 5	**Durability**
Description	The data capture rig will need to be able to endure long periods of off-road activity without the need for significant maintenance.
Rationale	As mentioned above, a researcher may need to spend full days at a time in the field without access to advanced maintenance equipment such as soldering irons. In these cases, the rig should be reliable.

Table 3.5: A table outlining system-level user requirement 5.

UR 6	**Ease of Operation**
Description	The data capture must be simple enough to operate that a researcher without in-depth technical knowledge of the rig's inner working should be able to make effective use of it.
Rationale	The data capture rig should be accessible to a wide range of researchers if need be. Thus, the pool of qualified users should not be so limited as to those involved in the development of the rig.

Table 3.6: A table outlining system-level user requirement 6.

3.2.2 Functional Requirements

The system-level functional requirements further refine the aforementioned user requirements, and are presented in much the same way as before. The functional requirements differ in that they attempt to specify what is meant by the high-level use requirements and attach a measurable metric to each one, where possible.

FR 1	**Operation Time**
Description	The data capture rig must be able to record continuously for up to 10 minutes without loss of data
Refines	UR 1
Rationale	The rig will at times be left recording whilst a subject is followed. A period of 10 minutes was selected, as this is plenty of time to capture interesting motion.

Table 3.7: A table outlining system-level functional requirement 1.

FR 2	**Data Storage Requirement**
Description	The data capture rig must be able to store up to 10 hours' worth of footage in internal storage.
Refines	UR 2
Rationale	The rig may be used in the field for a full day of work before storage may be cleared. 10 hours spans a full work day of constant recording.

Table 3.8: A table outlining system-level functional requirement 2.

FR 3	**Capture Range**
Description	The data capture rig should be able to capture a 1.5 m long subject from 100 m away with an in-frame length of no less than 50 pixels.
Refines	UR 3
Rationale	Larger cheetahs can reach 1.5m in length. 100m approximates a far, but serviceable range, and 50 pixels provides adequate pixel density for 2D pose estimation.

Table 3.9: A table outlining system-level user requirement 3.

FR 4	**Dynamic Zoom and Focus**
Description	The data capture rig will have a focal length range of (at least) 5 mm to 150 mm.
Refines	UR 4
Rationale	Lenses within these ranges are generally able to capture 1.5m subjects in high detail at 5m to 100m ranges.

Table 3.10: A table outlining system-level user requirement 4.

FR 5	**Durability Requirement**
Description	The data capture rig must be able to survive 12 hours of off-road activity with zero loss of functionality.
Refines	UR 5
Rationale	12 hours covers a full day of recording in the field as specified above, plus a 2 hour safety margin for additional travelling.

Table 3.11: A table outlining system-level user requirement 5.

FR 6	**Ease of Operation**
Description	The data capture rig shall use controls of the minimum possible complexity and not require excessive amounts of setup time or technical knowledge to operate.
Refines	UR 6
Rationale	The rig operators are not necessarily engineers; thus, the rig controls would benefit from simplicity.

Table 3.12: A table outlining system-level user requirement 6.

3.3 Mechanical Subsystem Design

This section will describe the mechanical design of all iterations of the WildPose rig as of the writing of this document. Images of the CAD models will be provided for each iteration and the design choices made in each instance will be explained.

3.3.1 Subsystem-Level Functional Requirements

The mechanical subsystem-level functional requirements are outlined below.

MECH FR 1	**Focus and Zoom Control**
Description	The focus and zoom rings of the camera should be mechanically linked to a control mechanism
Refines	FR 6

Table 3.13: A table outlining subsystem-level functional requirement 1.

MECH FR 2	**Durability**
Description	The data capture rig should be able to survive 12 hours of mild to moderate vibration without loss of functionality
Refines	FR 5

Table 3.14: A table outlining subsystem-level functional requirement 2.

MECH FR 3	**Portability**
Description	The data capture rig should be portable enough to be transported and set up by a single user with relative ease.
Refines	FR 6

Table 3.15: A table outlining subsystem-level functional requirement 3.

3.3.2 First Prototype

The first prototype of the data collection rig made use of two separate free-standing tripods: one to support the microcontroller and touch screen, and the other to house a moving platform upon which the lidar, camera, and rotary encoders sat. This design, as well as the second iteration of the rig, was static and required the operator to find a vantage point that was both safe and able to accommodate the tripods. Long-term this solution was deemed unsuitable; however, it proved useful for recording test and validation data within enclosures.

A CAD model representing the design of the first iteration of the rig is shown in Figure 3.2. Note that whilst the tripods appear isolated in the model, they were connected by various cables in reality to allow both data and power to flow between the sections where necessary. An exploded view of the CAD model, providing greater visibility into the intricacies of the mechanical design, is shown in Figure 3.3.

Functional Requirements Met

MECH FR 2	**Durability**
Status	Fully Satisfied
Rationale	The data capture rig was deployed in the field and suffered no major damage. However, extended periods of vibration were not experienced as the rig was operated from a static platform.

MECH FR 3	**Portability**
Status	Fully Satisfied
Rationale	The data capture rig was technically portable; however, the two tripod setup was cumbersome and time-consuming to set up.

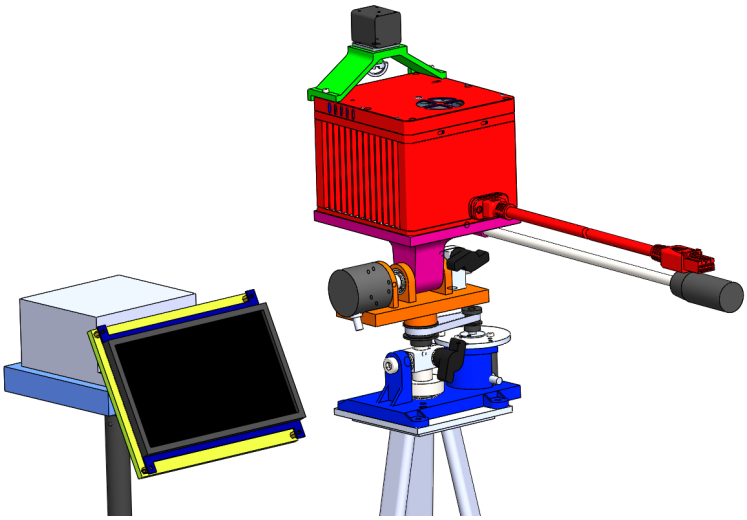

Figure 3.2: A CAD model of the first prototype of the data capture rig.

3.3.3 Second Iteration

After various test runs with the first prototype, the design was improved upon based on problems encountered in the field. The major shift in design was the use of a single tripod to house the entire rig rather than two separate stands. This was mainly decided upon due to the improved ease of transportation, increased user-friendliness and ease of operation, and to reduce the number of points of failure in the rig.

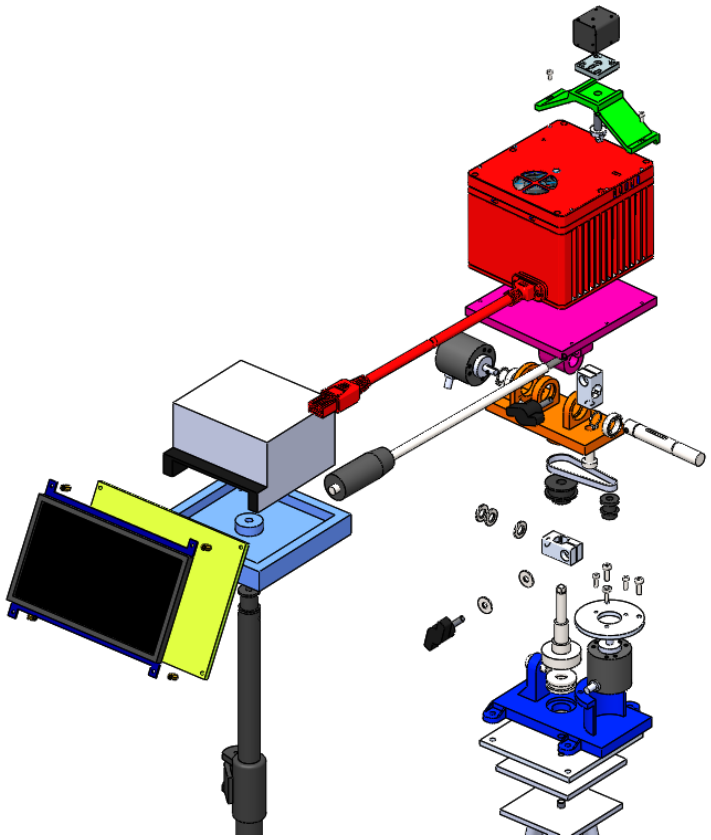

Figure 3.3: An exploded view of the first prototype CAD model.

A CAD model illustrating the design of the second iteration of the rig is depicted in Figure 3.4.

An exploded view of the CAD model of the second iteration is shown in Figure 3.5.

Functional Requirements Met

MECH FR 1	**Focus and Zoom Control**
Status	Partially Fulfilled
Rationale	A mechanical control system utilising combined motor-encoders was introduced for control over the zoom and focus rings. However, the joystick control component was not properly housed and was not deemed durable enough.

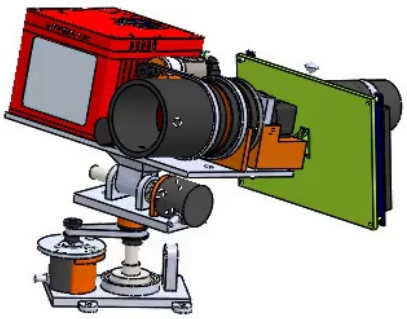

Figure 3.4: A CAD model of the second iteration of the data capture rig.

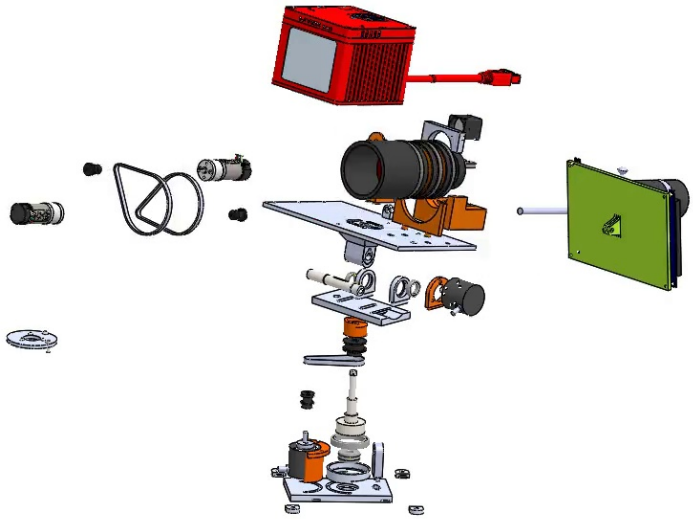

Figure 3.5: An exploded view of the second iteration CAD model.

MECH FR 2	**Durability**
Status	Partially Fulfilled
Rationale	Durability was improved in this iteration over the first due to a higher percentage of machined metal components and a single tripod. However, once again, extended periods of vibration were not tested.

MECH FR 3	**Portability**
Status	Fully Satisfied
Rationale	The data capture rig's portability was improved, once again, by the single tripod design, and setup time was significantly reduced.

3.3.4 Third Iteration

The third iteration of the rig was characterised by a move from the use of a tripod to the use of a car-mounted stand to house the machinery of the rig. This change was decided upon because the final data-gathering expedition was undertaken in Kgalagadi National Park, wherein it would not be possible to exit the vehicle to set up a tripod. The top platform design for the preceding iteration was kept, and the tripod section was replaced with a flat board that was strapped to the door of a motor vehicle. The operator would then sit inside the car and operate the rig from the back seat whilst recording subjects outside the window.

A photograph of the third iteration of the rig is shown below in Figure 3.6.

Figure 3.6: A photograph of the final version of the WildPose rig mounted on a car door.

Functional requirements Met

MECH FR 1	**Focus and Zoom Control**
Status	Fully Satisfied
Rationale	The previously implemented joystick control mechanism was kept, and a new 3D-printed housing for the component was designed and implemented for increased durability and protection from external environmental conditions.

MECH FR 2	**Durability**
Status	Fully Satisfied
Rationale	This iteration continued to make use of the machined steel parts from the previous iteration, which proved sturdy and durable. Multiple full daytime periods were spent on non-tar roads with the rig affixed to a car door and no major maintenance was required.

3.3.5 Subsystem-Level Technical Specifications

Once the final iteration of the data capture rig was implemented and tested in the field and it was determined that all functional requirements had been met, the final technical specifications for the rig were recorded. The mechanical subsystem specifications are outlined below, including the final component choices and the reasoning behind each.

MECH TS 1	**Focus and Zoom Control**
Description	To achieve precise focus and zoom control, motors were selected to actuate the zoom and focus rings of the camera lens, as well as a joystick component to provide a simple interface for the user.
Refines	MECH FR 1
Component Choice/s	MECH C1, MECH C2

Table 3.16: A table outlining subsystem-level technical specification 1.

Where the technical specifications required a component to be selected, the details of each chosen component are recorded in the tables below.

MECH TS 2	**Durability**
Description	The structural mechanical components of the rig, such as supporting arms, frames and brackets, are constructed from machined steel to maximise durability and minimise vibrations.
Refines	MECH FR 2

Table 3.17: A table outlining subsystem-level technical specification 2.

MECH TS 3	**Portability**
Description	The rig was constructed out of a detachable top platform that collapses to fit within a 30 cm x 30 cm x 30 cm area, and a collapsible tripod.
Refines	MECH FR 3

Table 3.18: A table outlining subsystem-level technical specification 3.

MECH C1	**99:1 Metal Gearmotor 25DX69L**
Rationale	Two of these parts were used to actuate the focus and zoom rings. The 25D motor is relatively cost-effective, contains a built-in rotary encoder with a high resolution, and provides ample torque for our use case.
Model/Part Number	25D Metal Gearmotor, Pololu Item No. 4867
Manufacturer	Pololu
Specifications	
Torque	98.78 km.mm
Motor Type	MP 12V brushed DC
Torque	16 km.mm
Gear Ratio	98.78:1
No-load Speed	79 RPM

The technical specifications - as well as the component choices - were experimentally validated during the various field tests undergone using each of the three iterations of the data capture rig.

MECH C2	**2-Axis Joystick Potentiometer**
Rationale	This is a simple component that provides a tactile and easily accessible interface for controlling the motors via dual potentiometers. It was chosen for its electrical simplicity, comfortable ergonomics, and durability.
Model/Part Number	Parallax Item No. 27800
Manufacturer	Parallax Inc.
Specifications	
Voltage Rating (DC)	10V
Resistance	10 kΩ
Switch Function	4-Way Directional

3.4 Electronic Subsystem Design

This section will describe the electronic design of the WildPose rig. Where possible, full electrical schematics will be provided and briefly explained.

3.4.1 Subsystem-Level Functional Requirements

ELEC FR 1	**Durability**
Description	The data capture rig's electronics should be able to survive 12 hours of mild to moderate vibration without loss of functionality
Refines	FR 5

Table 3.19: A table outlining subsystem-level functional requirement 1.

ELEC FR 2	**Motor Control of Zoom and Focus**
Description	The data capture rig shall make use of motors and a tactile input system to allow the user to interface precisely with the zoom and focus rings
Refines	FR 4, FR 6

Table 3.20: A table outlining subsystem-level functional requirement 2.

ELEC FR 3	**Storage Device Capacity**
Description	The chosen data storage component(s) for the rig should have at least 500 Gigabytes of usable storage in total.
Refines	FR 2

Table 3.21: A table outlining subsystem-level functional requirement 3.

ELEC FR 4	**Sufficient Memory**
Description	The chosen microcontroller for the rig should have sufficient RAM - in terms of speed and capacity - to allow for 10 continuous minutes of data capture.
Refines	FR 1

Table 3.22: A table outlining subsystem-level functional requirement 4.

3.4.2 First Prototype

The first prototype of the data capture rig was fairly basic in terms of electronics. It did not make use of rotating zoom and focus rings, as it instead made use of a fixed focal length lens. Thus, the only electronics present in this iteration of the rig were the electronics used to connect the pan- and tilt-measuring rotary encoders to the microcontroller. These minimal electronics were implemented using a veroboard, and as a result the first iteration was not robust enough to be considered a full success. However, the data capture expeditions undertaken with this iteration provided invaluable insight into enhancements to be made to future iterations.

Functional Requirements Met

ELEC FR 4	**Sufficient Memory**
Status	Partially Fulfilled
Rationale	The Jetson Xavier's 32 Gigabytes of RAM proved ample and the rig was generally able to record for 10 minutes without loss of data, but the system was prone to rare crashes which would cause the loss of valuable data in the field.

3.4.3 Second Iteration

The second iteration of the rig involved a complete overhaul to the electronics. Motors were introduced to control the zoom and focus rings of the new lens, and a tactile joystick was integrated into the electronic design for control of these motors. The motors had in-built rotary encoders, so these were integrated into the already existing encoder-buffer design.

This iteration also made use of veroboard for the implementation of the electronic subsytem. To protect the electronics from the elements, an insulating box was used to house the electrical elements. This resulted in higher robustness than the previous design; however, the use of veroboard remained undesirable and ultimately unreliable as loose connections and occasional loss of data did occur.

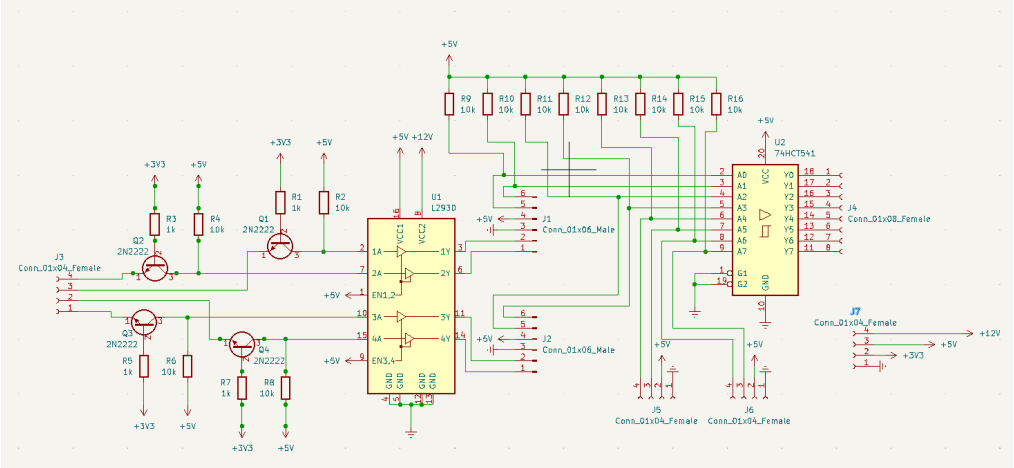

Figure 3.7: A circuit schematic depicting the input buffer, motor driver, and encoder headers for the second iteration of the rig.

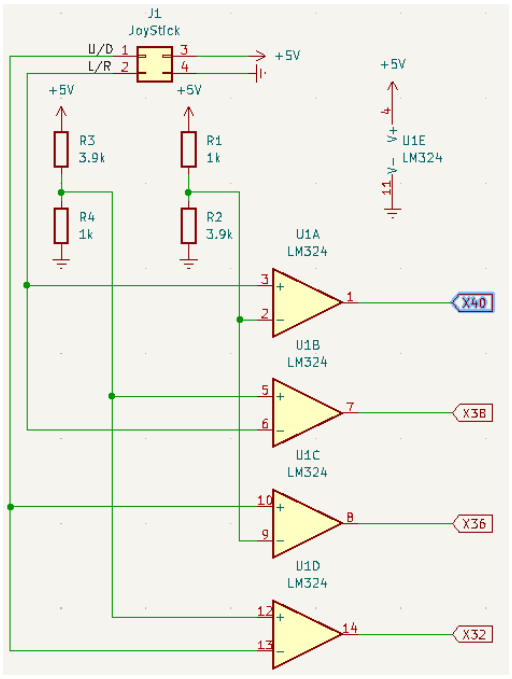

Figure 3.8: A circuit schematic depicting the joystick circuitry for the second iteration of the rig.

Functional Requirements Met

ELEC FR 1	**Durability**
Status	Partially Fulfilled
Rationale	In general, the rig's electronics were far more robust than in the previous iteration and the single tripod setup proved useful in minimising the points of failure. However, the rig did not undergo a long-term high-vibration test during the use of this iteration, and so could not be considered fully robust.

ELEC FR 2	**Motor Control of Zoom and Focus**
Status	Fully Satisfied
Rationale	In the second iteration, motors and a joystick were used to control the zoom and focus rings.

ELEC FR 3	**Storage Device Capacity**
Status	Fully Satisfied
Rationale	In the second iteration, a 1 Terabyte capacity NVME SSD was installed in the Jetson Xavier, providing more than enough storage capacity for ample data capture in the field.

ELEC FR 4	**Sufficient Memory**
Status	Fully Satisfied
Rationale	In this iteration, random crashes were mitigated and the rig succeeded in multiple test cases where it was left capturing for over 10 minutes.

3.4.4 Third Iteration

To improve durability and finalise the electronic design, the third iteration made use of a PCB design for the electronic subsystem. This greatly reduced the occurrence of loose connections and increased the robustness of the rig overall - an essential upgrade for the final data capture expedition where the rig would need to survive periods of high vibration.

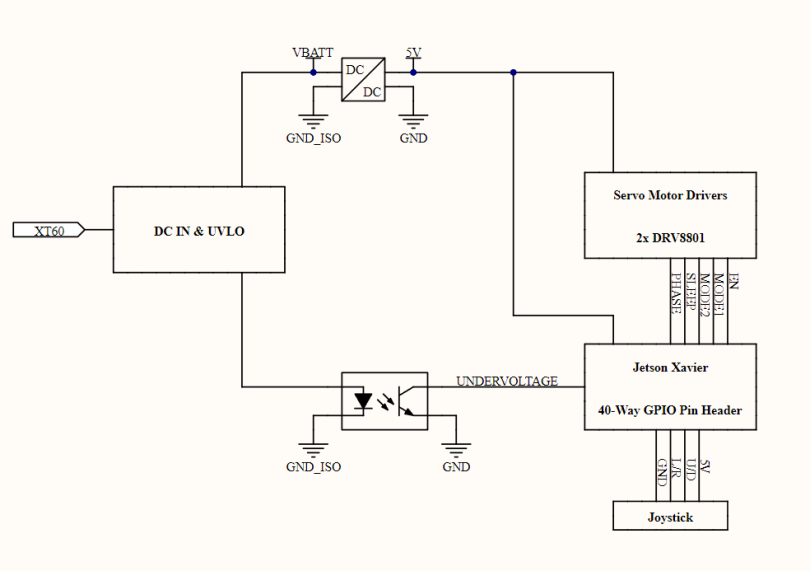

Figure 3.9: A high-level circuit schematic showing the overall electronic design for the third iteration.

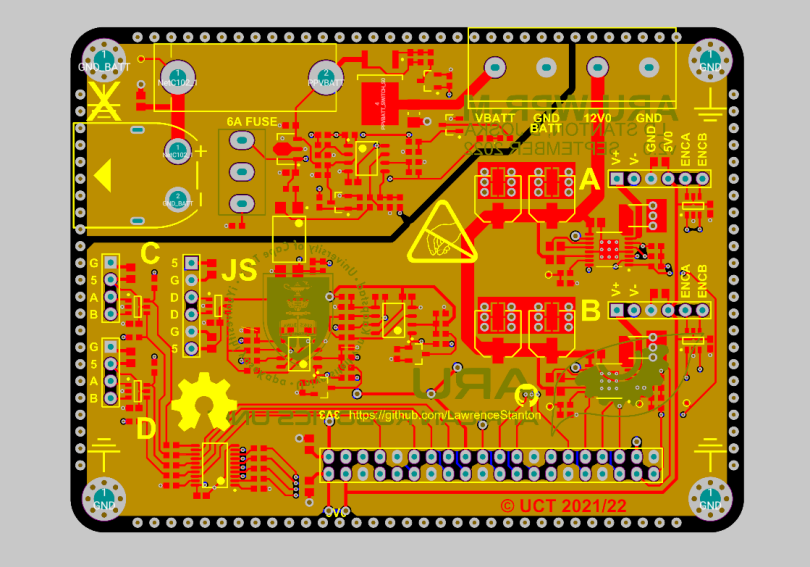

Figure 3.10: A CAD model of the PCB for the third iteration of the rig.

Functional Requirements Met

ELEC FR 1	**Durability**
Status	Fully Satisfied
Rationale	During the final data capture expedition to Kgalagadi National Park, the rig's electronics were able to survive multiple full days of high-vibration off-road travel without the need for major maintenance.

3.4.5 Subsystem-Level Technical Specifications

As with the mechanical subsystem, the electronic subsystem technical specifications pertain to the third and final iteration of the data capture rig. These specifications are outlined below.

ELEC TS 1	**Durability**
Description	To achieve maximum durability and minimal chance of electrical breakages or required maintenance, the circuitry of the rig is contained in a PCB with through-hole soldered components.
Refines	ELEC FR 1

Table 3.23: A table outlining subsystem-level technical specification 1.

ELEC TS 2	**Focus and Zoom Control**
Description	In accordance with the mechanical technical specifications requiring a motor and joystick, a motor driver IC was selected, and supporting circuitry was included on the PCB.
Refines	ELEC FR 2
Component Choice/s	ELEC C1

Table 3.24: A table outlining subsystem-level technical specification 2.

As before, the electrical components referred to in the above technical specifications are outlined in more detail below.

ELEC TS 3	**Rotary Encoders**
Description	To provide more data for both testing purposes and for cross-validation for the reconstructed camera pose from the IMU, rotary encoders were selected to measure the pan and tilt angles.
Refines	ELEC FR 2
Component Choice/s	ELEC C2

Table 3.25: A table outlining subsystem-level technical specification 3.

ELEC TS 4	**Microcontroller Selection**
Description	Given the processing power needed to encode video as well as synchronised lidar data, as well as the memory and storage requirements, an NVIDIA Jetson Xavier AGX was selected as the SoC for the rig.
Refines	ELEC FR 3, ELEC FR 4
Component Choice/s	ELEC C3

Table 3.26: A table outlining subsystem-level technical specification 4.

ELEC TS 5	**Sensor Selection**
Description	For the basic functioning of the rig, an RGB camera and lidar device with an in-built IMU were selected to provide the needed RGB and lidar sensor data.
Component Choice/s	ELEC C4, ELEC C5

Table 3.27: A table outlining subsystem-level technical specification 5.

ELEC C1	**DRV8801 H-Bridge Motor Driver**
Rationale	The DRV8801 motor driver was selected for its compatibility in terms of voltage range, cost-effectiveness, and robustness. Two of these chips were used to drive the motors in the rig.
Model/Part Number	DRV8801
Manufacturer	Texas Instruments
Specifications	
Max Supply Voltage (DC)	40V
Output Current	2.8A
Sense Voltage	500 mV

ELEC C2	**Optical Encoder 1000PPR 5-24V**
Rationale	Two of these optical encoders were selected to provide measurements for the pan and tilt angles. They were chosen for their low cost, high resolution, and wide input voltage range.
Model/Part Number	GTS06-OC-RA1000B-2M
Manufacturer	Generic
Specifications	
Resolution	1000 PPR
Supply Voltage	5-24V DC
Output Type	NPN Open Collector
Allowable Revolution	3000 RPM
Max Response Frequency	100 kHz

ELEC C3	**NVIDIA Jetson Xavier AGX**
Rationale	The Jetson Xavier AGX was selected as a more powerful alternative to the cheaper, more popular Jetson Nano. We require the Xavier's beefier specs for video encoding and the live processing and synchronisation that must be performed on the incoming lidar and RGB data. A 1TB NVME SSD was installed for greater storage.
Model/Part Number	NVIDIA Jetson AGX Xavier
Manufacturer	NVIDIA
Specifications	
GPU	512-Core Volta GPU w/Tensor Cores
CPU	8-Core ARM V8.2 64-bit CPU
Memory	16GB LPDDR4x
Storage	32GB eMMC + 1Tb NVME SSD

ELEC C4	**Ximea XiQ MQ022CG-CM**
Rationale	An RGB camera from the Ximea XiQ series was selected as the RGB sensor for the rig. This component was chosen for its small form factor and mature software toolbox, providing many options for interfacing with the camera in software. A generic 5mm - 150mm focal length lens was attached.
Model/Part Number	XiQ MQ022CG-CM
Manufacturer	Ximea
Specifications	
Resolution	2.2 MP
Max Frame Rate	170
FOV Range	12.88° - 147.1°
Sensor Type	CMOS RGB Bayer Matrix

ELEC C5	**Livox Tele-15 Lidar**
Rationale	The Livox Tele-15 was selected as a low-cost narrow FOV lidar with an onboard IMU. The narrow FOV, but far range for the cost suits our use case, as we mainly track a small subject and do not care much about the surroundings or landscape.
Model/Part Number	Livox Tele-15 Consumer Lidar
Manufacturer	Livox
Specifications	
Scanning Pattern	Non-Repeating
Scanning Rate	10 Hz
Point Density	24000
FOV	14.5° x 16.2°
Range	500m
Angular Precision	¡0.03°
IMU Sample Rate	200 Hz

3.5 Software Subsystem Design

This section will outline the software design of the WildPose rig. Note that the software described in this section covers only the embedded software design used for the data capture rig itself to control and coordinate data capture. It does not include descriptions of the software design behind the 3D reconstruction algorithm, which will be presented in a future chapter.

All data capture rig source code is publicly available and may be obtained from the following GitHub repository:
`https://github.com/African-Robotics-Unit/wildpose/tree/main`

3.5.1 Subsystem-Level Functional Requirements

SOFT FR 1	**Streaming Capability**
Description	The rig's programming should allow for video and lidar data streaming to its internal memory, enabling it to capture data for at least 10 minutes continuously without exceeding RAM limitations.
Refines	FR 1

Table 3.28: A table outlining subsystem-level functional requirement 1.

SOFT FR 2	**Optimal Frame Specifications**
Description	For the visual component of the recorded data, resolution and frames-per-second count should be balanced enough by the compression algorithm to record the subject at no less than 50 pixels in width per frame.
Refines	FR 3

Table 3.29: A table outlining subsystem-level functional requirement 2.

SOFT FR 3	**Graphical User Interface**
Description	The data capture rig should make use of a simplified GUI to control the functions of the rig to allow for ease of use.
Refines	FR 6

Table 3.30: A table outlining subsystem-level functional requirement 3.

3.5.2 First Prototype

The first version of the data capture rig made use of a custom-built C++ program to fetch and coordinate data from the lidar, camera, and rotary encoders. The software made minimal use of external libraries and was controlled via the command line.

A simplified diagram showing the overall flow of data for the first iteration's software design is shown in Figure 3.11.

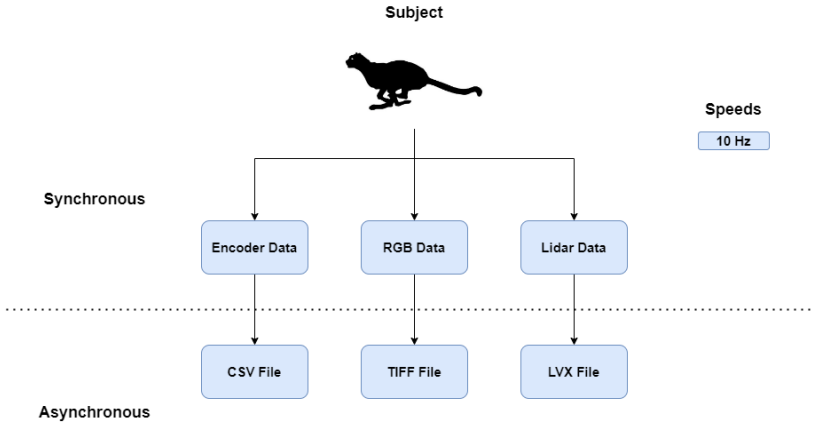

Figure 3.11: A flow diagram illustrating the software pipeline for the first iteration of the data capture rig.

Functional Requirements Met

SOFT FR 2	**Optimal Frame Specifications**
Status	Partially Fulfilled
Rationale	During early field tests, images were captured at 1080p - providing ample resolution to show subject features. However, the FPS count was low as the program was poorly optimised.

3.5.3 Second Iteration

The second iteration of the rig made use of the QT ecosystem including its core UI libraries, IDE, and internal object and thread management classes.

Custom functionality to interface with the GPIO pins was ported over from the first iteration and integrated into the QT application.

A simplified diagram showing the overall flow of data for the second iteration's software design is shown in Figure 3.12.

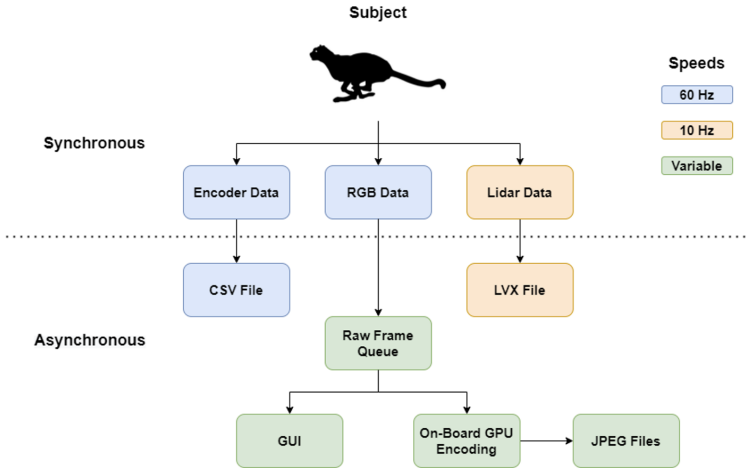

Figure 3.12: A flow diagram illustrating the software pipeline for the second iteration of the data capture rig.

Functional Requirements Met

SOFT FR 1	**Streaming Capability**
Status	Partially Fulfilled
Rationale	In the second iteration, H254 video streaming was utilised and the rig was able to record continuously for ample time. However, the rig was prone to random crashes causing the loss of data.

SOFT FR 2	**Optimal Frame Specifications**
Status	Fully Satisfied
Rationale	Frames were streamed to internal memory at 1080p and 60 FPS - providing rich visual information throughout recorded movements.

SOFT FR 3	**Graphical User Interface**
Status	Partially Fulfilled
Rationale	The second iteration made use of a QT-based GUI - however, on the small touch screen, controls were clunky and the screen was far too busy with unneeded parameter sliders.

3.5.4 Third Iteration

During the design process of the third iteration of the rig, it was decided that the software would be migrated to ROS. This was due to a few main reasons:

1. ROS contains in-built support and pre-written modules for several functionalities needed for the final rig design.
2. Much of the custom functionality already written for the previous iterations was developed in C++ and would be easily ported to a ROS program.
3. Code extendability and readability would be greatly improved by making use of the ROS ecosystem.

Therefore, the final released software design is implemented in ROS 1 on the Melodic distribution.

A simplified diagram showing the overall flow of data for the third iteration's software design is shown in Figure 3.13.

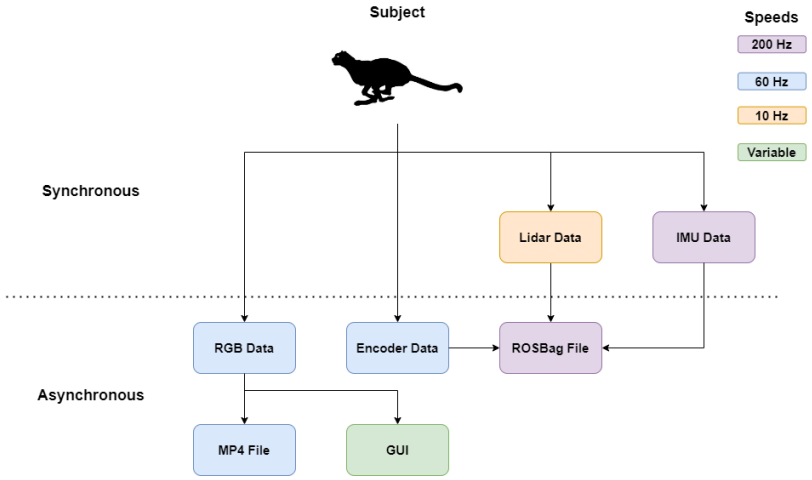

Figure 3.13: A flow diagram illustrating the software pipeline for the third iteration of the data capture rig.

Functional Requirements Met

SOFT FR 1	**Streaming Capability**
Status	Fully Satisfied
Rationale	In the third iteration, the data capture rig utilised ROS video streaming libraries and random crashes were eliminated, resulting in far more stability.

SOFT FR 3	**Graphical User Interface**
Status	Fully Satisfied
Rationale	The third iteration drastically reduced the number of controls on screen and filled it with the viewfinder - resulting in greater simplicity and ease of use.

3.5.5 Subsystem-Level Technical Specifications

Where relevant, the technical specifications outlined below describe various design choices that were made in the development of the software for the third iteration of the rig.

SOFT TS 1	**ROS and C++ Platform**
Description	To make use of the mature libraries of the ROS ecosystem, along with the speed and low-level access to meory of the C++ language, it was chosen that the final rig would make use of a custom-built C++ ROS program for all functionality.
Refines	SOFT FR 1, SOFT FR 2, SOFT FR 3

Table 3.31: A table outlining subsystem-level technical specification 1.

SOFT TS 2	**Video Encoding**
Description	To meet our requirements for high-speed video encoding and simultaneous writing to storage, we elected to make use of the H264 codec along with NVidia's NVEncoder library to make use of hardware acceleration.
Refines	SOFT FR 1

Table 3.32: A table outlining subsystem-level technical specification 2.

Chapter 4

Pose Reconstruction

This chapter describes the software design behind the 3D reconstruction of pose data obtained from the WildPose data capture rig. It will outline the process used to calibrate between the lidar and RGB datasets, as well as the trajectory optimisation-based method used to calculate 3D pose in the world frame.

4.1 Calibration

Since both latter versions of the WildPose rig used in the field involved a variable focal length telephoto lens, the calibration process is not straightforward. Typically a fixed focal length lens is used and both the intrinsic and extrinsic calibrations may be performed before deployment in the field. However, since both the zoom and focus may change dynamically, we require a method to calibrate on a per-recording basis; occasionally, it may even be necessary to re-calibrate mid-recording.

4.1.1 Dataset

First, it is pertinent to describe the selection of a calibration dataset such that our method may be validated and compared to state-of-the-art calibration methods. The dataset consists of a sampling of in-the-wild calibration data at various distances - between 5 and 100 metres between the camera and calibration target. This is representative of the distances at which the rig is typically situated in the wild.

The set of images collected at each distance contain varied checkerboard poses. The aim of each session of calibration data capture was to achieve maximum coverage of the checkerboard in terms of both absolute position in frame and relative angle. Using this strategy, we may obtain as fair an evaluation as possible when compared to existing methods.

4.1.2 Approach

As mentioned, the camera intrinsic calibration, as well as the lidar-camera calibration, needed to be determined on a per-recording basis due to the use of a telephoto lens. In the case of a non-varying focal length, intrinsic and extrinsic calibration may be performed prior to a data capture expedition as the parameters will not change. For this application, the use of a calibration checkerboard - or more modern extensions of this setup such as AruCo markers [42] [43] - is common and widely studied. These methods allow for automatic detection of object and image points. The intrinsic and extrinsic transformations may both then be optimised for using this set of information.

With a varying focal length problem, the approach is not so simple. In our case, lidar-camera calibration was achieved through the use of manually chosen image point to object point correspondences in collected lidar and RGB data. Essentially, the user manually selects distinct feature points in both the image frame and the point cloud, and an optimisation is performed to minimise the reprojection error between the two and obtain an accurate

transformation matrix.

Our implementation makes use of the OpenCV Camera Calibration toolbox. This toolbox allows for the construction of a custom set of object and image points, which may then be passed as inputs to the optimisation function to produce a set of intrinsic camera parameters as well as the extrinsic rotation and translation vectors. Recall that the camera matrix and extrinsic parameters translate a set of object points in the form (x, y, z) to a set of corresponding image points in the form (u, v) as per the equation below.

$$\begin{bmatrix} \mathrm{u} \\ \mathrm{v} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathrm{f_x} & 0 & \mathrm{p_x} \\ 0 & \mathrm{f_y} & \mathrm{p_y} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathrm{r_{00}} & \mathrm{r_{01}} & \mathrm{r_{02}} & \mathrm{t_0} \\ \mathrm{r_{10}} & \mathrm{r_{11}} & \mathrm{r_{12}} & \mathrm{t_1} \\ \mathrm{r_{20}} & \mathrm{r_{21}} & \mathrm{r_{22}} & \mathrm{t_2} \end{bmatrix} \cdot \begin{bmatrix} \mathrm{X} \\ \mathrm{Y} \\ \mathrm{Z} \\ 1 \end{bmatrix} \tag{4.1}$$

By supplying OpenCV's calibration function with appropriate image points and their matching object points, we may optimise for all extrinsic and intrinsic parameters simultaneously as would be done with a typical calibration problem. It should be noted, however, that the object points in this case do not relate to real-world coordinates as they ordinarily do. Since the outcome of this process is an intrinsic and extrinsic calibration between the camera and the lidar, our object points consist of selected points in the lidar point cloud and are relative to the rig frame. Regardless, the underlying mathematics do not change, and the subsequent translation from the rig frame to the world frame is further discussed later in this section.

With these tools at our disposal, all we require is a method to select corresponding image point and object points pairs. We achieve this through the use of a simple GUI program implemented in Python and making use of Open3D's point cloud handling library. With this program, the user is presented with an image and is able to select point coordinates in accordance with any consistent visual indicators they choose. These visual indicators need to be features that are easily recognisable in both an image

Figure 4.1: Screenshots showing a human-labelled set of corresponding image and object points for a calibration target.

and a point cloud - for example, the edges and corners of a checkerboard or the extremities of an animal such as tail tips and paws. The user is then presented with a 3D view of a corresponding point cloud, and may select points in the 3D space once again based on the same visual indicators. From these selected points, the image and object point matrices are built and passed to OpenCV's calibration function where the final output is obtained. An example of a hand-selected corresponding set of image and world points is shown in Figure 4.1, where a calibration checkerboard was chosen as the target.

A flow diagram illustrating the general process undertaken when calibrating the system using our method is shown in Figure 4.2.

All code for the RGB-lidar calibration tool is publicly available and may be obtained at the following GitHub repository link:
`https://github.com/DJoska/pcd_cam_calibrator`.

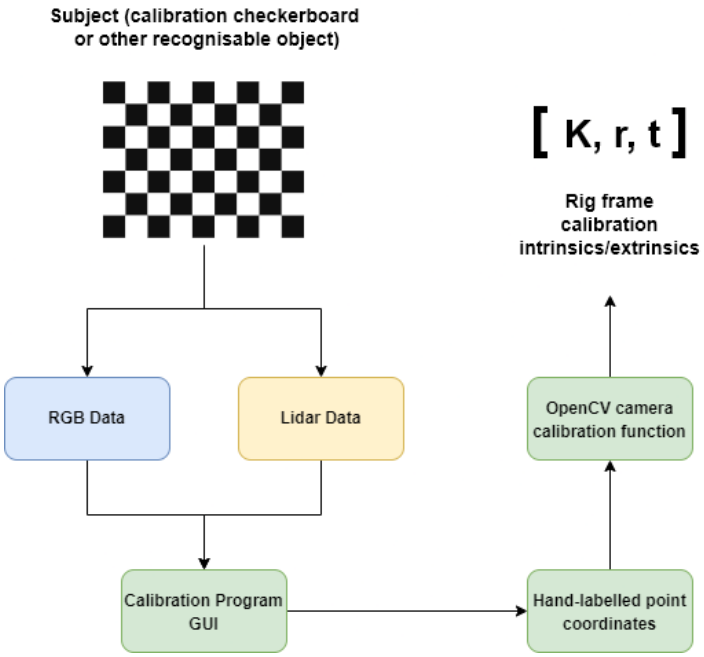

Figure 4.2: A flow diagram showing the overall flow of data through the calibration method.

4.2 Reconstruction

Once a suitable extrinsic calibration has been achieved for a particular filmed movement using the aforementioned technique, the calibration matrices are passed on as inputs to the reconstruction algorithm along with the raw visual and lidar data. During this stage, a handful of algorithms with various purposes are run before the final 3D pose estimates may be obtained. Here, each step in the process will be described and previously explained theory and mathematics may be developed further where necessary.

4.2.1 Dataset

The first aim of the pose reconstruction portion of this project is the development of a cheetah locomotion dataset. This dataset contains high-speed RGB footage of cheetahs in the wild, along with corresponding

synchronised point cloud sequences recorded via lidar. The dataset as of the writing of this thesis is comprised of two main parts largely obtained on two separate field expeditions: the **Hartbeespoort Dataset**, and the **Kgalagadi Dataset**. Sequences from both datasets are used in this document to present qualitative and quantitative results.

Hartbeespoort Dataset

The Hartbeespoort dataset was built during a data collection expedition to the Anne Van Dyk Cheetah Centre in Hartbeespoort, North West Province. The data was collected over the course of 5 days, during each of which between 1 and 3 cheetahs were filmed during their exercise runs. These runs consisted of a lure placed on a motorised track, which would be activated once a cheetah is released into the exercise area. The lure would then be pulled around the track in a predetermined route at high speeds, causing the cheetah to chase it instinctively. The data capture rig was used to film each cheetah run and gather lidar and RGB data to add to the dataset.

This portion of the dataset made use of the second iteration of the data capture rig.

Kgalagadi Dataset

The Kgalagadi Dataset was built during an expedition to Kgalagadi Transfrontier Park in the Southern Kalahari, which spans across sections of Botswana and South Africa. This data was collected over a 4 day span, during which the rig (affixed to the door of a vehicle) was driven around the park and used to record any animals of interest encountered. The main animal of interest was, of course, the cheetah, and several individuals were recorded performing various types of motion. A couple interesting segments of motion are presented in the following sections of this document.

To collect the Kgalagadi data, the third version of the WildPose rig was

used, which was the only version that was designed to be mounted on a car door.

Table 4.1 shows a summary of various differences between each of the datasets.

	Img Resolution	Rig Version	Lidar Format	IMU
Hartbeespoort	2048x1088	2	LVX File	No
Kgalagadi	852x480	3	ROSBag File	Yes

Table 4.1: A table outlining differences between the two datasets

Synchronisation

The Hartbeespoort dataset made use of a software trigger in the data acquisition loop, upon which both an RGB frame and - every 6 frames - a lidar point cloud were acquired. The RGB frame and point cloud were then saved directly to internal storage with corresponding indices. This method of acquisition results in closely synced data from the time of exposure. Minor synchronisation error only occurs in situations where the subject is moving at high speeds and the relatively long exposure of the lidar results in motion blur within the point clouds. However, this situation is unavoidable barring a costly upgrade to high-speed lidar devices, and such accuracy is unneeded once the process reaches the trajectory optimisation stage since far greater sources of uncertainty are introduced.

The Kgalagadi dataset was synchronised post-recording. This was done through the identification of a pivot point with recognisable positional data in both the point cloud and visual image. Generally, this was chosen as the point where the subject's paw made contact with the ground, as the frame containing initial contact is easily visually identified. A similar method was used to synchronise multi-angle frames in AcinoSet [10] - there, a bright LED light was switched on and the frame where light was first visually detected was chosen as the pivot point. Once the pivot point is identified in a corresponding RGB-lidar pair, the sequence may be synchronised around this point.

One possible source of error in this method is frame drift; however, as before, greater sources of uncertainty will be introduced downstream. The effect of frame drift is also mitigated by choosing a pivot point present near the centre of the examined sequence and keeping the window of inspection short, which often tends to be the case during the analysis of cheetahs.

This also helps mitigate the effect of dropped frames - an unfortunate but common occurrence with the use of ROS video packages at high frame rates. In cases where frame loss occurs, a uniform loss is assumed and the timeline for synchronisation is adjusted by multiplying each frame's timestamp in seconds by a factor δ_{loss}, defined as follows:

$$\delta_{\text{loss}} = \frac{Total\ Expected\ Frames}{Total\ Recorded\ Frames} \tag{4.2}$$

Thus, the ratio of visual information to lidar/depth information is changed by this factor; however, the real-world speed of the subject is preserved in the reconstruction since the lidar point clouds remain unaffected. In this way, we are able to recover pose even from datasets where the video files have dropped frames. Again, this slight loss of infromation density in the reconstrcution is generally eclipsed by larger sources of error downstream.

4.2.2 Reconstruction Overview

A high-level overview of the flow of data throughout the pose reconstruction process is shown via the flow diagram in Figure 4.3. To summarise the steps of the reconstruction process:

- DeepLabCut is used to generate 2D keypoint estimates for each RGB frame.
- The set of RGB images, 2D keypoint estimates, and lidar point clouds are read by the program and dataframes of the 2D and 3D measurements are built.

- A sparse bundle adjustment is performed on the set of 2D measurements, yielding 3D estimates that may vary along epipolar lines.
- Pseudo-RGBD measurements are built using the point clouds and SBA estimates.
- The full trajectory estimation is performed using the pseudo-RGBD estimates and a corresponding piecewise cost function.
- World frame coordinate transformation is performed on the output of the FTE stage.

Each of these stages will be elaborated upon below.

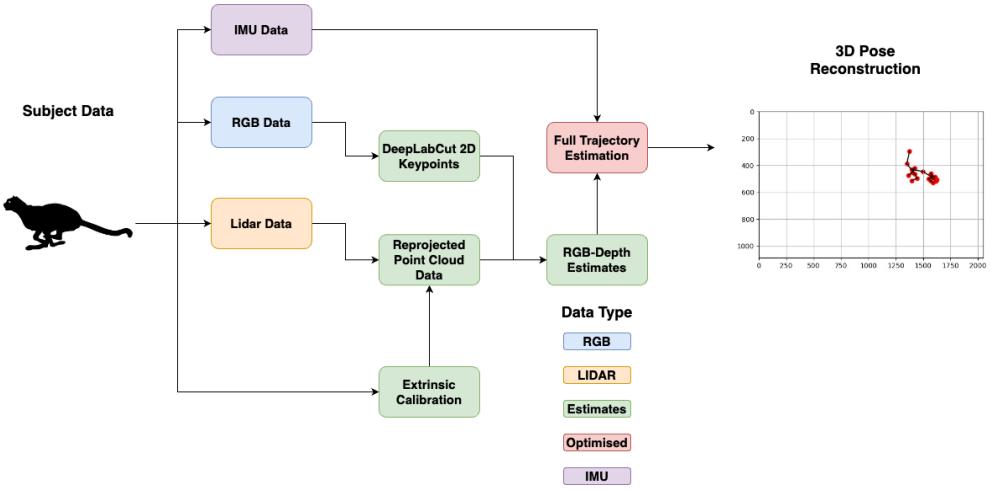

Figure 4.3: A flow diagram showing the basic functionality of the wildpose data capture rig software

4.2.3 Sparse Bundle Adjustment

The first stage in the pose reconstruction process is the sparse bundle adjustment, or SBA. This step is used to provide a naive estimate for the 3D position of the subject throughout a trajectory to serve as initialisation for the later steps.

The theoretical backing for the use of sparse bundle adjustment has been outlined in a previous section. Here, the specific implementation in the pose reconstruction process will be explained.

Since we only make use of one camera view, we do not expect accurate esimates from the SBA. However, by finding an optimal set of 3D points to match a set of 2D keypoint estimates provided by a DeepLabCut neural network, we are able to drastically reduce time taken to reach an optimal result during the FTE stage and end up with a more accurate final result.

Given a 2D point estimate p, the SBA was constructed so as to minimise the reprojection error e when an estimated corresponding 3D point, P, is translated using some function F. Denoting the point index as i, the reprojection of a 3D point to a 2D point may be summarised in the equation below.

$$\hat{\mathrm{p}}_\mathrm{i} = \mathrm{F}_\mathrm{i}(\mathbf{P}_\mathrm{i}) \tag{4.3}$$

Note that this estimate is inaccurate in 3D space as performing an SBA from a single view yields a result that varies along epipolar lines. Reprojected points produced by this method may appear accurate in the frame, but depth remains an unknown.

The reprojection error is simply calculated as:

$$\mathrm{e}_\mathrm{i} = \hat{\mathrm{p}}_\mathrm{i} - \mathrm{p}_\mathrm{i} \tag{4.4}$$

The error is then minimised through a Cauchy redescending robust cost function as used in the original AcinoSet paper [10]. This specific cost function was defined piecewise and chosen to improve outlier rejection in the neural network estimates. The below equation summarises the cost function pulled from the original work.

$$C(e) = \begin{cases} \frac{e^2}{2} & \text{for } |e| < a \\ a|e| - \frac{a^2}{2} & \text{for } a \leq |e| < b \\ ab - \frac{a^2}{2} + \frac{a(c-b)^2}{2}(1 - (\frac{c-|e|}{c-b})^2) & \text{for } b \leq |e| < c \\ ab - \frac{a^2}{2} + \frac{a(c-b)^2}{2} & \text{for } c \leq |e| \end{cases} \tag{4.5}$$

Once the redescending cost function is minimised, the resulting set of 3D points are then used to construct a set of pseudo-RGBD estimates. This is done by reprojecting both the 3D estimate and points from the point cloud to the image plane. A nearest-neighbour lookup is used on the reprojected 2D coordinates to create associations between the lidar points and the 3D estimates. This is done for each point by simply finding the closest reprojected lidar point in the frame for each reprojected SBA point - that is, the point that results in the lowest 2D Euclidean distance magnitude. We now have an associated 3D lidar point for each naive 3D SBA estimate - allowing us to infer depth. Next, a Euclidean distance is calculated from the camera origin to the lidar point for each estimate. This is used as the depth reading for each point, which along with the reprojected x- and y-coordinates from the SBA output form the set of pseudo-RGBD estimates. These RGB-D estimates are used in the FTE stage to calculate a depth error, which is discussed further below. The process of creating these RGBD estimates is summarised in Figure 4.4.

Since we have now built a depth correspondence for each SBA point, the RGB-D estimates are projected back to the 3D coordinate system (x,y,z) to obtain initialisation points for the FTE stage. We have now associated a depth with the naive SBA outputs, resulting in a much more accurate initialisation and subsequently a faster and more accurate optimization.

4.2.4 Full Trajectory Estimation

Once an SBA has been performed and an initial state is estimated, the full trajectory optimisation or FTE performs the majority of the work in

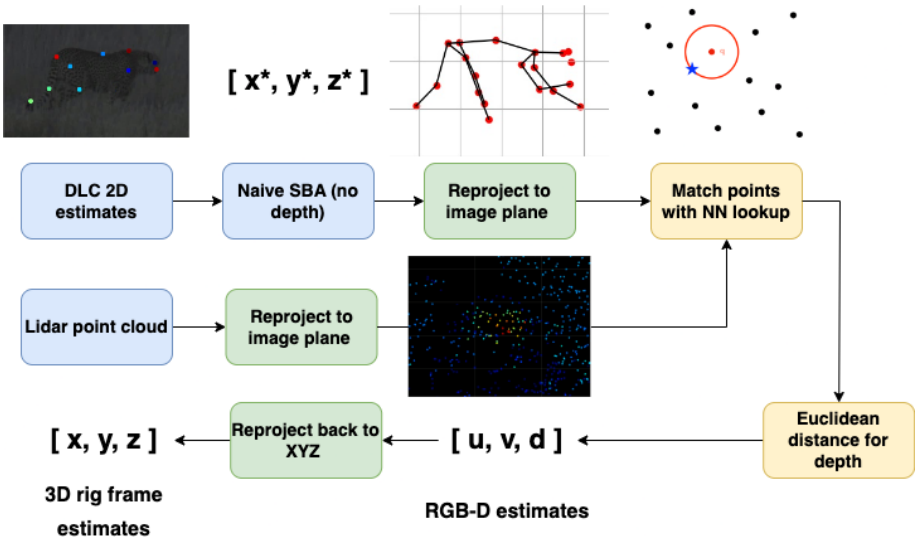

Figure 4.4: A flow diagram showing the overall process of creating the pseudo-RGBD estimates.

producing relative pose estimates. The 3D pose estimates calculated by this method are given in the rig frame - that is, they are measured relative to the movable rig platform. Conversion of these estimates to the world frame is then performed post-optimization.

The FTE method once again follows the method established in AcinoSet [10] relatively closely. Some adjustments to the algorithm are made to account for the specific nature of the RGB and lidar data of WildPose, as opposed to the multi-view synchronised RGB data used in the former method.

The FTE model comprises of two main aspects: *constraints* and the *cost function*. There are two main types of constraints imposed on the trajectory estimation module, the first of which is the *model constraint*. This constraint eliminates physically impossible poses based on a pre-defined pose model for cheetah morphology. A constant acceleration model was assumed to further reduce variance. To account for possible variations in this model that occur in reality, a slack variable w was added to represent model noise. Using implicit Euler integration, a set of equations describing the state is thus presented below:

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \Delta t \dot{\mathbf{x}}_k \quad \text{for} \quad k = \{2, \ldots, N\} \tag{4.6}$$

$$\dot{\mathbf{x}}_k = \dot{\mathbf{x}}_{k-1} + \Delta t \ddot{\mathbf{x}}_k \quad \text{for} \quad k = \{2, \ldots, N\} \tag{4.7}$$

$$\ddot{\mathbf{x}}_k = \ddot{\mathbf{x}}_{k-1} + \mathbf{w}_k \quad \text{for} \quad k = \{2, \ldots, N\}. \tag{4.8}$$

Next, a *measurement constraint* is imposed. This constraint ensures that a reconstructed 3D pose matches the visual 2D pose as closely as possible during optimisation. To parameterise this constraint, the set of 3D positions s_i of each marker are obtained from the state x_i through a simple mathematical function F:

$$\mathbf{s}_i = F(\mathbf{x}_i) \tag{4.9}$$

Following this, the 3D points are reprojected to the image plane and compared to the 2D neural network estimates m whilst including a second slack variable, v, to account for measurement noise:

$$\mathbf{m}_{i,j} = F_j(\mathbf{s}_{i,j}) + \mathbf{v}_{i,j} \tag{4.10}$$

Next, we construct the cost function. As per the original method, the first part of the cost function is the measurement error e_{meas}. Since we only have one camera, this is constructed as follows:

$$e_{meas} = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{l} C\left(\frac{\mathbf{v}_{i,j,k}}{\sigma_{meas}}\right). \tag{4.11}$$

Then, the model error e_{model} is constructed normally by totalling the squares of the acceleration noise:

$$e_{model} = \sum_{i=1}^{n}\sum_{j=1}^{p} \frac{\mathbf{w}_{i,j}^2}{\sigma_{modelj}^2} = \sum_{i=1}^{n}\sum_{j=1}^{p} \left(\frac{\mathbf{w}_{i,j}}{\sigma_{modelj}}\right)^2. \tag{4.12}$$

Finally, an extra component to the cost function is added to incorporate the lidar measurements. This is the depth error e_{depth}, and is calculated based on the pseudo-RGBD measurements discussed previously. The depth error is then defined as:

$$e_{depth} = \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{l} C\left(\frac{\mathbf{d}_{i,j}}{\sigma_{depth}}\right). \tag{4.13}$$

Where σ_{depth} is calculated as the standard deviation in the set of depth values for the RGB-D estimates. These three errors are then simply summed together to construct the overall cost function for the FTE:

$$\min_{\mathbf{x},\,\dot{\mathbf{x}},\,\ddot{\mathbf{x}},\,\mathbf{s},\,\mathbf{w},\,\mathbf{v}} \quad e_{meas} + e_{model} + e_{depth}. \tag{4.14}$$

The trajectory optimisation used in this study was implemented in Python using Pyomo [44]. An IPOPT MUMPS linear solver was used [45].

4.2.5 World Frame Transformation

Once relative pose is reconstructed, the final stage in the process is the world frame coordinate transform. This stage will translate the 3D pose predictions produced by the previous stage from the rig frame to the world frame (also known as the inertial frame).

This transform is done using one of two methods in this work, depending on which dataset is being considered. These methods are **monocular visual odometry** and **IMU incorporation**.

Visual Odometry

In the cases where IMU data is not recorded, a monocular visual odometry technique was used to estimate camera pose relative to the world frame. Once this pose is estimated for each frame of recorded data, the camera pose matrix may be applied to the output of the FTE stage to obtain a set of 3D estimates in the world coordinate system.

Specifically, SURF features were calculated for a given sequence of images in order to estimate camera pose from frame to frame. SURF provides slightly better accuracy and speed than typical SIFT feature extraction [40] and is relatively simple to apply using open-source packages.

IMU Incorporation

In the cases where IMU data is recorded, the sensor readings are simply used to translate the rig frame estimates obtained by the FTE stage to the world frame. It is assumed that these IMU readings are accurate to the degree that any error propagated to the final estimates will be greatly overshadowed by other sources of error in the process, most likely those arising from the FTE or SBA stage.

The rig's platform is assumed not to be accelerating, ie. its linear acceleration is the vector $[0; 0; g]$. This is verified by inspection of the three linear acceleration components of the IMU readings a_x, a_y, and a_z throughout the collected datasets. The important readings are therefore those relating to angular velocity, as the rig is often rotated to track a subject throughout a movement. These three readings - ω_x, ω_y, and ω_z - are taken from each IMU sample and used to construct a rotation matrix for each frame of the reconstruction, thus completing the conversion of the FTE output to the real world frame. Note that the angular velocities here are measured in rad/s.

To obtain per-frame rotation matrices, the angular velocities are integrated

to obtain an angular position for a given point in time corresponding to a reconstructed frame. Since the values are of course discrete, Simpson's rule for numerical integration is applied. This is done for each of the axes as follows:

$$\theta_x = \int_0^t (\omega_x)_t \approx \frac{\Delta t}{3}((\omega_x)_{t_0} + 4(\omega_x)_{t_1} + 2(\omega_x)t_2 + ... + 4(\omega_x)_{t_{n-1}} + (\omega_x)_{t_n}) \tag{4.15}$$

$$\theta_y = \int_0^t (\omega_y)_t \approx \frac{\Delta t}{3}((\omega_y)_{t_0} + 4(\omega_y)_{t_1} + 2(\omega_y)t_2 + ... + 4(\omega_y)_{t_{n-1}} + (\omega_y)_{t_n}) \tag{4.16}$$

$$\theta_z = \int_0^t (\omega_z)_t \approx \frac{\Delta t}{3}((\omega_z)_{t_0} + 4(\omega_z)_{t_1} + 2(\omega_z)t_2 + ... + 4(\omega_z)_{t_{n-1}} + (\omega_z)_{t_n}) \tag{4.17}$$

Once angular positions for each of the three axes for a given frame have been calculated numerically, the rotation matrices are simply determined using the basic x, y, and z axis rotation matrices as below:

$$\mathbf{R}_x(\theta_x) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_x) & \sin(\theta_x) \\ 0 & -\sin(\theta_x) & \cos(\theta_x) \end{bmatrix} \tag{4.18}$$

$$\mathbf{R}_y(\theta_y) = \begin{bmatrix} \cos(\theta_y) & 0 & \sin(\theta_y) \\ 0 & 0 & 0 \\ -\sin(\theta_y) & 0 & \cos(\theta_y) \end{bmatrix} \tag{4.19}$$

$$\mathbf{R}_z(\theta_z) = \begin{bmatrix} \cos(\theta_z) & \sin(\theta_z) & 0 \\ -\sin(\theta_z) & \cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.20}$$

Finally, these rotation matrices are applied to each rig frame point coordinate vector using basic matrix multiplication:

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \mathbf{R}(\theta) \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{4.21}$$

Chapter 5

Results

This chapter will describe and discuss the results obtained during each phase of the project. It will provide both qualitative and quantitative experimental validation on the previously outlined datasets, as well as discuss the impact and implications of these results when compared to current state-of-the-art methods.

5.1 Calibration

First, our lidar-camera calibration method requires experimental validation. Various calibration methods were tried during the course of this study, and most proved ineffective for our specific use case. This is likely due to a few main reasons:

- Many existing methods are not tested on in-the-wild calibration data, and tend to be optimised for lab environments with static backgrounds.
- To eliminate labour and tedium - which is an understandable goal - most modern methods are automatic. However, our datasets are

tricky vision problems in and of themselves and automatic methods do not work satisfactorily.

- Our use case is extremely specific, and on real data requires the use of a non-planar subject. Experimental validation on non-planar subjects in the literature is fairly thin, and does not tend to lend itself well to our problem.

These posited reasons for undesirable performance will be further explored in the coming sections.

5.1.1 Qualitative Results

We first present qualitative analysis of the WildPose calibration method consisting of plots depicting reprojection performance among other visual metrics.

Shown below in Figure 5.1 are reprojected 3D ground truth labels for a planar calibration checkerboad rig at a distance of 5 metres. This is a best-case scenario for a calibration dataset, as it is:

1. Captured in a laboratory setting with a static background, ie. not in-the-wild
2. Tested over a relatively short distance, meaning a far denser point cloud at the subject distance
3. Well-framed and highly controlled in terms of positioning of the checkerboard, allowing it to occupy a maximum proportion of space within the frame leading to richer visual data

Performance is, as expected, relatively high when tested over this subset of data. A more challenging example is shown below in Figure 5.2 - this

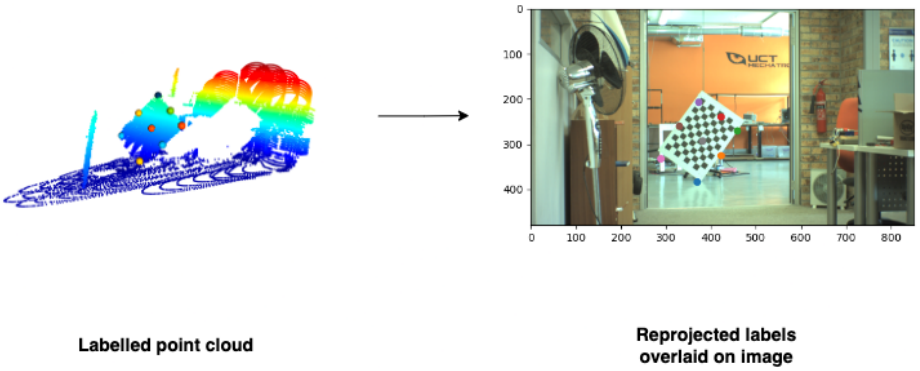

Figure 5.1: Reprojected point cloud labels for a planar rig at a 5m distance.

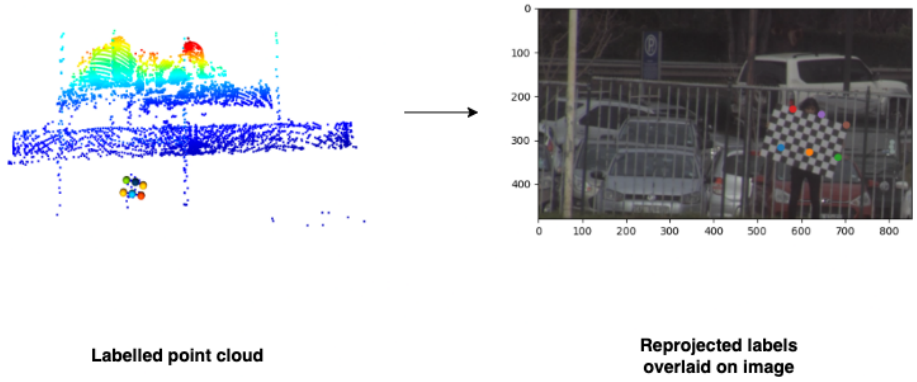

Figure 5.2: Reprojected point cloud labels for a planar rig at a 100m distance.

figure depicts a test conducted over a 100 metre distance against a dynamic "in-the-wild" background.

As can be seen, the point cloud is far sparser and the calibration rig does not occupy as large a proportion of the frame as before. Thus, visual performance is also demonstrably worse; however, considering that this is somewhat of a worst-case scenario for a calibration dataset, the reprojection accuracy is acceptable. This is nonetheless a more realistic depiction of the level of calibration performance when transferred over to a real subject.

5.1.2 Quantitative Results

In this section, we present quantitative evidence of our calibration method's performance across varying subject morphology, as well as a range of distances.

Shown below in Table 5.1 is a comparison of the reprojection RMSE of our method for a planar and a non-planar calibration rig on a challenging multi-distance calibration dataset. In the case of a non-planar calibration

		5m	**25m**	**50m**	**75m**	**100m**
Planar	RMSE	20.39	107.00	104.06	96.20	94.64
	Std Dev	3.26	33.82	43.29	31.78	38.60
Non-Planar	RMSE	30.61	95.44	115.10	114.00	118.42
	Std Dev	3.66	16.33	54.75	5.77	13.39

Table 5.1: Reprojection RMSE and standard deviation in pixels for each calibration method

Shown below in Figure 5.3 are histograms of the reprojection errors for a selected spread of distances across both the planar and non-planar rig configurations.

5.1.3 Discussion

As is demonstrated in the above results, calibration accuracy tends to be far higher at short ranges of around 5 metres. Performance then significantly decreases at the 25 metre range, but remains mostly consistent thereafter up to the maximum tested range of $100m$. Although the drop in accuracy is notable, the consistency in the performance of our calibration method over larger distances is promising. The high accuracy at short distances is largely unimportant for our application, as it is extremely unlikely that a subject captured in the wild will remain within 10 metres of the rig throughout its entire sequence of motion.

The larger distances in our tested calibration dataset are relatively accurate estimations of the method's performance on real subject data, and indeed

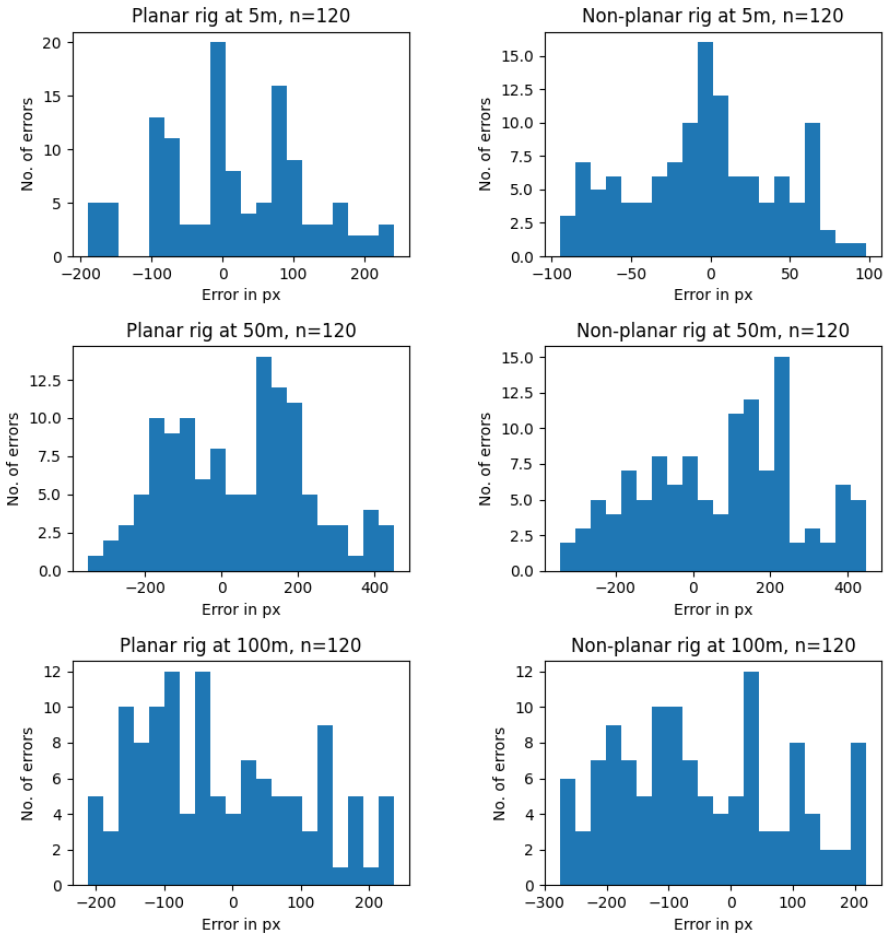

Figure 5.3: Histograms showing the reprojection errors for the planar and non-planar test rigs at 5, 50, and 100 metre distances.

are more challenging in some cases due to the small calibration rig sizes. Examples of real reprojected data will be presented in the subsequent section as further validation of both the calibration accuracy and the entire reconstruction pipeline as a whole.

5.2 Reconstruction

In this section, we will present qualitative and quantitative evaluations of our reconstruction method on real in-the-wild data chosen from our Hartbeespoort and Kgalagadi datasets. Where necessary, elaboration on the method's performance in specific instances will be provided and a further discussion of the implications of these results is included in a later subsection.

In general, each method of evaluation will be performed on sequences pulled from each of the two datasets to provide a complete picture of performance. Note the differences in world frame coordinate transformation between the two datasets outlined in the previous chapter.

5.2.1 Qualitative Results

Hartbeespoort Data

Shown in Figure 5.4 are a selected batch of frames from a reconstructed motion sequence form the Hartbeespoort dataset. In this sequence, the subject is running forwards in a straight line, with minimal occlusion due to dust or foliage. Distance from the subject to the camera plane is approximately **90m**. This is an example of a low-challenge sequence from the Hartbeespoort dataset. We categorise this as low challenge for the following reasons:

1. Motion blur throughout the sequence is minimal.
2. Occlusion is minimal, as there as minimal dust, foliage, and/or haze obscuring any part of the subject.
3. The subject is isolated - there are no other animals present in the frame that may be mistaken for the subject itself by the network.

4. The poses are not complicated and do not change drastically from frame to frame due to the low speed of motion.

Although the distance to the subject can be considered near the higher end for our collected dataset, the higher resolution of the Hartbeespoort images compensate for this and neural network keypoint estimations are high-accuracy in this case.

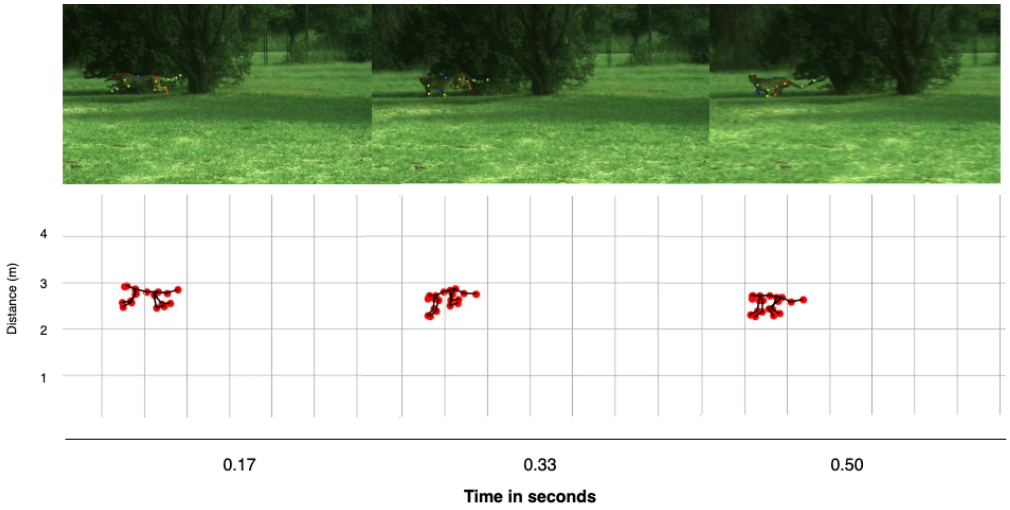

Figure 5.4: Frames with overlaid neural network predictions and corresponding pose reconstructions for a straight running motion from the Hartbeespoort dataset.

Conversely, shown in Figure 5.5 are another selected batch of frames we classify as high-challenge. Here, the subject is performing a direction-changing manoeuvre and uses a flick of its tail to assist in the movement. Distance to the subject here is approximately **80m**. We consider this challenging for our method for a few reasons:

1. Motion blur is higher during this motion due to the higher speed.
2. In most frames, the subject is in a more complicated position due to the direction change - resulting in occlusion due to the subject's own body.
3. Occlusion due to external sources such as dust and foliage is higher in this motion.

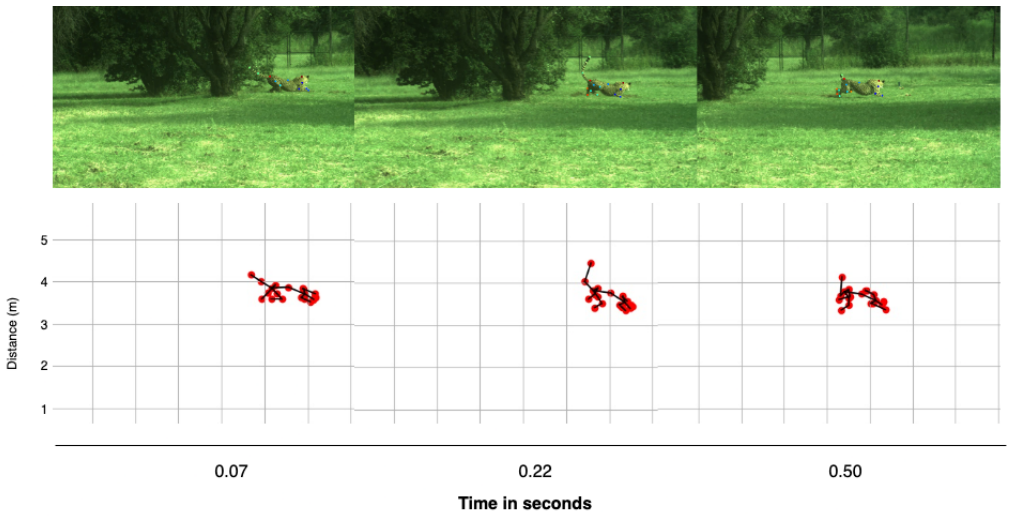

Figure 5.5: Frames with overlaid neural network predictions and corresponding pose reconstructions for a flick motion from the Hartbeespoort dataset.

Kgalagadi Data

Shown in Figure 5.6 are selected frames from a sequence of motion captured during the Kgalagadi expedition. During this sequence, the subject is performing a simple and slow walking motion, and occlusion is minimal - barring some obstruction of the paws which is common. Range from the rig to the subject in this instance is around **50m**. As before, we classify this as a low-challenge sequence for the following reasons:

1. Distance to the subject is near the lower range of focus for the rig.
2. The subject is not obscured by excessive vegetation, dust, or other such sources of occlusion.
3. The subject is isolated - there are no other animals present in the frame that may be mistaken for the subject itself by the network.
4. The poses are not complicated and do not change drastically from frame to frame due to the low speed of motion.

Shown in Figure 5.7 are more selected frames from another sequence of

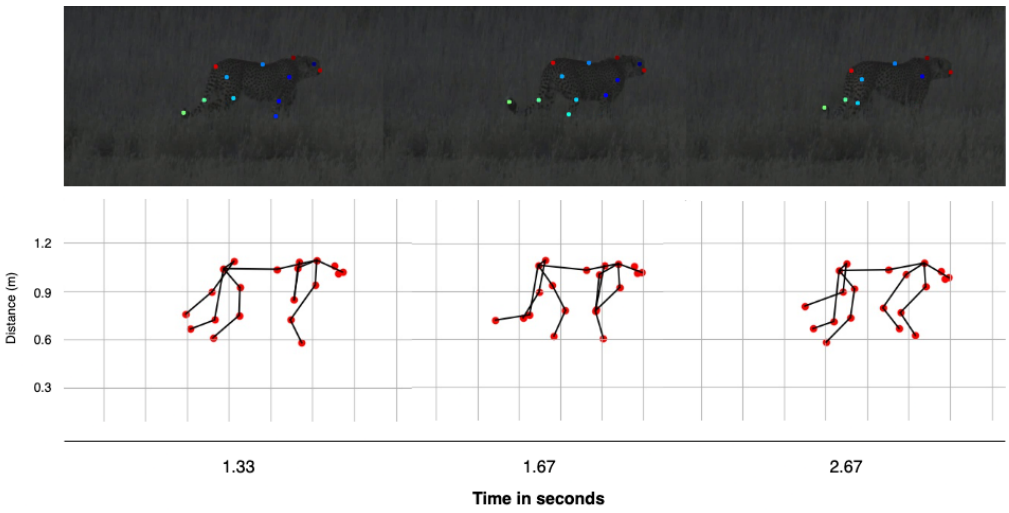

Figure 5.6: Frames with overlaid neural network predictions and corresponding pose reconstructions for a walking motion from the Kgalagadi dataset.

motion from the Kgalagadi - this time, the frames are drawn from a hunt we were fortunate enough to capture on record. Distance to the subject is similar at around **50m**; however, the poses are much more complicated. This sequence represents a high-challenge problem for the network, since:

1. The subject is occluded not only by dust, but by its prey.
2. The subject is not isolated and the network often mistakes the Springbok's limbs for that of the cheetah's.
3. In this sequence, the subject is moving quite fast - meaning large changes in pose from frame to frame, as well as added occlusion.

5.2.2 Quantitative Results

Hartbeespoort Data

Shown in Figure 5.8 are histograms depicting the reprojection errors measured on a labelled subset of the above described "run" and "flick"

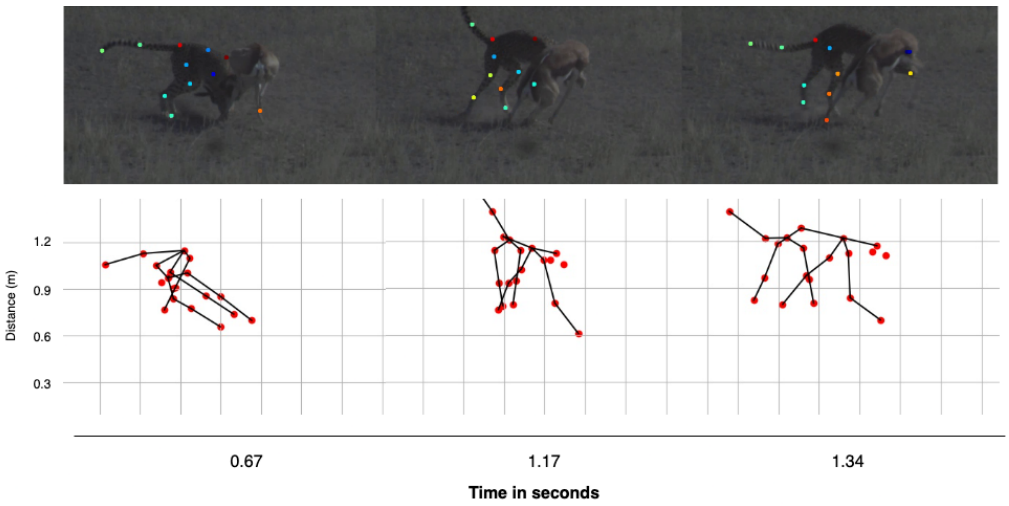

Figure 5.7: Frames with overlaid neural network predictions and corresponding pose reconstructions for a hunting sequence from the Kgalagadi dataset.

sequences. Both signed and absolute reprojection errors are shown to provide a deeper understanding of the distribution of errors for each movement type. Consistent with the qualitative comparison, the hunting sequence evaluation displays a higher number of larger errors (100 pixels and above) than the walking sequence.

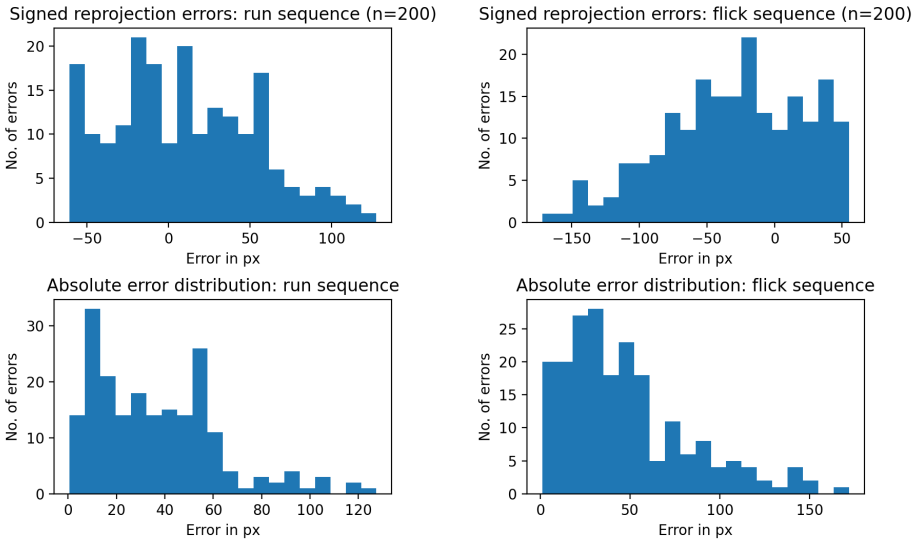

Figure 5.8: Histograms showing signed and absolute reprojection errors for selected labelled sequences from the Hartbeespoort dataset.

Additionally, Table 5.2 shows a numerical summary of the performance of the method on the Hartbeespoort dataset.

	RMSE	**Std Dev**	**Outlier Ratio**
Run	36.48	26.03	0.03
Flick	47.42	35.38	0.10
Overall	42.03	31.60	0.06

Table 5.2: Reprojection RMSE and standard deviation in pixels for the run and flick sequences from the Hartbeespoort dataset

To provide 3D error metrics, a 3D pose adjustment method was used. To obtain ground truth, the output of the FTE was run through the world coordinate transforms as specified above to obtain world frame estimates. Once calculated, these methods were then visually inspected and, if inaccurate, adjusted to a more visually accurate 3D position manually. Reprojections to the image plane were used to further inform accurate pose on the part of the labeller. Using this method, most points need not be adjusted as they are deemed suitably accurate by a human labeller; however, where adjustments are needed, the Euclidean distance of the adjusted point to the estimated point is recorded and presented as a measure of overall 3D accuracy.

Figure 5.9 below depicts histograms of the adjustments performed for the Hartbeespoort data.

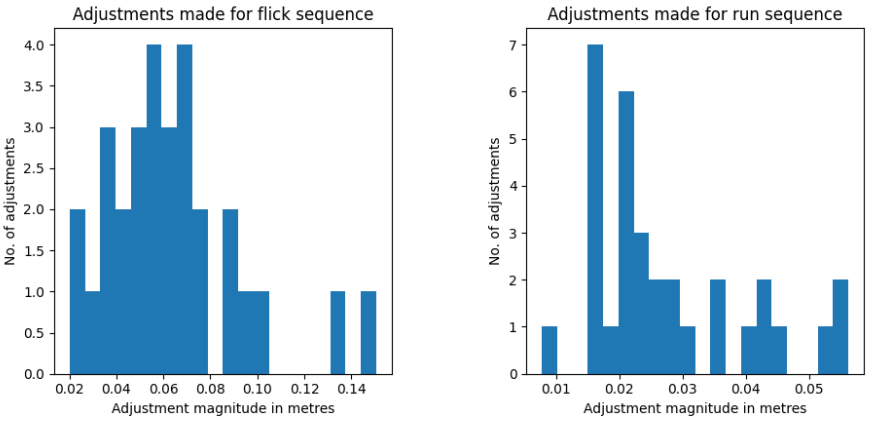

Figure 5.9: Histograms showing adjustment magnitudes for the flick and run sequences from the Hartbeespoort dataset.

Additionally, a numerical summary of the adjustments is presented below in Table 5.3.

	RMSE (cm)	Std Dev (cm)	Percentage Adjusted
Run	2.74	1.23	8.00 %
Flick	6.36	2.90	7.50 %
Overall	4.49	2.85	7.75 %

Table 5.3: Numerical breakdown of 3D adjustment magnitudes on the Hartbeespoort data

Kgalagadi Data

Shown in Figure 5.10 are histograms depicting the reprojection errors measured on a labelled subset of the above described hunting and walking sequences. Consistent with the qualitative comparison, the hunting sequence evaluation displays a higher number of larger errors (50 pixels and above) than the walking sequence. Recall that the resolution of the Kgalagadi images is significantly lower than the Hartbeespoort images (480p vs 1080p).

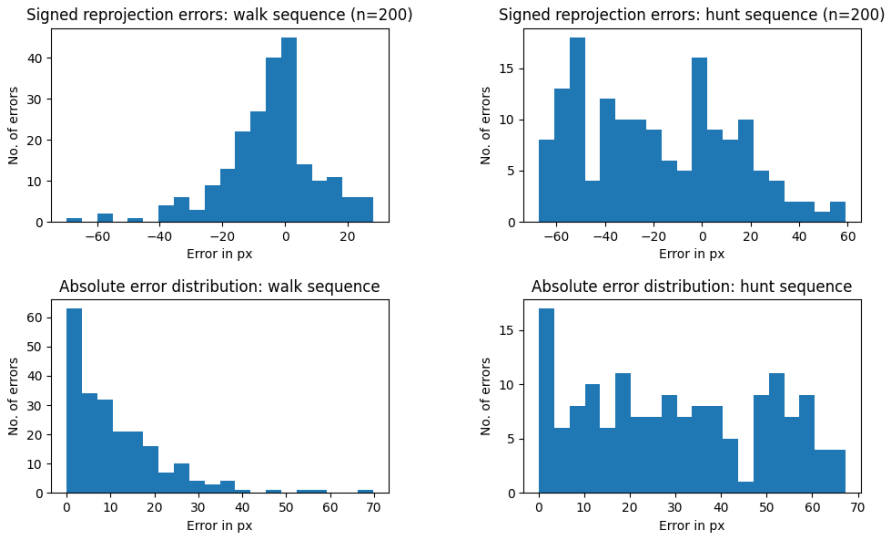

Figure 5.10: Histograms showing signed and absolute reprojection errors for selected labelled sequences from the Kgalagadi dataset.

Additionally, Table 5.4 shows a numerical summary of the performance of the method on the Kgalagadi dataset.

	RMSE	Std Dev	Outlier Ratio
Hunt	30.18	19.80	0.23
Walk	11.42	11.01	0.01
Overall	19.14	17.84	0.10

Table 5.4: Reprojection RMSE and standard deviation in pixels for the hunt and walk sequences from the Kgalagadi dataset

As before, 3D ground truth data was adjusted on the Kgalagadi dataset and compared to the 3D estimates. The results are presented in Figure 5.11.

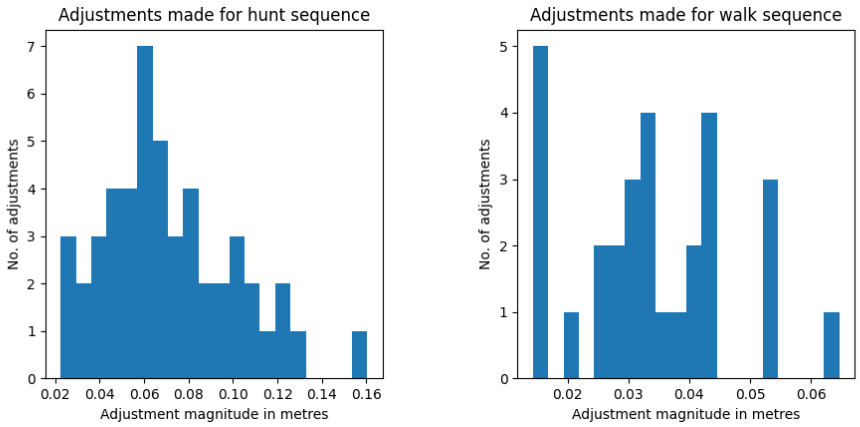

Figure 5.11: Histograms showing adjustment magnitudes for the hunt and walk sequences from the Kgalagadi dataset.

Once more, a numerical summary for the adjustments is provided in Table 5.5.

	RMSE	Std Dev	Percentage Adjusted
Run	3.41	1.29	7.25 %
Hunt	7.18	2.98	12.25 %
Overall	5.77	3.08	9.75 %

Table 5.5: Numerical breakdown of 3D adjustment magnitudes on the Kgalagadi data

5.2.3 Discussion

In general, the method performed noticeably better on low-challenge data than on high-challenge data as expected. However, performance on high-challenge data across both datasets was still admirable, and in most metrics remained fairly close to the low-challenge data in accuracy. One significantly varying metric was the proportion of outliers - in the case of the Kgalagadi data, the outlier ratio was over **23 times** higher in the hunt sequence than the walk sequence. This large discrepancy may be explained by the fact that the hunt dataset was by far the most challenging sequence across all collected data - displaying most of not all confounding factors that plague modern computer vision problems: dust occlusion, motion blur, multiple animal subjects in view, complex poses, and high speeds.

Both qualitative and quantitative comparison revealed that performance on the Hartbeespoort dataset was higher than the Kgalagadi dataset - again, this was to be expected. This occurred despite the subject being further away from the data capture rig on average in Hartbeespoort than in the Kgalagadi. Higher resolution visual data as well as more ideal, controlled conditions are presumably to blame for this. The Hartbeespoort data had the advantage of being recorded from a stationary tripod setup in a typically bright, controlled environment - whilst the Kgalagadi data needed to overcome the non-trivial vibrations from the vehicle mount as well as more varied environmental and lighting conditions.

These results, both visual and numeric, are promising considering the overall difficulty of the datasets. The adjustment metrics provide a more comprehensible, real-world picture of accuracy by summarising performance in average deviation from "ideal" positions in cm. These metrics are very low when one takes into account the range at which they were measured and the size of the subject itself.

A results video showcasing 3D reconstruction sequences may be viewed at `https://youtu.be/F2Tbeyh4HcE`.

Chapter 6

Conclusions

In this chapter, this study will be concluded with a brief summary of our findings as well as recommendations for future related research.

6.1 Conclusion

To conclude, in this thesis we have presented WildPose: a method for the 3D motion capture of cheetahs in the wild. A mechatronic design for a data capture rig was described and evaluated over several iterations. The rig was then used to gather RGB and lidar data from several subjects across locations in Hartbeespoort as well as Kgalagadi National Park. This data was then fused in a trajectory optimisation-based method derived from prior work that proved effective in reconstructing 3D pose, even at long ranges of up to 90 metres.

The reconstructed pose data was tested rigorously against various methods of obtaining ground-truth data to provide an in-depth analysis of performance. Although accuracy was higher in general when higher resolution images were used, even low-resolution videos at 480p provided rich enough information to yield 3D pose errors of under 10 cm at 50 m range when fused with dense lidar data.

We present these findings as promising evidence that multi-sensor fusion is able to compensate for many of the shortcomings of pure vision-based methods when it comes to pose estimation. Our demonstrated accuracy on cheetahs - which tend to be tricky subjects - may be extended to a larger variety or species using the same underlying multi-sensor fusion techniques.

6.2 Recommendations

Here, recommendations for future work are made based on the challenges and successes experienced during the course of this study.

The first major area for future work is the extension of these methods to other species. As mentioned, these methods may be transferred to other species with relatively minor adjustments. The biggest change would be to develop a neural network for pose estimation of the new species, which requires a non-trivial amount of training data. However, tools such as DeepLabCut [3] make neural network development easy, and most of the work boils down to the acquisition of training data. A new kinematic model for the FTE stage must also be developed; however this can be done quite quickly. Theoretically, the method may be extended to any user-defined species.

Another area for possible future work is the inclusion of more sensors onto the data capture rig. Accuracy could possibly be improved even further (especially in high-occlusion scenarios) through the addition of extra sensor data (such as mmWave radar readings or additional cameras) into the FTE as an added set of measurements. The programmatic addition of an extra set of sensor data would be relatively straightforward as this study itself demonstrates the use of this exact idea.

The reconstruction algorithm may be improved further through more post-processing work on the lidar point clouds. Point clouds are rich in information, especially when used for 3D pose reconstruction or mapping - thus, more information could be gleaned than what is currently used in

WildPose.

The experimentation of WildPose or similar systems at higher ranges also presents an interesting problem. The maximum range studied here is around 90 m - however, for some applications, the subjects may be even further away. Further study into longer-range motion capture would provide a wider range of tools to future researchers who find themselves needing to reconstruct pose at extreme distances.

The calibration method used in this study may also be further evaluated and developed. The use of non-planar calibration rigs is not well studied; more investigation into the topic would benefit the field of computer vision. A concept that would greatly improve dynamic calibration would be calibration using a non-rigid rig. If such an algorithm were to be achieved, the animal subjects themselves could be used as calibration targets.

Another area of research that may be of interest is the incorporation of another layer of machine learning on top of the FTE's 3D estimates. Currently, a 2D convolutional neural network is used only to provide 2D keypoint estimates as the first stage in the process. Neural networks have been used for 3D pose estimation in the past [46] - a similar method may be employed to further increase the accuracy of the WildPose method.

Acronyms

EKF Extended Kalman Filter.

FOV Field of View.

FTE Full Trajectory Estimation.

GPS Global Positioning System.

IC Integrated Circuit.

IMU Inertial Measurement Unit.

SBA Sparse Bundle Adjustment.

SoC System on Chip.

Glossary

locomotion Movement, or the ability to move between two points..

morphology A particular shape or form of an object or being..

odometry The use of sensor data to estimate the change in position of a target over time..

pose estimation The process of determining an object's position and orientation in some coordinate frame..

stereo camera A setup involving multiple cameras at differing angles used to infer depth whilst recording a subject..

trajectory optimisation The process of calculating a trajectory that meets a set of given constraints whilst maximising some objective..

Bibliography

[1] "Spot® - the agile mobile robot." `https://www.bostondynamics.com/products/spot`. Accessed: 2023-01-29.

[2] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *CVPR*, 2017.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

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

[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] D. Joska, L. Clark, N. Muramatsu, R. Jericevich, F. Nicolls, A. Mathis, M. W. Mathis, and A. Patel, "Acinoset: A 3d pose estimation dataset and baseline models for cheetahs in the wild," *CoRR*, vol. abs/2103.13282, 2021.

[11] N. Niknejad, J. Caro, R. B. Puhl, Y. Bao, and E. A. Staiger, "Estimation of equine stride length and stance duration using stereo 3d videography and deep learning," in *2022 ASABE Annual International Meeting*, p. 1, American Society of Agricultural and Biological Engineers, 2022.

[12] A. Patel, B. Stocks, C. Fisher, F. Nicolls, and E. Boje, "Tracking the cheetah tail using animal-borne cameras, gps, and an imu," *IEEE sensors letters*, vol. 1, no. 4, pp. 1–4, 2017.

[13] B. J. Hosticka, "Cmos sensor systems," *Sensors and Actuators A: Physical*, vol. 66, no. 1-3, pp. 335–341, 1998.

[14] D. Mehta, S. Sridhar, O. Sotnychenko, H. Rhodin, M. Shafiei, H.-P. Seidel, W. Xu, D. Casas, and C. Theobalt, "Vnect: Real-time 3d human pose estimation with a single rgb camera," *Acm transactions on graphics (tog)*, vol. 36, no. 4, pp. 1–14, 2017.

[15] E. Brachmann, F. Michel, A. Krull, M. Y. Yang, S. Gumhold, *et al.*, "Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3364–3372, 2016.

[16] G. Moon, S.-I. Yu, H. Wen, T. Shiratori, and K. M. Lee, "Interhand2. 6m: A dataset and baseline for 3d interacting hand pose estimation

from a single rgb image," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XX 16*, pp. 548–564, Springer, 2020.

[17] Y. Cai, L. Ge, J. Cai, and J. Yuan, "Weakly-supervised 3d hand pose estimation from monocular rgb images," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 666–682, 2018.

[18] R. Juarez-Salazar, J. Zheng, and V. H. Diaz-Ramirez, "Distorted pinhole camera modeling and calibration," *Applied Optics*, vol. 59, no. 36, pp. 11310–11318, 2020.

[19] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 ed., 2004.

[20] M. Pollefeys, R. Koch, and L. V. Gool, "Self-calibration and metric reconstruction inspite of varying and unknown intrinsic camera parameters," *International Journal of Computer Vision*, vol. 32, no. 1, pp. 7–25, 1999.

[21] K. Konolige and W. Garage, "Sparse sparse bundle adjustment.," in *BMVC*, vol. 10, pp. 102–1, 2010.

[22] A. De la Escalera and J. M. Armingol, "Automatic chessboard detection for intrinsic and extrinsic camera parameter calibration," *Sensors*, vol. 10, no. 3, pp. 2027–2044, 2010.

[23] J. Weng, P. Cohen, M. Herniou, *et al.*, "Camera calibration with distortion models and accuracy evaluation," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 14, no. 10, pp. 965–980, 1992.

[24] U. Wandinger, *Introduction to lidar*. Springer, 2005.

[25] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time.," in *Robotics: Science and Systems*, vol. 2, pp. 1–9, Berkeley, CA, 2014.

[26] T. Hare, M. Masson, and B. Russell, "High-density lidar mapping of the ancient city of mayapán," *Remote Sensing*, vol. 6, no. 9, pp. 9064–9085, 2014.

[27] R. W. Wolcott and R. M. Eustice, "Visual localization within lidar maps for automated urban driving," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 176–183, IEEE, 2014.

[28] M. Flood, "Laser altimetry: From science to commerical lidar mapping," *Photogrammetric engineering and remote sensing*, vol. 67, no. 11, 2001.

[29] T. Seel, J. Raisch, and T. Schauer, "Imu-based joint angle measurement for gait analysis," *Sensors*, vol. 14, no. 4, pp. 6891–6909, 2014.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

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

[39] 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.

[40] H. E. Benseddik, O. Djekoune, and M. Belhocine, "Sift and surf performance evaluation for mobile robot-monocular visual odometry," *Journal of Image and Graphics*, vol. 2, no. 1, pp. 70–76, 2014.

[41] 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.

[42] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.

[43] G. H. An, S. Lee, M.-W. Seo, K. Yun, W.-S. Cheong, and S.-J. Kang, "Charuco board-based omnidirectional camera calibration method," *Electronics*, vol. 7, no. 12, p. 421, 2018.

[44] 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.

[45] A. Wachter and L. T. Biegler, "On the implementation of an interior point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.

[46] A. Grinciunaite, A. Gudi, E. Tasli, and M. Den Uyl, "Human pose estimation in space and time using 3d cnn," in *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III*, pp. 32–39, Springer, 2016.