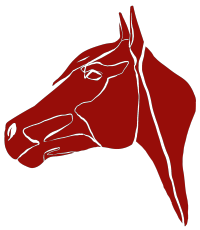
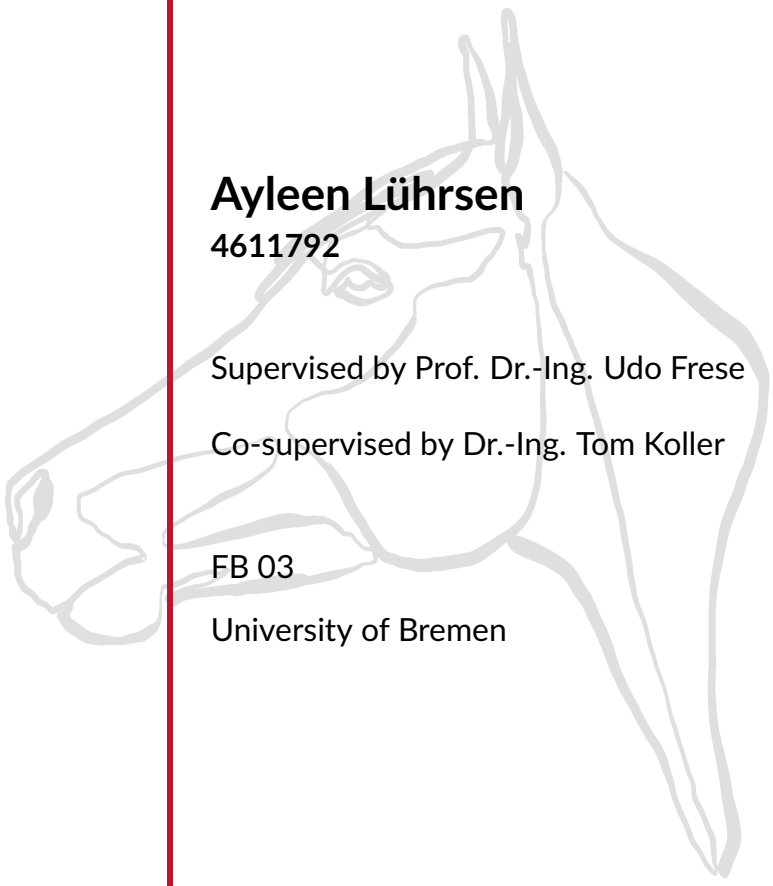


EquiNeT: Pose estimation for objective lameness diagnosis in horses with deep learning

A Thesis submitted for the Degree of
Master of Sciene** in **Computer Science



Ayleen Lührsen
4611792

Supervised by Prof. Dr.-Ing. Udo Frese

Co-supervised by Dr.-Ing. Tom Koller

FB 03

University of Bremen

August, 2025

Master Thesis

University of Bremen
Faculty 3: Mathematics and Computer Science

University of Bremen, Thursday 14th August, 2025

Name Ayleen Lührsen
Matrikelnummer 4611792

A) Eigenständigkeitserklärung

Ich versichere, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Alle Teile meiner Arbeit, die wortwörtlich oder dem Sinn nach anderen Werken entnommen sind, wurden unter Angabe der Quelle kenntlich gemacht. Gleiches gilt auch für Zeichnungen, Skizzen, bildliche Darstellungen sowie für Quellen aus dem Internet, dazu zählen auch KI-basierte Anwendungen oder Werkzeuge. Die Arbeit wurde in gleicher oder ähnlicher Form noch nicht als Prüfungsleistung eingereicht. Die elektronische Fassung der Arbeit stimmt mit der gedruckten Version überein. Mir ist bewusst, dass wahrheitswidrige Angaben als Täuschung behandelt werden.

B) Erklärung zur Veröffentlichung von Bachelor- oder Masterarbeiten

Die Abschlussarbeit wird zwei Jahre nach Studienabschluss dem Archiv der Universität Bremen zur dauerhaften Archivierung angeboten. Archiviert werden:

1. Masterarbeiten mit lokalem oder regionalem Bezug sowie pro Studienfach und Studienjahr 10 % aller Abschlussarbeiten
2. Bachelorarbeiten des jeweils ersten und letzten Bachelorabschlusses pro Studienfach und Jahr

Ich bin damit einverstanden, dass meine Abschlussarbeit im Universitätsarchiv für wissenschaftliche Zwecke von Dritten eingesehen werden darf.

C) Einverständniserklärung zur Überprüfung der elektronischen Fassung der Bachelorarbeit / Masterarbeit durch Plagiatssoftware

Eingereichte Arbeiten können nach § 18 des Allgemeinen Teil der Bachelor- bzw. der Masterprüfungsordnungen der Universität Bremen mit qualifizierter Software auf Plagiatsvorwürfe untersucht werden. Zum Zweck der Überprüfung auf Plagiate erfolgt das Hochladen auf den Server der von der Universität Bremen aktuell genutzten Plagiatssoftware.

Ich bin nicht damit einverstanden, dass die von mir vorgelegte und verfasste Arbeit zum o.g. Zweck dauerhaft auf dem externen Server der aktuell von der Universität Bremen genutzten Plagiatssoftware, in einer institutionseigenen Bibliothek (Zugriff nur durch die Universität Bremen), gespeichert wird.

Mit meiner Unterschrift versichere ich, dass ich die obenstehenden Erklärungen gelesen und verstanden habe und bestätige die Richtigkeit der gemachten Angaben.

Signature of Student _____

Date Thursday 14th August, 2025

You have power over your mind - not outside events.
Realize this, and you will find strength.

Marcus Arelius, Meditations

Acknowledgements

Many thanks to my supervisor Prof. Dr.-Ing. Udo Frese, who agreed to supervise this topic and was always a great help throughout the entire process and also to Dr.-Ing. Tom Koller, who took the time to evaluate the work as a second reviewer. A big thank you to Jens Körner, who approached the university with the topic of objective lameness diagnostics and thus enabled me to work on my dream topic.

Of course, I must also express my heartfelt thanks to my wonderful mother – without you, neither my passion for horses nor my studies would have been possible. And finally, a big thank you to all my friends and acquaintances who have supported me mentally during this time and, in some cases, contributed video material.

TO MAJA AND ALBERON, WHO I LOVE SO DEARLY.
AND TO ALL THOSE, WHO NEVER BELIEVED THEY WOULD MAKE IT. YOU WILL.

Abstract

Objective lameness evaluation in horses is an important aspect of equine welfare, as accurate detection of locomotor abnormalities enables timely and effective treatment. Video-based markerless motion analysis offers a practical and accessible approach, but requires reliable pose estimation models that can perform well under varying conditions. This work investigates the use of synthetic data to enhance the training of neural networks for equine pose estimation. A novel, adaptable Python framework for generating realistic synthetic datasets for pose estimation tasks was developed, producing over 900 high-quality images of horses in diverse and realistic settings. In addition, a dataset of real-world videos was collected for evaluation. Several network architectures were trained in DeepLabCut using different combinations of synthetic and real data. Results indicate that deeper architectures allow for the use of higher-resolution inputs compared to prior related research, improving the archievable detection of subtle details. The best-performing configuration was a ResNet-101 trained on a mixed dataset of synthetic and real samples, though differences to HRNet-w48 trained on the same dataset were negligible in visual inspection. The findings confirm that synthetic data is a valuable complement to real-world data in this context. This work establishes a solid foundation for developing clinically applicable, markerless lameness detection systems, but further research to improve the consistency and accuracy of the recorded trajectories is needed.

Contents

List of Figures

List of Tables

1

Introduction

1.1 | Motivation

Equine lameness has always been an issue that heavily impacts the welfare, health and usability of horses. Lameness as a veterinarian term describes abnormal gait and stance behavior, for example leading to an uneven movement pattern and avoidance of putting strain on the leg. Depending on the cause of lameness and stress on the horses body, the lameness can present itself in various degrees and types of lameness.

To minimize long term negative impact on the horses health, it is important to identify and treat lameness as early as possible. However, especially the early stages are often hard to detect if the cause of lameness is not a traumatic injury but rather a gradual process i.e. wear and tear of the musculoskeletal system due to age or wrong training.

Recognizing light degress of lameness is a challenging task for lay persons and even veterinarians. Even if the latter group is experienced in recognizing the lameness, it can be hard to articulate their assessment in understandable terms to their peers due to different parameters used and therefore leading to misunderstandings when discussing a medical case[19, 17]. Overall, both the veterinarian-to-veterinarian and the veterinarian-to-horse-owner communication could greatly benefit from measurable and objective parameters to identify lameness and make a diagnosis.

Digital tools such as IMU-Sensors provide the necessary data for an *objective lameness evaluation*, and over the past years there has been a number of studies done how these sensors can be incorporated in lameness assessments [6, 9]. IMUs however do have a major drawback, which is their need for specific equipment in addition to inaccuracies due to different placements on the horses body and a high sensitivity to natural asymmetries in the horse not related to actual lameness, which needs to be considered when evaluating the sensor data. [18]

With the rise of *deep learning*, another method for objective lameness assessment emerged: by using videos of moving horses and an inference performing a pose estimation on each frame, it became possible to track body parts and measure their movement

over time, therefore allowing to find deviant movement patterns without any specific hardware needed other than a smartphone camera.

A common problem with computer vision appraoches like this in very specific fields is usually obtaining suitable training data in large amounts to train such models. As horses come in many different sizes, shapes and colors and additionally are often trained and clinically examined in outdoor conditions and therefore subject to varying backgrounds and lighting conditions, it is challenging to train a model that can generalize well enough to be used in practical real world scenarios by both veterinarians and horse owners.

A possible solution to this problem is *synthetic data*, which means artificially created data samples, in this case images. Usually synthetic images are created as renders of 3D-scenes, modeled in a software such as *Blender* [51]. While these images are not suitable for fine tuning of a deep neural network, they can be used to increase the size and variance of the training data set when real data is hard to obtain or there are not enough resources available to label such data. The main goal of using synthetic data is to increase the models ability to generalize on unseen, real data.

In cooperation with the orthopaedic department of the *Hanseklinik für Pferde (engl.: Hanse Equine Hospital)* in Sittensen, this thesis strives to develop a robust model that is suitable for real world usage in the day-to-day business of the clinic by using both real and synthetic datasets.

Therefore, this thesis will answer the following question:

To what extent can synthetic data serve as an alternative or complement to real-world data for training neural networks in animal pose estimation for equine lameness evaluation?

To answer this question, the thesis will make three main contributions:

- Development of a Python library to create synthetic data for keypoint based pose estimation
- Creation of multiple 3D-assets of horses and corresponding training facilities and a dataset that is rendered in Blender using those assets and the previously described Python package
- Training and comparison of different *Neural Neetwork* architectures and training strategies (synthetic, real and mixed data)

1.2 | Structure

In this section the general structure and content of this thesis will be described.

Due to its interdisciplinary nature, both technical and veterinary basics necessary will be described in chapter 3 and 2.

In chapter 2 a basic overview of the anatomy of the horse, its gaits and how lameness is defined and diagnosed will be given. This is necessary to give some context to certain design choices made in later chapters, especially in chapter 5 and 6.

Chapter 3 will start off with a short introduction (Deep) Neural Networks in section 3.1. These section will be rather short and go over the most important concepts and definitions, as there is plenty of literature available for the more interested reader. In section 3.3 the concept, benefits and challenges of synthetic data will be described, as it is the basis for the dataset used in this thesis. The topic of pose estimation will be introduced in section 3.2.

PoseCraft, a python package for synthetic data generation that was developed for this thesis is presented in chapter 4. The chapter will include the motivation for developing a new framework, list the design criterias and constraints that were taken into account, describe the software and how it can be used as well as name some challenges that were overcome during development.

A practical example how *PoseCraft* can be used will be presented in chapter 5: Here, the dataset that was generated for the training of the network is presented. Since many of the 3D Assets were created or adapted for the purpose of this thesis, some of the steps in Blender will be described and design choices explained. There will also be an overview of the statistics of the dataset, such as the number of images, keypoints and classes.

The generated dataset will be put to use in chapter 6. Using the *DeepLabCut* framework, multiple models are trained on both synthetic and real data. The collected set of real world data and the annotation process will also be described. All models trained will be thoroughly evaluated and compared with each other, followed by a discussion of the results.

To reconnect the three main contributions of the thesis again, Chapter 7 will present a general conclusion and highlight some possible ways to go forward with the results of this thesis.

As this thesis heavily relies on visual material, a lot of examples can be best given in visual form. To not clutter the main part of the thesis, Appendix A will contain additional examples and figures that did not make it into the main part of the thesis, but might be interesting for some readers and give a more detailed impression of the

datasets. In Appendix B, some code snippets can be found to illustrate how the software written for this thesis works. Finally, in Appendix C a list of all 3D assets created or modified for this thesis can be found.

2

Veterinary Background

In this chapter a the reader will be provided with the necessary knowledge of horse anatomy and lameness diagnosis. Certain terms such as lamness will be defined, and related research in the area of objective lameness evaluation will be presented.

2.1 | Anatomy and Gaits of the Horse

While most likely everyone knows what a horse looks like, this section will go over some lesser known traits of the equine body and what types of gaits a horse has.

2.1.1 | Basic anatomy of the horse

Horses are quadruped[1] mammals with the a muscosceletal system that is highly optimized for effective running. Their exterieur includes the head (with eyes positioned laterally for wide vision), neck (providing balance and aiding in movement), withers (the ridge between the shoulder blades), back, loin, and croup leading to the tail. Their legs are long and strong. The forelimbs will carry more of the weight due to the distribution of the bodymass, while the hind legs offer power and propulsion for movement.[11]

While most of the bones in the human skeleton have an equivalent part in the skeleton of a horse, one important difference is that horses, like other quadrupeds as dogs and cats, do not have a *clavicle (collar bone)*. Instead, their shoulders are connected to the body by soft tissue such as muscles and ligaments known as the *thoracic sling*. [11, p. 59] This allows the trunk more flexibility in movement and the horse can use the muscles to rise and lower its trunk or shift it slightly to the left or right side, which is especially useful in curves at a higher speed. [11, p. 79]

[1] four-legged

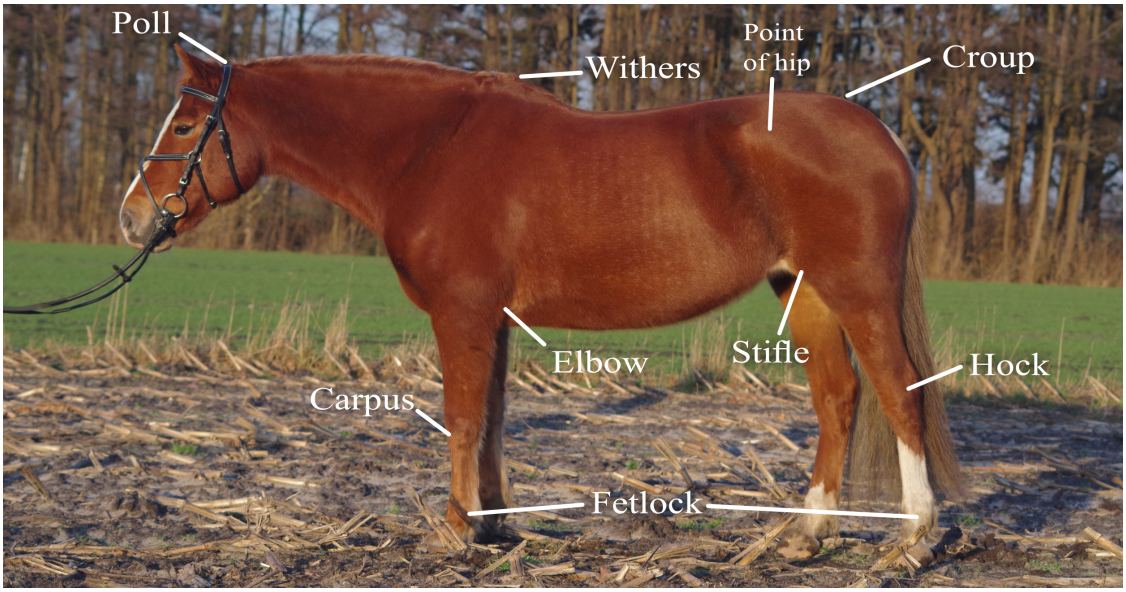


Figure 2.1: A horse viewed from the side with annotated names of the body parts.
Based on [11]

2.1.2 | Gaits of the horse

Most horses move using four different gaits[2]. Those basic gaits are walk, trot, canter and gallop. Some breeds can perform so called *ambling gaits*, however, those as well as the gallop are not relevant here.

During movement, a horse's limbs follow repetitive patterns known as **strides**. Stride length refers to the distance covered between successive placements of the same hoof. Each stride consists of distinct phases: the **stance phase**, when the hoof is in contact with the ground and bearing weight, and the **swing phase**, when the limb is lifted and moving forward through the air. The stance phase can be further divided into *initial ground contact* (the hoof touches the ground), *impact phase* (a short phase after touching the ground, where that the muscles have not adapted yet), *loading phase* (the limb is actively carrying weight) and *breakover* (the heel leaves the ground and rotate around the toe). [11, p. 80f]

Walk: The walk is the slowest gait. It has four-beat rythm where each hoof strikes the ground independently. It is an symmetrical gait. At any point during walk, the horse

[2]In german language, there is no distinction between *canter* and *gallop*, and usually people will speak of three basic gaits, as the gallop is a faster version of the canter with a slight difference in the footing sequence

will bear weight on two or three limbs, and there is no moment of suspension. The average speed in a medium walk is $\approx 6\ km/h$. [11, p. 82-98]

Trot: The trot is faster than the walk. It has a two-beat rythm. The diagonal limbpairs will move forward at the same time, followed by a moment of suspension before reaching the ground again with the other two limbs swinging forward. Therefore, each stride features two phases of suspension. Like walk, a horses trot is symmetrical. How fast a horse moves in trot depends on the collection[3] and the individual locomotion of the horse as well as it's size. A range between 10 km/h and 16 km/h will be accurate for most horses. [11, p. 99-120]

Canter: The canter has a three-beat rhythm. It is an asymmetrical gait: depending on the lead, one hind limb initiates the stride, followed by the diagonal pair (consisting of the other hind limb and the opposite forelimb), and finally the leading forelimb. This is followed by a moment of suspension before the next stride begins. The speed by which a horse can move in canter depends on similar factors as it does for the trot. At high speeds, it will be faster than the trot of most horses, but a collected canter can be slower than an extended trot. [11, p. 121-139]

As it is asymmetrical, the canter is not a part of lameness examinations.

2.2 | Lameness in Horses

In this section, the term *lameness* will be defined. Different types and grades of lameness will be presented, and the procedure of a clinical lameness examination

2.2.1 | Definition of Lameness

Lameness describes an abnormal movement pattern in horses, usually as a result of pain. The causes of lameness can be varied and may be caused, for example, by trauma (single or repeated), developmental disorders, infections or many other reasons. [2, p. 67] A lame horse will usually try to reduce the load on the affected limb with compensatory movements. This can become apparent through untypical movements of the head (the horse "nods" each time the affected limb is bearing weight), a reduced extension of

[3] Collection means that the horse's hind legs carry more weight, which is particularly desirable when riding over long distances in order to keep the horse healthy. Collection is usually associated with a more upright neck and a raised torso.

Table 2.1: Grades of lameness as defined by the AAEP.

Grade	Description
0	Lameness not perceptible under any circumstances.
1	Lameness is difficult to observe and is not consistently apparent, regardless of circumstances (e.g. under saddle, circling, inclines, hard surface, etc.).
2	Lameness is difficult to observe at a walk or when trotting in a straight line but consistently apparent under certain circumstances (e.g. weight-carrying, circling, inclines, hard surface, etc.).
3	Lameness is consistently observable at a trot under all circumstances.
4	Lameness is obvious at a walk.
5	Lameness produces minimal weight bearing in motion and/ or at rest or a complete inability to move.

joints like the fetlock or even an altered pattern of leg movement. [2, p. 67] This compensatory behavior can lead to alterations of the body, for example a wide and flat hoof on the weight-bearing, sound limb and a more narrow and upright hoof on the lame, unloaded limb. [2, p. 68]

An observer will notice an asymmetrical movement pattern in the horse's locomotion. However, it must be stressed that almost every horse does move in slight asymmetrical way especially on curved lines, and asymmetry is not a strict indication of lameness or pain. [41]

Lameness becomes usually most apparent during the trot; since canter is an asymmetrical gait it is not suitable to spot lameness. In walk, due to the abscence of a floating phase, a lameness will only become apparent if it is severe. A light lameness is usually not detectable in walk (see Table 2.1). [11, p. 120]

2.2.2 | Degrees of Lameness

There are various scales used to classify the severity of lameness. One widely used scale is that of the *American Association of Equine Practitioners (AAEP)*. It divides lameness into five degrees, which are based on the gaits in which the lameness is visible and the extent to which the affected leg is avoided. The five degrees of lameness are listed in Table 2.1, as it was found in [2, p. 137] and [22, 7].

2.2.3 | Types of Lameness

It is not only important to know whether a horse is lame, but also what limb is affected and what type of lameness is occuring. Lameness can be classified in different ways. Two of the most important distinctions for this thesis are:

Forelimb Lameness vs. Hindlimb Lameness: Whether the affected limb is a frontleg or a hindleg will have a big impact on the observable symptoms. A forelimb lameness will usually be expressed by an untypical head-movement in motion and an asymmetric trajectory of the withers. For a hindlimb lameness, the trajectory of the hips and pelvis will deviate. [11, p. 120]

Supporting Limb Lameness vs. Swinging Limb Lameness: By far the most common type of lameness is the supporting limb lameness. The horse will express pain when the impaired hoof touches the ground and bears the weight of the body. It often caused by injuries to bones, joints or supporting structures such as tendons. If a horse shows signs of a swinging limb lameness, the symptoms will become evident during the motion of the limb. Often, the cause of this type of lameness is located in the upper part of the limb or skeleton. [2, p. 68]

2.2.4 | Procedure of a lameness examination

The purpose of lameness examination is not just to determine whether a horse is lame, but also which limb or limbs are affected. The author was invited by Dr. Jens Körner to observe several lameness examinations at the *Hanseklinik für Pferde*. A lameness examination as described in [7, 2] usually consist of these steps (with deviations between practicioners):

Observe the standing horse and examine its confirmation. Sometimes, the body of the horse can already give hints whether issues with the musculoskeletal system or other parts of its body are present. Therefore the horse should be thoroughly examined.

Trotting on a straight line. The horse will be presented in walk and trot on a straight line, usually on hard ground. The veterinarian looks out for deviating movement patterns of the head (head nods) and hips (hip nods).

Trotting on a circle on hard and softground. The strength of asymmetries in locomotion can vary depending on whether it moves on a straight or curved line. [40, 41] Therefore, the examination usually includes the presentation of the horse on a circle. The ground substrate is also proven to have an influence on the severeness of the asymmetries it can be beneficial to lunge the horse on both hard and soft surfaces if local conditions permit it. [38]

Manipulation of the limbs. In order to make the change in the horse's movement more apparent, a so-called flexion test is often performed. This involves bending a joint sharply for a few minutes, which can exacerbate existing lameness and make it more visible, but there is a risk of false positives on legs that are actually healthy. [2, p.115]

Another method of manipulation is injection with anaesthetics. Starting at the joints in the hoof, an anaesthetic is injected into the nerve to suppress the pain impulse during movement. Once the anaesthetic has taken effect, the horse is repeatedly put back into motion and examined. If it still shows signs of lameness, the process is repeated with the next higher joint. In this way, the cause of the lameness can be localised. [2, p.157]

This way of examining horses by visual inspection through a professional veterinarian is called *subjective lameness evaluation*. Studies have shown that especially less experienced veterinarians and laypersons will often overlook or misclassify signs of light lameness. Research on the reliability of subjective lameness evaluation is presented in Section 2.3.1.

2.2.5 | Measurable parameters in lame horses

In veterinary research it is a topic of interest to identify quantitative measurable values to evaluate whether a horse is lame. These values can be measured with modern sensors like *IMUs (Inertial Measurement Units)*[4] or camera based motion tracking systems.

Often surprising to laymen is that during a lameness evaluation the main focus does usually not lay on the legs and the trajecories of the upper body are usually more interesting. The consensus among researchers is that it is primarily the movement of the head and hips that can provide information about existing lameness. [2, p. 126-134]

However, especially the head trajectory will often be very unstable as the horse does not only use it's neck to balance itself during movement, but also look around. Especially in new stressfull situations and unknown environments like during the examination in a horse clinic, horses will often lift their head high to get a better view of their surroundings and move the head a lot unrelated to the normal movement.

More recently, the trajectory of the withers garnered more attention by researchers. While it is less sensitive to lameness especially on straight lines compared to the head movement, it can deliver additional information and help differentiate between forelimb and hindlimb lameness. [2, p. 129]

[4] A sensor that will usually measure acceleration (Accelerometer), rotation rate (Gyrometer) and sometimes the magnetic field (Magnetometer).

When a horse trots, its body moves down as it "falls" on the first pair of limb in their stance phase and then up when it pushes off the ground with this pair of legs during one half of the stride, and again down and then up during the other half, with the other diagonal limp pair. The head, withers and pelvis normally follow this pattern. In a healthy horse, the highest and lowest point of this periodic curve should be the same for both halves of a stride. [2, p. 126f]

However, in a lame horse, this will usually not be the case. In an effort to relief the affected limb and quickly move the bodyweight back onto the other limb, it will lift it's head and body up, leading to different peaks of the trajectory in a stride. [2, p. 126f]

Generally, a forelimb lameness will be indicated by the trajectories of head and withers, and a hindlimb lameness by the trajectory of the pelvis. A higher tempo in trot will lead to a higher amplitude of the recorded signal. [39]

2.3 | Lameness Evaluation

As previously mentioned, subjective lameness evaluations can lack reliability. In this section, research on this topic and emerging approaches to objective lameness evaluation will be presented.

2.3.1 | Reliability of subjective Lameness Evaluation

Multiple studies investigated the agreement between different veterinarians when examining the same horses for lameness.

In "Rater Agreement of Visual Lameness Assessment in Horses during Lungeing" [17] the authors collected videos of horses and send them to practicing veterinarians with varying levels of experience. The veterinarians who accepted the invitation to participate, finished the presented task in form of an online questionaire and were not suspect to concerns regarding their integrity ($n = 86$) were classified as either *experienced* (> 5 years of practical orthopaedic experience) or *less experienced* (≤ 5 years of experience). The authors found that especially the less experienced veterinarians have a very poor inter-rater agreement[5]. The veterinarians in the experienced group had a higher level of inter-rater agreement, however the authors still noted that it was only "moderately acceptable". This implicates more reliable ratings for more experienced veterinarians. The lowest level of agreement was calculated for whether a horse was

[5] *Inter-rater agreement:* Agreement for a certain decision/diagnosis between different persons

sound[6], meaning some of the study participants did not see a mild lameness or rated a sound horse as lame.

Starke and May conducted a similar study in "Veterinary Student Competence in Equine Lameness Recognition and Assessment"[46]. In this study, the participants consisted of veterinary students in different years of their studies, again classified as either experienced or less experienced. The experienced students were better at evaluating the shown videos than the inexperienced students, but still taking the wrong choice for 1 in 4 horses. The results for sound horses were on the same level as chance. In this study, the authors additionally made use of an eye-tracking method to investigate which parts of the horse were mainly looked at to identify lameness. The authors only used th front and rear view and no side view, however the results implied that with increasing experience the students paid less attention to the movement of limbs and more attention to the upper body parts like head and hip movement. The authors conclude that both groups results were inadequate, and students should receive better peceptual training before going into clinical rotations.

Another study from 2019 by Starke and Oosterlinck also investigated inter-rater agreement, this time on animated 3D models of horses rather than real video footage[47]. Participants consisted of both practicing veterinarians and veterinarian students. Again, the results were rather poor and performancewise in line with the studies described prior. The authors further found that there is no correlation between the self-rated confidence of the viewer and the actual correctness of his or her evaluation.

Considering the studies cited above, there is a lot of evidence that subjective lameness evaluations lack realiability, especially if the veterinarian or student is inexperienced, and additional objective methods could be a great benefit to the field for both horse owners and veterinarians.

2.3.2 | Objective Lameness Evaluation

Objective Lameness Evaluation describes the usage of quantitative, technology-based methods that can be used as additional help during lameness examinations. There are many different forms of objective lameness evaluation systems, which are based on force plates, IMUs or camera-based systems. [21, 44] For this thesis, the two former ones will be ignored; instead, the use of video based systems will be the focus. As this thesis was originally a proposal made by the *Hanseklinik für Pferde*, they wished for a system that would work with videos recorded on smartphones, but the processing of

[6] not showing any signs of lameness or abnormalities in movement

the video would not necessarily have to happen on the device, therefore leading to less restrainments regarding the required computing power.

2.3.2.1 | Historical context

There has been a number of studies done that use video analysis to evaluate lameness in horses. Video analysis was used as early as 1986 to give a better assessment of unusual movement patterns in the horse [5]. However, the analysis of the video footage recorded by analogue highspeed cameras relied on external hardware to to visualize the trajectories of selected keypoints on the moving horse.

With more technical advancements, Motion Capture systems found their way into the field. For those, the horse is equipped with markers that are attached to their skin at certain skeletal landmarks. Those markers can be tracked by one or multiple cameras, providing trajectories of the markers. Usually multiple cameras are used, allowing for tracking in a 3D space. While the tracking itself is usually very accurate, the markers move with the soft tissue they are attached to, which leads to noise compared to the actual position of bones and joints that are supposed to be measured. [49]

These motion capture systems have another downside: the necessary setup is quite time consuming and the equipment is expensive. Not every horse is examined in a clinic; often, the veterinarian will do the examination at the stable where the horse is kept. This makes a regular use of such systems in daily use very difficult.

Markerless systems are a lot easier to use: With the rise of Deep Learning, many complex Computer Vision problems could now be solved by Neural Networks. While markerless pose estimation for humans or animals is not as popular as other fields of research such as object detection with bounding boxes or segmentation, there is still a lot of relevant research and modern research that will be looked at more closely in the next subsection.

2.3.2.2 | Markerless Pose Estimation for Lameness Evaluation

Due to the downsides of additional equipment needed for other approaches and the growing camera quality and computing power of mobile devices, markerless pose estimation for clinical use has been on the rise both in commercial solutions (see [56]) and research (see [15, 48]).

Both Feuser et al. [15] and Wang et al. [48] trained *Deep Neural Networks* using *DeepLabCut* [34]. Those were used to do pose estimation on horses lunged on a circle or trotted on a straight line and recorded with smartphone cameras. The collected keypoints were then processed to extract their trajectories and evaluated as described in Section

2.2.5. Feuser et al. collected data of 65 horses for the training group, resulting in 454 frames from 215 videos. They trained a *ResNet-50* on these images. They report an error of 6.14 pixel on the evaluation data. The video resolution was reduced by 40% from $1920 \times 1080\ p$ (*FullHD*) to $768 \times 432\ p$. The authors concluded that a better and more accurate model would be needed to make the system feasible for the detection of light lameness (Grade 1-2, see Table 2.1), but the collected results matched the clinical representation of the horses with a more severe lameness (Grade 3-5).

While Wang et al. did a similar setup but filming horses on a straight line, they trained both a *ResNet-50* and *ResNet-101*. Unfortunately, they only report on the train loss as metric, making it hard to determine why they chose *ResNet-50* over the deeper variant and see how the network performs on unseen data.

In [26] by Lawin et al. the commercial *Sleip* was evaluated against a multi-camera motion tracking setup. They conclude that a smartphone app can be suitable for that task. However, it should be noted that some of the researchers are associated with the company behind the *Sleip* app.

Using videos collected on a smartphone seems to be a promising approach for objective lameness evaluation if the models trained for the task reach a sufficient performance. However especially when filming from the inside of a circle, a method to seperate the movement of the horse itself and other noise in the signal, for example due to an unsteady hand has to be developed. In [15] a keypoint on the torso of the horse is used as center, however, this would collide with the idea that the vertical movement of the withers is an important additional metric for lameness evaluation. Other approaches use a tripod and film the horse on a straight line/from the outside of a circle to avoid this problem.

3

Technical Background

Due to the interdisciplinary nature of this work, not every reader may have the technical background knowledge necessary to understand the later chapters. Therefore, this chapter explains some of the core concepts in the field of deep learning. Accordingly, these explanations will aim to explain these concepts as clearly and simply as possible, rather than getting lost in the depths of mathematical and theoretical details.

3.1 | Neural Networks and Deep Learning

3.1.1 | Machine Learning

Machine Learning is a wide field that includes many different methods and concepts. One trait they all share however is that they produce their output by learning from examples rather than being explicitely programmed to do so. *Machine Learning* methods are often divided into three big subclasses: *Supervised Learning*, *Unsupervised Learning* and *Reinforcement Learning*. [27]

Since the latter two fields are not of relevance for this thesis, all following explanations that might be ambigous in the broader context are made in regard to *Supervised Learning*. Also it should be noted that a lot of machine learning models do not work with images; the input of a model could be almost anything that can be represented in a digital format, such as text or audio. However, as for this thesis images are used, they will be the focus in examples and explanations.

3.1.1.1 | Supervised Learning

Supervised Learning is based on the principle of showing a model many different examples that have been manually annotated by humans. Those annotations, so called labels, are also refered to as **Ground Truth**. If the model learns as desired, it will be able to annotate new, unseen data with this information just as a human would. An example

that fits this thesis: If we show a model a lot of images of horses and with each image and also tell the model how that coat color of the horse is called, the model should eventually be able to name the color of the horse on an image it has not seen before. This would be a classic example for a *Classification* task. [29]

There are other types of task than just classification. Another popular technique would be *Segmentation*, where a model is trained to find all pixels of an image that belong to a certain class or instance. For example, being shown an image of a horse, find all pixels that show the horse, therefore creating a mask that seperates the horse from the background. To archieve this, the network would have see many different images of horses for which a human annotated all the areas of the image that actually show the horse and are not part of the background or other foreground objects.

The task that is solved by the models in this thesis is called *Pose Estimation*. In the context of *Human Pose Estimation* or *Animal Pose Estimation* it usually refers to so called *Keypoints* that can be identified and tracked in images and videos. More on that can be found in Section 3.2.

3.1.1.2 | Training and Testing of Machine Learning Models

What was priorly described as "showing a model" examples is in more technical terms the **Training** of a model. How the training actually looks like depends heavily on the type of model; more on the training of *Neural Networks* in the next subsection. The data that is used to train a model should be as diverse and extensive as possible. How much data is needed depends on the chosen model and the complexity of the task. If the model is not presented with enough data during training, it can **overfit**. This means it will perform very good on the training data, but not be able to generalize the underlying patterns to new, unseen data. Therefore, it should always be *tested* on a seperate dataset.

3.1.2 | Neural Networks and Deep Learning

One type of *Machine Learning Models* that not only opened up a whole new world of possibilities for data processing especially on images and audio but also caught a lot of public interest. *Artificial Neural Networks* are not dissimilar in their basic principle to the functioning of the brain, which is where they get their name from. An *Artificial Neural Network* consist of multiple layers. The so called *Input-Layer* will as the name suggests, get the input as basis for the calculations. In case of an image, that would be the color values of each pixel. The output is generated by the *Output-Layer*. What form this output has depends on the type of model. For a classification model, the output would be the likelihood of each class for that specific input. Between the input layer and output

layer is a varying number of *hidden layers*. Each of those layers consists of many *artificial neurons*, which are connected with each other. Each of those connections is assigned a **weight**, that will be used to calculate the value that is passed on to the connected nodes in the next layer, before finally reaching the *Output-Layer*. Which value a *weight* is assigned is determined during the training of the model. A layer can have multipe channels; an intuitive example is a network that will take a RGB-Image as input. The input layer will not only have an input neuron for each pixel in the image, but actually need three neurons per pixel, one for each color value of the input. [29]

A *Neural Network* can be seen as a mathematical model to approximate a **non-linear** function. The more layers and overall parameters a network has, the more complex the function can be. This is why **Deep Learning** led to very good results in fields such as image processing; *deep* refers to a high number of *hidden layers*. However, it is not feasable to stack an unlimited amount of layers on top of each other to get a better model due to the **Vanishing Gradient Problem**. The vanishing gradient problem occurs when gradients in deep neural networks become extremely small during backpropagation, causing earlier layers to learn very slowly or stop learning altogether. [3]

3.1.2.1 | Training of Neural Networks

Since the training of *Neural Network* involves very specific technical terms, these will be explained here. During training, a sample of the training data is passed into the network, more specifically passed into the input layer. Starting with the input values of the sample, the calculated values depending on the input and weights of the neuron connections will be passed from layer to layer, until they reach the output layer. The error between the calculated output and the actual ground truth is used to calculate the **loss** with a so called **loss function**. In a process called **Backpropagation**, the weights are then adjusted to minimize the *loss*. This is done repeatedly many times. In this thesis, the repetitions of this process are measured in **epochs**. In one epoch, the network will be presented each image from the training dataset exactly once. How many iterations one epoch takes is however not only dependant on the number of training images, but also the **batch size**. The batch size decides how many images the network is presented at the same time, in a *batch*. This means the weights of the network will not be adjusted based on a single image, but multiple images at once. Therefore if the dataset consists of 64 images, one epoch will take 64 iterations with batch size 1, or 16 iterations with a batch size of 4. A bigger batch size will speed up the training process, but is limited by the available computing power/memory of the computing unit, which is usually a *GPU*. [31]

3.1.2.2 | Transfer Learning

A pretrained network is a model that has undergone initial training on a huge benchmark dataset such as *ImageNet [8]* or *COCO [28]* that were made accessible to other researcher to test their architectures or use them as foundation to train a network for a domain-specific task. These learned parameters can then be transferred to a new task via transfer learning. Here the whole network or only it's head (referring to the last few layers) is fine-tuned on the dataset that is collected from the actual target domain. Since many of the features are already represented from the initial dataset, the task- and domain-specific training will need significantly less training data and time. [30, 37, 50]

3.1.3 | Architectures for Neural Networks

Over time, many different architectures of neural networks emerged that address various issues with a fully connected deep neural networks. The most important ones for this thesis are the concept of *Convolutional Neural Networks* and the addition of *Residual Connections* in *Residual Networks*.

3.1.3.1 | Convolutional Neural Networks

Convolutional Neural Networks are a type of neural networks that try to reduce the number of weights a network has to learn during training. They employ convolutional layers, where a filter of varying size (often a 3×3 matrix) is slid over every area. While it is hard to prove, it is assumed that early layers typically detect low-level patterns (edges, textures), while deeper layers combine these into higher-level representations (shapes, objects). This hierarchical feature extraction enables CNNs to perform tasks such as image classification and object detection efficiently, with fewer parameters than fully connected networks. *Convolutional Neural Networks* usually consist of multipe convolution blocks, which usually consists of a *Convolution layer* with an *activation function*, and a *Pooling Layer*. A pooling layer decreases the resolution of the input by combining values of multiple pixels into one. By this decrease in resolution the *Receptive Field* of the output neurons grows. *Receptive Field* refers to the amount of pixels that will have an influence on the output value of that neuron. However, this decrese in resolution leads to a loss of information. [24][4][1]

3.1.3.2 | Residual Networks

Residual Networks (often called ResNets) are a type of networks that try to solve the vanishing or exploding gradients that can appear in very deep networks. ResNets solve

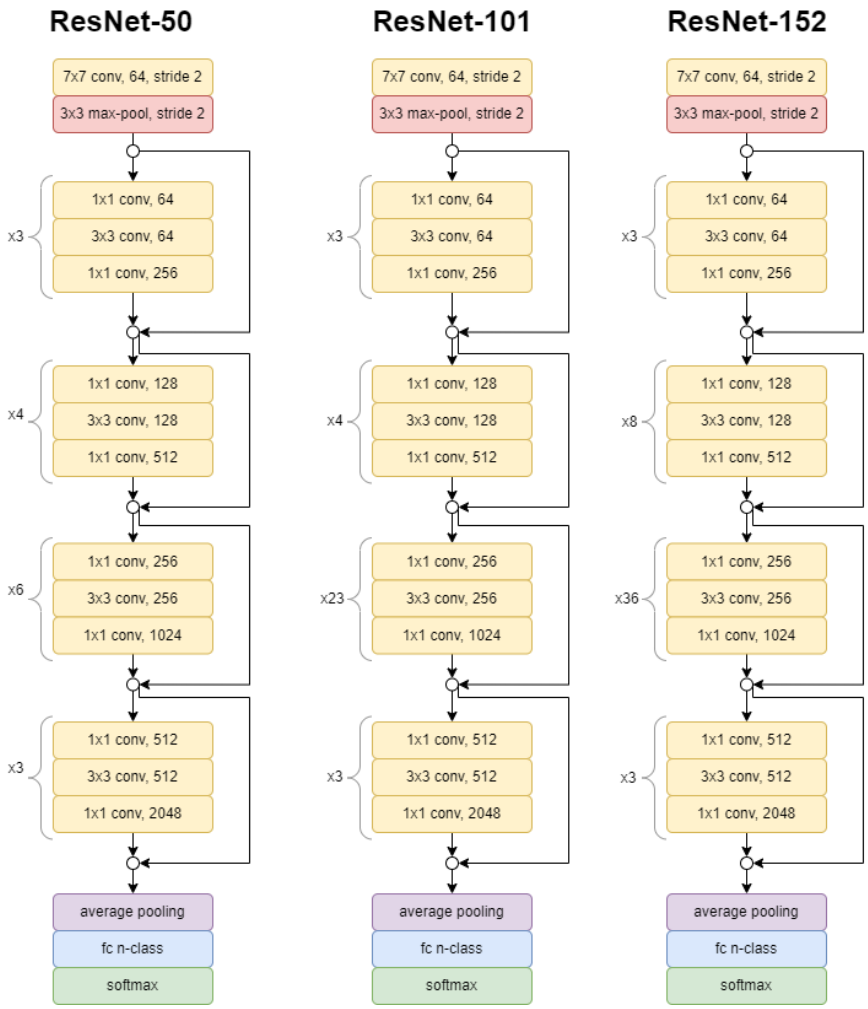


Figure 3.1: Visualization of the ResNet Architectures.
Source: [35]

this by adding *Residual Connections*, that allow information to skip certain layers and get passed directly to a deeper layer of the network. [20, 3]

Figure 3.1 shows the structure of residual networks of varying depths. ResNet-50 and ResNet-101 are also used later in this thesis.

3.2 | Pose Estimation

The term **Pose Estimation** describes a task in Computer Vision, where the goal is to detect poses of a human or animal by identifying priorly specified **keypoints**. Often those keypoints are attached to easily identifiable landmarks of the body, such as eyes,

elbows or hands. How many keypoints are used and where they are on the body is task-dependant. *Pose Estimation* can be applied to images with multipe individuals or just a single one; this can have an influence on the network architecture that is most suitable for the task. This task is most interesting to use on videos; however, usually the model to solve this task will receive each frame of a video (which are images) as an input to work on, rather than processing a whole video at once. [32] [25]

3.2.1 | Approaches

There are two major types of models for Pose-Estimation: **Top-Down** and **Bottom-Up**.

3.2.1.1 | Top-Down Pose Estimation

In this approach, each individual is first identified in a *bounding box*. The *pose estimation* to identify the keypoints is then only done on the bounding box of an individual rather than the whole image. As this needs two models to solve the task (one for the bounding boxes, one for the keypoints), this has higher computational requirements. However, it has the benefit of a better chance to find all individuals in an image, therefore being especially suitable for multi-individual pose estimation. [25]

3.2.1.2 | Bottom-Up Pose Estimation

For the *bottom-up* approach, the model will first find keypoints of all individuals in an image, and connecting them in the next step to form a *skeleton* for each individual. One downside is that the model might struggle to find keypoints on individuals that are either very small or very big. However, this approach requires less computational power and is usually suitable for single-individual pose estimation. [25]

3.2.2 | How are the Keypoints estimated?

The common approach to train a model for pose estimation is based on **heatmaps**. A *heatmap* will map a probability for something to each pixel or, if the heatmap has a lower resolution than the input image, group of pixels for an input image. For pose estimation, this means the model will output a heatmap for each specified keypoint. Each pixel of the heatmap has a value p, with $0 \leq p \leq 1$, representing the likelihood of that keypoint being visible in that exact spot on the input image.

An example for such a heatmap is shown in Figure 3.2. The highest probability will be used as position of the keypoint, if the probability is above a certain threshold.

Figure 3.2: Here, all calculated heatmaps are overlayed onto the input image. The keypoints used for the detection can be recognized well and are clearly distinguishable despite being overlaid.

3.3 | Synthetic Data

3.3.1 | What is Synthetic Data?

Synthetic data refers to data that is artificially generated through statistical modeling or simulation, rather than being directly collected from real-world observations. It has emerged as a critical resource in machine learning, particularly in scenarios where real data is scarce, sensitive, or costly to obtain. The concept of synthetic data was first proposed by Rubin, who introduced the idea that statistical agencies could use artificially generated data to release statistical findings publicly while maintaining the privacy of individuals in the original datasets. [43, 42, 10]

In the context of supervised learning, acquiring labeled data can be especially challenging. The process of annotating data is often labor-intensive and time-consuming, as it typically requires manual effort from human annotators [36, 33, 12]. The input data for machine learning models can take various forms - for example in the form of text, images, or numerical values. Therefore, synthetic data also exists in multiple formats. Emam [12] classify synthetic data into two categories:

- **Synthetic data based on real data**: This involves generating new data that mimics the statistical properties of an existing dataset, often by imitating its probability distribution.
- **Synthetic data not based on real data**: This type is generated independently, such as through simulations that model real-world environments or phenomena.

Besides synthetic data, there is also the term of *augmented data*, which is a different thing. Augmentation techniques use an existing data sample as a base and then apply transformations to expand the dataset by size or variability. In computer vision, for example, this might involve rotating or mirroring images to create new training samples, but also changes of brightness, saturation or contrast of the image. [36].

3.3.2 | Benefits and Disadvantages of Synthetic Data

3.3.2.1 | Benefits

Data availability and scalability. In many scenarios, real data is limited or difficult to obtain. Synthetic data can be used to augment the size of training datasets, helping to prevent overfitting and improve model generalization [36].

Automatic labeling. Manual labeling is a significant bottleneck in machine learning, particularly in computer vision tasks such as image segmentation. Synthetic data, generated in controlled environments such as 3D engines, allows for automatic generation of labels, including segmentation masks. This greatly reduces the time and effort required for data annotation [33, 36].

Privacy protection. The use of personal data in AI systems raises significant privacy concerns. Regulations such as the GDPR impose strict requirements on data collection and storage [13]. Synthetic data offers a way to sidestep many of these issues, as it does not contain identifiable personal information and can be shared more freely [33].

Data diversity. In real-world data collection, it is often difficult to capture sufficient variation in factors such as lighting conditions, object occlusion, or rare edge cases. Synthetic data can be manipulated to introduce these variations deliberately, resulting in more robust and generalizable models.

3.3.2.2 | Disadvantages

Domain gap. A key concern in using synthetic data is the domain transfer problem. For example, in computer vision, synthetic images rendered from 3D models often differ in appearance from real-world images. This discrepancy, known as the domain gap, can impair the model's performance on real data. To mitigate this issue, synthetic data should be as realistic as possible to ensure a smoother transition between synthetic training data and real-world test data [36].

Lack of real-world imperfections. Synthetic data typically lacks the sensor noise and imperfections present in real-world data. In computer vision, this might include noise introduced by camera sensors (e.g., ISO noise), motion blur, or lighting artifacts, which can affect model accuracy if not accounted for in the training data.

There are methods that strive to reduce the domain gap and make the translation to real-world data easier. Examples for this would be the addition of gaussian noise, color shifts, occlusion and blur effects archieve a higher data variety and make it more realistic.

3.4 | Related Research

Using synthetic data to train deep neural networks for pose estimation tasks on animal is not a new concept; this section will take a look at similar relevant research.

Jiang et al. did an extensive survey on the existing research for animal pose estimation. They mention the lack of publicly available datasets and why 3D-generated, synthetic data can be a way to deal with the difficulty of collecting big datasets with enough variety in light, camera angles and poses. They present different metrics that can be used to evaluate pose estimation models. The one that is used by most toolboxes and frameworks is the *RMSE*. It is suitable even for very precise models, where other metrics such as *PCK* would not be able to catch slight differences in performance [23].

Synthetic datasets in practice are presented by Fangbemi et al. [14] and Shooter, Malleson, and Hilton [45].

In [14] the authors train a *ResNet-50* to infer 3D-animations of a skeleton from 2D videos of cougars. For this, they use the model of a cougar that is rendered from different camera angles and add a random photo as background. To bridge the domain gap, they try to techniques: The first on is a style transfer applied to both training and test data, and the second one is to convert both training and test images to greyscale. While the latter led to a significant improvement of performance, the former performed worse than using the images without any augmentation. They report success in this task, but also that the model will often struggle when presented complex videos with occlusions or mutiple objects.

Shooter, Malleson, and Hilton create a synthetic dataset of dogs in the game engine Unity3D. They use various models and textures for that model, which is then placed in a 3D environment. They apply post-processing effects such as adding grain and changing the saturation and brightness of images randomly, to create a bigger variability in the

dataset. They compare pose estimation models trained solely on synthetic data, fine-tuned with real data, and trained on mixed datasets. They report that the models trained on synthetic data only perform poorly, while fine-tuning led to an improvement. The best results were archieved on a mixed dataset.

One thing that becomes apparent in synthetic datasets is the lack of directional light; 3D models are rarely illuminated by lightsources that will throw shadows or can be reflected by the coat of an animal. Also the 3D environment is either not existant or barely more a than a textured plain. Therefore, the created images lack the sharp contours direct sunlight will often create in real images taken on the outside, and the contrast between shadow and illuminated areas. It is possible that more realistic 3D environments would lead to an improved performance of networks trained on synthetic data and could be an even more valuable addition in domains where real data is sparse and hard to collect.

4

PoseCraft: A framework for synthetic data generation

In this chapter, the python package *PoseCraft* will be presented. It was developed as part of this work to provide framework to create synthetic datasets for pose estimation. This chapter will describe the architecture of the package, give a short example how it can be used and list some options for improvement in the future.

The package is set to be released publicly[1] after some clean-up work and improvement of stability.

4.1 | Motivation

Synthetic data for Computer Vision tasks is not a new concept as described in the previous chapter; there has been a number of research projects that used 3D render engines to generate datasets that can be used to train neural networks. Frameworks for synthetic data generation exist based on software like Blender [51], Unreal Engine [55], or Unity [57]. A well-known example for the creation of synthetic data is the Framework *BlenderProc* [54]. However, while it covers task such as segmentation and depth mapping, it does not support the creation of synthetic data for pose estimation natively.

Blender is not only a very flexible and powerful tool, it is also free software[2] released under the GNU-License which grants any user the freedom to use it as they see fit and offers a *Python* interface through the package `bpy` [52]. As many machine and deep learning projects are done with Python, most potential users would already be familiar with that language and dont have to learn another software that offers extensive functionality such as game engines. Addiditionally, *Blender* is a lot more lightweight than

[1] github.com/aylue/PoseCraft

[2] The definition of free software is explained by the GNU Operating System: *"Free software" means software that respects users' freedom and community. Roughly, it means that the users have the freedom to run, copy, distribute, study, change and improve the software. Thus, "free software" is a matter of liberty, not price.* [16]

full game engine. Based on this and previous *Blender* experience of the author, it was chosen for this thesis.

4.2 | Approach

The goal is to create a frame work that allows the user to randomly assembly assets to 3D scenes, as this has proven to be effective for synthetic datasets. Users should have more options to create more realistic lighting conditions and 3D environments, without losing the advantage of variety by randomly selected backgrounds or models. So, instead of using random plain images as background that will rarely fit the camera angle, lighting and proportions of the 3D-scene, the package should incorporate the option to use a more realistic lighting by the combination of a 360° HDRI background image or skybox for the whole scene (the `World` in a Blender file) and one or multiple light sources that will emit directed light that matches the world. The user is also encouraged to create more realistic 3D environments than just textured planes; to take full advantage of those, the camera will not just move around the object, but instead the object can be moved to random locations within the scene. To avoid being placed to far away or clipping into other meshes of the environments, the user can create an object called `PlacementArea` in the 3D scene, and random coordinates to place the object on will only be created in the volume or the on the faces of this object. The camera should be rotated in a way that it is always directed towards the object. To create more variability, a random offset within a user-defined range is applied everytime to both rotation and translation. To make the package actually useful for pose estimation, it offers the option to pose a rigged object into a randomly selected pose. These poses have to be predefined by the user. In this thesis, a new pose was chosen each time the horse model was moved to a new coordinate. The keypoint object that is used should have invisible `Empty` objects attached, that are used as keypoints. In summary the user needs to provide the following assets:

- One or multiple 3D meshes connected to an armature for which the pose estimation is done
- One or multiple Blender files that contain at least a camera and a mesh called `PlacementArea` for the random coordinates
- One or multiple Blender files that contain a `World`. This can be the default grey background, a skybox, or a 360° HDRI image, depending on the needs of the project. Any lightsources in the scene will also be used.

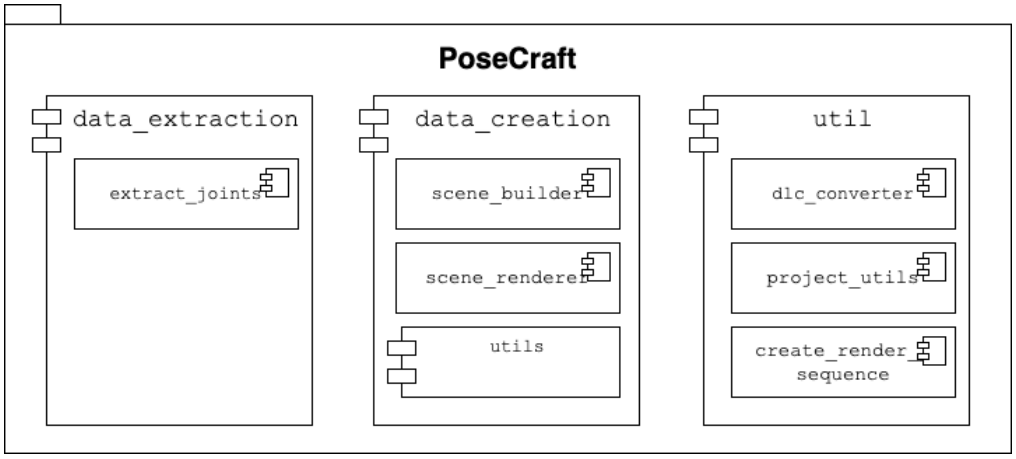


Figure 4.1: Diagram of the architecture of PoseCraft.

- One or more `.json` files that contain information about how the armature is posed. The user can extract them through a Blender plug-in, after posing the model by hand prior.

These assets are placed into a project directory. The user can then write a script using the library and decide how many of the possible transformations should be applied. An example for such a script can be found in the appendix of this thesis in Listing B.1. A list of random combinations is then created, assembling model, 3D scene and background/lights into a single file. After that, the script can be executing `python scriptname.py` and the render process will start. The images are saved alongside with a `.csv` file that contains the image coordinates of the keypoints.

4.3 | Architecture of PoseCraft

In this section, the different modules of the `PoseCraft` package will be described. Currently the package is meant to be used by importing it and incorporating its functions into a script that can be run. An overview can be seen in Figure 4.1.

Data Generation. The main functionality of the package, which is the generation of synthetic images, is found in the `data_creation` module. The module contains two classes (`SceneBuilder` and `SceneRenderer`) that help with the composition of the *Blender* files to create a dataset and will render the images. Some auxiliary functions can be found in the `datacreation.utils` submodule. Those auxiliary functions are for example the calculation of how the camera must be positioned to face the target object,

transform dataframes or reading data from the configuration files. The `SceneBuilder` class will create the scene by loading the objects from other files and create the randomized coordinates where the objects will be placed. The `SceneRenderer` class will then select a pose for the object, render the scene and save the images and annotations.

Data Extraction. This submodule is currently containing only one file, which will extract the bone rotations of an armature object to a `JSON` file. Currently this is only implemented in form of a *Blender* addon, which can be used by copying the code of the module into the `Script` window in the GUI application. While the addon should be available too to make the usage easier, this module will be reworked to allow it to be used in a script, so a user can for example extract the bone rotations of armatures from all files in a directory.

Auxiliary Functions. The `Utils` submodule contains three files that help with the project management.

- `project_utils.py`: Contains functionalities such as the creation of a new project directory.
- `create_render_sequence.py`: Contains a function that creates a `.csv` file of different combinations of all available objects, 3D environments and lighting conditions.
- `dlc_converter.py`: Can be used to copy all or a subset of generated images into the directory of a *DeepLabCut* project, which is used later in this thesis to train a neural network for pose estimation. It will also convert the `.csv` files with the annotations to the format used by *DeepLabCut*.

4.4 | Development

4.4.1 | Designchanges during Development

During the development of PoseCraft, there were some changes on design decisions that were initially made.

Running as a Blender package. Initially the idea of PoseCraft was based on a script that was executed within the system wide installation of Blender. Blender can be run from the commandline, with the option to pass Python scripts that will be executed

upon startup (still using the *bpy* Python-API). However, this is not only very inconvenient to use, it also affects the system installation of Blender and makes it complicated to use packages that are not included that are not included in the Python distribution that is installed with Blender. Therefore, the framework was then changed to be executed in Python, allowing for the use of virtual environments[3]. To make it even more userfriendly, it was then further restructured to a python package that can be installed on a users system.

Moving away from configuration files. In a first approach, the creation of the 3D scenes was heavily reliant on configuration files. The idea was to save diskspace by reusing the same 3D environment but different placement of the coordinates and camera settings in it. However, this was prone to errors as the creation of configuration files was not automized but done manually via copy-and-pasting and then adapting them. To avoid this, the configuration files were scraped all together and each 3D scene needs to contain a camera and a mesh which is used as ermitter for the random coordinates.

Usage of proper logging. At first, most information for the user was output through `print()` statement. However, this is not suitable especially when the rendering process should run on remote machines, as the output might become unavailable upon disconnection. Therefore, the `logging` package was introduced to make the logs more structured and most important save them into files.

4.4.2 | Challenges and Solutions

During development, different problems were encountered and solved. Some of them are described here.

Wrong calculation of 2D keypoint coordinates After the pipeline was generally working (i.e. composition of scenes, placing the keypoint object etc.), plotting the keypoints on the images showed that their positions were calculated wrong. This had multiple reasons, at first a mistake in the calculation of the world coordinates for the keypoints (giving false coordinates) and then using the wrong object property as coordinate (technically correct calculation of keypoints, but their location was the rest pose, and not their actual position in the image after the mesh to which they were attached was transformed). The correct way to get the position of a keypoint `kp` is as follows:

[3]Virtual Environments function like independent Python installations; therefore, collisions between installed packages or their version can be avoided

```
bpy_extras.object_utils.world_to_camera_view(bpy.context.scene, camera, kp.
    matrix_world.translation)
```

Listing 4.1: Get camera space coordinates for a Blender object in bpy

Important is using `kp.matrix_world.translation` and not just the world matrix if a object is translated because the parent object to which it is attached is transformed.

Mathematically correct, but visually inaccurate keypoint positions. After being calculated correctly, the keypoints were sometimes still slightly off. This was caused by the parented object: At first, the vertices that are used to calculate the position of a keypoint in the image were attached to the armature. However, depending on the distance to the parent and how much the parent bone was rotated, the vertices would not be displaced in the same way as the mesh and therefore showing inaccurate positions. This was solved by attaching the keypoint vertex to the closest vertex of the mesh instead of an armature bone.

Importing the HDRIs as world background from one file to the other One bug that took a while to figure out was an error when trying to import the `world` data block from one file to the other. This happened because at first it was implemented to use the `append` function, that is used to copy objects such as meshes or lights (see Listing B.1). This does however not work for worlds, so that no data at all copied to the mainfile. It was resolved by doing it as shown in Listing B.1.

GPU activation in Docker When running the code in Docker, the process did not use the GPU but rendered the images on the CPU instead. This happened despite explicitely activating the GPU and selecting *Cycles* as render engine in the code.

This had two reasons:

- The GPU not explicitely being selected when the container was started from the image
- The CPU not being disabled and therefore Blender still using it as primary device over the GPU.

The first issue was solved by using the `--gpus` flag when starting the container. For the second reason, it was necessary to explicitely disable the CPU in Blender after selecting the GPU as render device and save the settings. Code for this is shown in Listing B.1 in the appendix.

4.4.3 | Future Work

The range of functions should be further improved to increase user friendliness and make working with the package easier.

Calculation of occluded keypoints. As it is sometimes better to not label keypoints that are not visible in the picture, the package should offer to calculate whether an object is occluded and then leave the coordinates for that keypoint empty.

Save Information about the assembled scene in the directory where the images are saved. As each combination of model for pose estimation, background and 3D scene gets it's own directory where the images are saved, it would be beneficial to save which assets where used for each directory and offer a function that can count how many time an asset was used, if a user wishes to do a statistical analysis of the dataset.

Less parameters for the functions. Right now, the API required a lot of parameters for many of the functions, that should not be necessary as the information could be sourced elsewhere.

5

Synthetic Dataset Generation

In the previous chapter a python package to generate synthetic data for keypoint detection was introduced. This chapter will present how the python package was used to create a dataset that can be used to train a neural network for keypoint detection on horses. For both environment and horsemodels, the design criterias will be discussed and examples presented. A list of all 3D assets used as well as information on the *Poly Haven* Library can be found in Appendix C.3.

5.1 | Horses

Figure 5.1: A selection of models to represent the variety of horse types in the dataset.

Horses come in many different shapes, sizes and colors. However, in datasets euro-

pean warmbloods and thoroughbreds are often overrepresented, despite the fact their phenotype of athletic build and an often dark coat has a rather low variability and many other types of horses are underrepresented. Since it can be hard to get real world data of specific horse types, the 3D-Models of the horses for this dataset are supposed to cover as many different phenotypes as possible.
To get a usable horse model, at least the following things are needed:

1. Creation of a 3D base model
2. Rigging the model to allow for posing
3. UV-Mapping the model to allow for texturing
4. Create different textures to represent different coat colors and patterns

At this stage the model will look like a horse, but it will not be very realistic. Since lighting conditions can be very different and the fur of animals can look very different depending on the lighting, the models were also equipped with particle systems to create more realistic interactions with light. For mane and tail, hair particle systems were used aswell instead of modeling them as meshes.
This made a few additional steps necessary:

5. Create a hair particle system for the body fur with the `Quick Fur` modifier in Blender
6. Create a hair particle system for the mane and tail
7. Brush the curves that are used as guides for the particle system to create a natural look
8. Create a weight map for the particle systems to influence the length and density of the fur according to the body

5.1.1 | Mesh

As foundation for the 3D Model a public model distributed under a free license was used, which is depicted in figure C.1.

The base mesh was then remeshed and refined using the *Sculpt Mode* in *Blender*. This way the resolution was increased, allowing for more details. Further, this helped to smooth out edges and cornes when the model was posed with its armature. The tail was cut of and replaced by a short cylinder to simulate a real horstail. This mesh

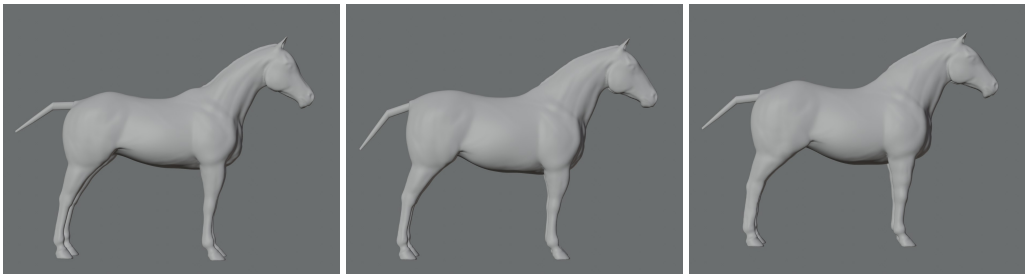


Figure 5.2: Variations of the adapted horse model.

is used as base where hair particles can be attached to. The different variations of the horse model can be seen in figure 5.2. All models share the same UV-Map and were not remeshed further, so they dont need their own textures.

To be able to precisely track the keypoints, additional objects of the type `EMPTY` are attached to the mesh onto the closest vertex. Those will not be visible rendered, but their coordinates can be extracted and transformed to 2D image coordinates. A list of the keypoints is also given later in Table 5.1.

5.1.2 | Rig

To be able to pose the mesh to represent different stance phases during trot, the mesh has to be rigged. *Blender* offers the Plug-In *Rigify*[53]. This plugins comes with multiple Metarigs, which are ready-to-use armatures that can be bound to a mesh. Some of the bone groups had to be slightly translated and rotated to better fit the mesh, but other than that the rig could be parented to the base mesh with automatic weights[1] and no further adjustments. The mesh and the rig can be seen in figure 5.3.

5.1.3 | Textures

For the dataset, a total of 14 different textures were handpainted. Since it proved to be difficult to unwrap the mesh into a well structured UV-Map[2] in *Blender*, another approach was chosen: The mesh was exported to an `.obj` file and imported into *Procreate* on a *2020 iPad Pro*. In *Procreate* it is possible to paint a base color texture directly on the 3D mesh and use multiple layers while doing so. This made it easier to deal with chaotic

[1] Weights are used to determine how much a bone affects a specific area of the mesh. For really good results and with enough experience a manual adjustment of the weight map could yield even better results, but the automatic settings worked well enough.

[2] A UV-Map is used to assign pixels of an image to the corresponding faces of a mesh. However, for complex meshes with many faces, it can be difficult to unwrap them without creating a lot of ”isles”, which refers to many small groups of neighboured vertices instead of a few big groups.

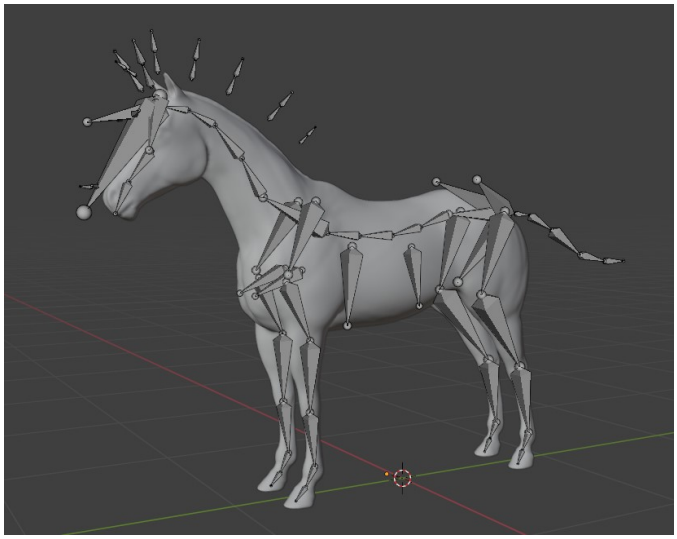


Figure 5.3: One of the horse mesh variants with the attached rig.

UV-Maps. After that, the texture can be exported as an image and used for the model in *Blender*. Each texture consisted mostly of three layers:

- Base color
- Marking/coat patterns
- Light and shadow effects

The base color gives the horse its main color and is just a solid color. The second layer contains markings like socks, blazes, spots or flecked hairs which can be combined with all the base colors. The third layer contains light and shadow effects to give the horse a more plastic look and is applied to all coat variants. As example, those three layers are shown in Figure 5.4.

5.1.4 | Fur simulation

As previously mentioned, the models were additionally equipped with particle systems to simulate fur and hair. This creates a more realistic look and allows for more realistic interactions with light.

Using the `Quick Fur` modifier in *Blender*, a particle system is added to the body mesh, which creates a fur-like appearance. In the *Sculpt*-Mode for curves, the curves that are used as guides are then brushed to lay flat on the body and account for typical hair swirls in horse coats. The visible hair is then interpolated between the hair curves.

Figure 5.4: Examples of layers that form a texture. The first image is the base color, the second image adds markings and coat patterns, and the third image adds some basic light reflections.

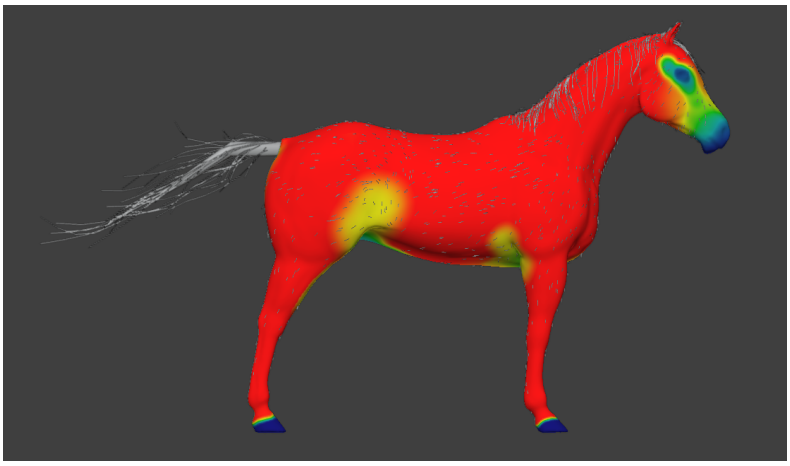


Figure 5.5: Weight map for the horse model, used to determine the density of the hair particle system at certain places.

The density (the amount of hair) is set by a weight map. The weightmap for the model is depicted in Figure 5.5. Red means a very high density of hair, while dark blue stands for no hair at all. This way, the hooves and certain parts of the body such as nostrils and eyes are not covered in hair. As base color, the base color of the underlying mesh were the hair is attached to is used. A similar weight map is used for the length of the hair aswell. This allowed to simulate the longer hair on the legs some breeds such as Irish Cobs have, that will hide the contours of the leg/fetlock and pastern area.

The impact of the hair particle system on the light effects can be seen in figure 5.6. It compares the appearance of the horse model with and without the hair particle system rendered. Environmental light from the `World` has mostly an impact in form of the color that is reflected by the hair. Light sources such as a `Sun` will lead to stronger light reflections based on their angle and the strength of the light source. The darker the

Figure 5.6: Horse model up close. On the left, no hair particles are rendered, on the right, the particle system is visible.

basecolor, the more the light reflections become visible. The particle system also adds a nice textured look to the model, making it look less cartoonish. The material settings also have a big impact how the hair looks like. Here, the `Principled Hair BSDF` shader was used to archieve a realistic look. The settings of the shader vary between the different models and are not uniform.

However, the quality of the hair rendering was heavily bottlenecked by the computer used to create the models; *Blender* would repeatedly crash when working on a model that had a very high density of hair. To make up for this, the density was kept lower. To still cover the body, the hair length and thickness had to be higher than it realistically would be.

5.1.5 | Poses

The base armature was posed into 21 different poses, which were created using video frames of moving horses as reference to make them realistic. The bone rotations were then exported into a JSON-File and saved in the project directory. Some examples of the different poses for the horse can be seen in Figure 5.7.

5.1.6 | Horses for the Dataset

With all these modification, a total of 19 different horse models were created by combining the 3 rigged mesh variants with the 14 texture variants. All models used can be

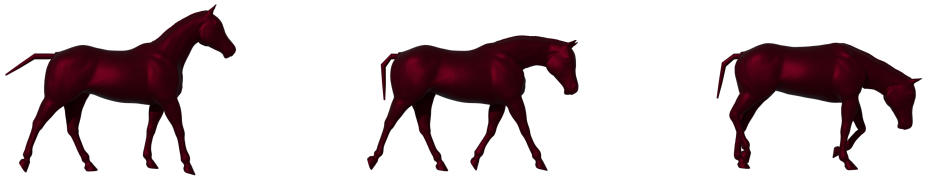


Figure 5.7: Three examples of poses that were created for the dataset

found in Appendix C.1.

5.2 | Environment

This section will give some insight into the design choices and creation process of the 3D scenes in which the horse model will be placed later on. The dataset features multiple indoor and outdoor environments that resemble the classic structure of real world horse training facilities and are later on combined with a randomly selected skybox to get as much variety as possible without having to create too many different 3D environments.

5.2.1 | Design Criteria

As mentioned in the introduction of this section, the 3D scenes resemble the idea of real world horse training facilities. Usually, horses are lunged in riding arenas (indoor/outdoor) or on round pens, both of which are included in the dataset. There are some characteristics that are very common in these environments:

- Surrounded by a fence. This fence can often be wooden or an electric fence. Especially indoor riding arenas often feature so called kicking boards, which are wooden boards that are attached to the lower part of the fence to prevent horses from kicking through it but allow spectators to see the horse, and riders and horses to still see their environment.
- The standard dimensions of a riding area as defined by international guidelines are 20 m $\times$ 40 m or 20 m $\times$ 60 m, but other sizes are possible. A round pen is a smaller, and, as suggested by the name, round area with a diameter of 15 m to 25 m.
- The ground is usually made of sand, which provides good grip. The other option is grass, especially if a stable does not have a designated riding arena and uses part of the pastures instead.

- The surroundings can be very diverse and cannot be generalized. For outdoor arenas, some kind of plantation is often present, such as trees or bushes.
- Indoor arenas can range from very dark with little to no natural light to very bright with large windows and skylights. Especially skylights can lead to harsh shadows and strong light contrasts, which can be problematic for the neural network to learn from.

The dataset therefore tries to reflect these typical characteristics.

5.2.2 | Creation process

The creation process of the 3D scenes was usually quite similar for every scene. The following steps were taken to create the 3D scenes that are used in the dataset:

1. Starting with a new Blender file, the ground plane is created and scaled to the desired size. How the ground is created in detail is described in section 5.2.2.1.
2. After that, a fence or kicking board is created around the ground from simple cylinder- or cube meshes that are adapted in the edit mode to the desired size and shape.
3. For indoor scenes, a hall is created by adding walls and a roof.
4. The material of the meshes is adapted. To create a more realistic look, for many parts textured materials from *PolyHaven* are used.
5. For the indoor scenes, the wall and/or roof get cutouts that are replaced with a glass material (using the `Glass BDSF` shader) that allows light rays to pass through. For outdoor scenes, add some kind of plantation and other objects that are available in the *PolyHaven* library.

5.2.2.1 | Ground creation

Two types of ground are represented in the dataset: sand and grass.

Sand: The sand ground is created by adding a plane mesh to the scene and scaling it to the desired size (one of the standard sizes). To make it possible to deform it, the plane is subdivided multiple times. With the *Displacement* modifier, the different faces are moved up or down, based on a height map that is created in *Blender* with a cloud-like texture. After that, a material from *PolyHaven* is applied to the ground, so it also has

a realistic texture. In the final step, the sculpting tool is used to imitate how footprints are created in sand and how sand accumulates in corners, for example. Example images of this process can be seen in Figure 5.8.

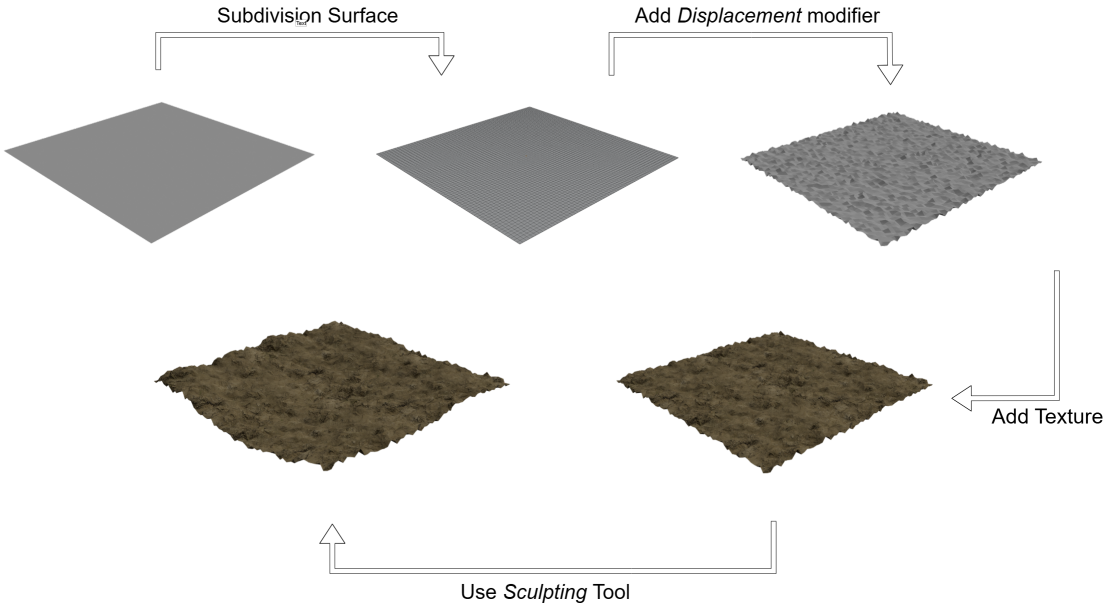


Figure 5.8: Steps for the creation of sand in Blender.

Grass: To create a grass ground, a plane was created and textured with a grass or dirt texture from *PolyHaven*. The grass like appearence is archieved by using a particle system on that plane, which is set to use meshes of little grass blades as particles that were also imported from the *PolyHaven* asset library. It should be noted that this type of ground does increase the rendering time of the scene by a large margin, as the density of the grass particles/the number of children rendered has to be quite high.

5.2.2.2 | Skybox and Light

All the backgrounds were saved into seperate files, so they can be used with all the different 3D scenes created prior. To create realistic environments and lighting condiftions, 360° HDRIs[3] from *PolyHaven* were used. Those images range from pictures of the plain sky to scenes in nature or urban environments. This way it was possible to add a lot more variety to the background of the rendered images with just a few different 3D scenes. The HDRIs were downloaded in a *4k* resolution, as this was the smallest possible resolution that did not lead to visible pixelation in the background.

[3]High Dynamic Range Images

While the skybox will already provide light that matches the background in temperature and intensity for each direction on the surface of the objects, it will not produce shadows. To compensate for this, in each scene that features directional light like outdoor scenes with sunlight, a light source was added to create shadows. The nly light-source suitable for this task is the `Sun`, as it creates directed light everywhere in the scene, so it doesnt have to be adapted to the composition of the 3D scene it is eventually placed in. The direction of the sun and the intensity of the light as well as its color are set to match the HDRI skybox. For example, a background featuring a sunset will have a light source of a slightly orange color and a very low angle, so the shadows are long and the light is not too harsh.

5.2.3 | Created Assets

The finalized 3D scenes used for the dataset generation are two indoor scenes and four outdoor scenes, one of which is a Round Pen. For almost all scenes, there are mutliple versions with slightly different objects placed in the scene, a different lens for the camera or a different position of the area that is used to create the coordinates where the horse will be placed later on. In addition to the 3D scenes, 14 additional files containing a skybox and matching lights were created.

Some examples of rendered images of the 3D scenes with a HDRI skybox and light added can be seen in Figure 5.9.

5.3 | Dataset

The resulting dataset consists of 95 different combinations of environment, horse model and background image. For each of this combinations, 10 images were rendered, resulting in a total of 950 images. For each of this images, a predefined pose for the model was randomly selected. The keypoints that were calculated for each image are listed in Table 5.1.[4]

Unfortunately, there is no complete list of all the combinations of environment, horse model and background image that were used to render the images, more information on that is described in Section 5.3.2.

[4]The english translation of body part names is rather inconsistent; for example, the carpus is colloquially called knee. Therefore, some of the keypoint names might be confusing as they were directly translated from the german word for that body part.

Figure 5.9: Examples of the 3D scenes created for the dataset. The images show the scene with a HDRI skybox added.

Bodypart	Keypoint name	
Mouth	`kp_mouth`	
Poll	`kp_neck`	
Withers	`kp_withers`	
Highest Point of the Croup	`kp_croup`	
Tailbase	`kp_tailbase`	
Elbow	`kp_elbow_l`	`kp_elbow_r`
Carpus	`kp_carpal_l`	`kp_carpal_r`
Fetlock Frontleg	`kp_fetlock_front_l`	`kp_fetlock_front_r`
Point of Hip	`kp_hip_l`	`kp_hip_r`
Stifle	`kp_knee_l`	`kp_knee_r`
Hock	`kp_hock_l`	`kp_hock_r`
Fetlock Hindleg	`kp_fetlock_hind_l`	`kp_fetlock_hind_r`

Table 5.1: A list of all the keypoints calculated for each image in the dataset.

5.3.1 | Computational Cost of the dataset generation

The dataset was rendered using a Machine learning server with a *Intel(R) Core(TM) i5-7500* CPU, a *NVIDIA Titan V* GPU and *32 GB* RAM running `Ubuntu 22.04.4 LTS`. The *PoseCraft* package was installed in a *Docker* container based on the `python:3.11-slim` image. The dataset was rendered in multiple batches. For 95 different combinations

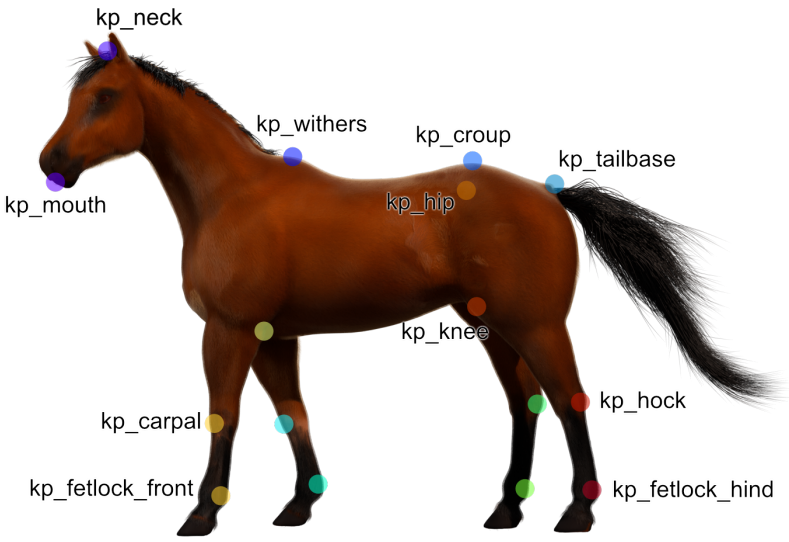


Figure 5.10: Keypoints attached to the horse model.

of environment, horse model and background image with 10 rendered images each, the sum of render time is approximately 20 hourse based on the directory timestamps. However, the amount of time needed is highly depended on the 3D scene. For some scenes, 10 images were rendered in less than 10 minutes, other scenes took more than 60 minutes for the same amount of images. The GPU was also shared with another unrelated *Docker* container for an unknown amount of time, which might have increased the render time.

5.3.2 | Limitations and Opportunities for Improvement

While the dataset generated was sufficient to train a network as described in Chapter 4, there is room for improvement. This subsection will describe some issues with the generated dataset, that were discovered too late to improve them in time for this thesis but should be considered in future work.

More keypoints. To make the images more useful for other tasks, more keypoints should have been added, even if they are not used for this thesis. For example, the hooves and the shoulder joints should have been included, as it is common to use annotate these keypoints in other datasets.

More environments. It might have been benefecial to crete more unique environments, even though it is not proven that this would actually improve the performance of the network.

Bias towards sunny lighting conditions. The dataset shows a bias for lightboxes that resemble sunny weather. While strong sunlight can lead to challenging lighting conditions and it is important to train the network on these, cloudy weather with less environmental light is most likely underrepresented in the dataset.

Optimizing the Armature for the horse model. The armature was not optimally adapted before it was parented to the mesh. In this process, the bones were mostly rotated and translated to fit them into the mesh. More scaling would have been good, or, alternatively, manual adjustment of the weightmaps for the vertex groups.

Occlusion of keypoints. The *DeepLabCut* userguide recommends to ignore keypoints that are invisible in the image and add no coordinates for them. While this is easy if keypoints are annotated manually, the dataset generation process does currently not allow for this in every case. Keypoint coordinates are removed in two cases:

- The keypoint is out of the image boundaries, e.g. if the horse is too far to the left or right, or the camera is zoomed in very close.
- Since the horse will always be visible from a lateral view, some keypoints are always invisible, as they face the outside and are hidden by the torso of the horse. The keypoints where this applies are **elbow, hip** and **knee**. They are removed for either the left or the right side of the body, depending on the the overall rotation of the horse towards the camera (the rotation switches between 90 ° and -90 ° after every rendered image).

However, there are cases where the keypoints of the outer legs are occluded by the corresponding inner leg. In a video, this will only be the case for very few frames, but it does happen. It was not possible to remove these keypoints in an automted way after the dataset generation was finished. This has to be done in *PoseCraft* and a check for visibility should be added in a future version (see Section 4.4.3).

Statistical analysis of the dataset. Due to an user error, the CSV-File containing the combinations of each render batch were not saved but instead overwritten on every

new run. Therefore, the information how many times each asset was used is unfortunately unavailable. In a future version of *PoseCraft*, it should be implemented that each directory will also include a file that saves the information about the used assets, so that a statistical analysis of the dataset is possible on whatever subset the user selects (see Section 4.4.3).

6

Deep Learning for pose estimation on horses

After the last few chapters were mainly focused on the creation of a dataset, in this chapter we will put this dataset to use. The goal is to create a neural network (at least partially) trained on synthetic data, that can detect keypoints on horses in real-world videos. For this, the python library *DeepLabCut* is used, which is a framework for training neural networks for keypoint detection on animals and humans. In this chapter, the dataset creation will be described in more detail, as well as the training of the neural network and the evaluation of the results.

6.1 | Training- and Testdata

This section will take a look at the dataset used to train and evaluate the network. In addition to the synthetic dataset generated with *PoseCraft*, a number of real-world videos was collected to evaluate the performance of the network on real-world data. In Figure 6.1 an example of the annotated images from both synthetic and real-world dataset is shown. More images from both datasets can be found in Appendix A.1.

Since the synthetic dataset was already described in detail in section 5.3, this section will primarily focus on the collected real-world data and any type of pre-processing and annotation process of the data.

6.1.1 | Recorded Videos

The videos were recorded at different locations and with different horses. Some of the videos were provided by the *Hanse Equine Hospital* and taken as part of the lameness diagnosis process of horses presented to the veterinarian. Other videos were recorded by the author of this thesis or friends and acquaintances of the author. Everyone willing to participate was provided with a list of instructions:

Figure 6.1: On the left an annotated image from the synthetic dataset, on the right an example of a video frame from the real-world dataset. The the Ground Truth keypoints are plotted onto the image with *DeepLabCut*.

- The horse should be lunged in a circle, the video recorded from the center of the circle.
- For the head any equipment such as a bridle, a halter or a cavesson is fine, but the body of the horse hsould be visible and the horse should not be waring equipment such as a saddle or a rug
- The video should be recorded in landscape mode and at least in *HD* resolution ($720p \times 1280p$) or higher.

All videos were recorded with a smartphone camera, hand held by the person lunging the horse or by a second person standing next to the lunge line handler in the middle of the lunge circle.

The final dataset consists of 28 videos that are recorded in *HD* ($720p \times 1280p$) or higher resolution. Videos of higher resolution were downscaled to a *HD*, as this quality should be sufficient for the the analysis, but keeps the computing time reasonable. To get a balanced dataset in regards to horses being lunged on both left and right hand, some of the videos were mirrored horizontally to get an even distribution. The length of a video did not matter, as the number of frames extracted from the video is the same for every video (20 frames). For the extraction of frames, the default settings of the *DeepLabCut* package were used, which extracts frames by *k-means* clustering to get frames of as much variance as possible from the video. However, some of the extracted frames were removed later on, because the horse was filmed from an angle that the network would have had no chance to properly learn.

6.1.1.1 | Common errors in recorded videos

From initially over 40 videos, unfortunately a lot of the videos had to be discarded as they did not meet the criterias for the dataset. Most common reasons for discarded videos were:

- The video resolution was too low. Especially if videos were sent via an instant messenger app instead of being uploaded to the provided cloud service.
- The horse was equipped with a rug or a girth and saddle pad. While this is common in the day to day training of horses, there is a chance that important keypoints are occluded and a horse would not wear this in a clinical examination situation. Therefore, the network is not required being able to handle this kind of additional equipment.
- Video was filmed in portrait mode rather than landscape mode.

6.1.2 | Annotation of Videos

For both synthetic and real data, the *DeepLabCut* GUI was used to make and validate annotations.

6.1.2.1 | Real-world videos

The video frames were annotated through the *DeepLabCut* GUI, which features a tool to annotate extracted frames of a video. Each of the visible keypoints was annotated. For the hock joints, the tail was sometimes in front of the joint, in which case the keypoint was still annotated, primarily to keep it consistant with the synthetic dataset, where joints occlouded by other body parts were also annotated. The outer leg pair being occlouded by the inner leg pair was annotates if at least a small piece of the joint was visible, otherwise the labels were left empty.

6.1.2.2 | Synthetic dataset

After being imported into the *DeepLabCut* project, the synthetic dataset and the corresponding annotations were checked in the previously mentioned *DeepLabCut* GUI. No changes were made to the annotations with two exceptions:

- The frame being so dark that the horse was not visible; some or all keypoints for this frame were removed.

- Especially the smaller horse models were sometimes positioned in a way that one or more of the fetlock joints were not visible, as they were sunken into the uneven sand surface. As this was pretty severe in some cases, the keypoint on that joint of all affected legs were removed.

6.1.3 | Data Augmentation

As mentioned in Section 3.3, synthetic data often suffers from a domain gap, which can lead to poorer performance. Augmentation techniques to add more noise and imperfections are therefore recommended to make the synthetic data more realistic. As different light conditions were already covered in the data generation itself, the main concern would be to add noise and blur to the rendered images. *DeeplabCut* offers both of these techniques as well as additional scaling and rotation to be applied during the training of the neural network. The augmented results are only available in memory and not saved to the disk as images, therefore no examples can be shown here. This made an additional step to further augment the dataset unncessary.

6.1.4 | Datasplits

To be able to evaluate both the performance of the network in general, but also the quality of the synthetic dataset compared to a real one, different splits of the dataset will be created to train networks on.

- Train the network solely on synthetic data and compare it against real-world test data
- Train the network solely on real-world data and compare it against real-world test data
- Train the network on both synthetic and real-world data at the same time and compare it against real-world test data
- Train the network on synthetic data and then fine-tune it on real-world data and compare it against real-world test data

The number of images per split and in sum is shown in Table 6.1

While some horses or environments are present in multiple videos, the splits were made in a way that no horse or environment is present in both training and test data. This ensures that the metrics reflect the network's ability to generalize to new, unseen data, rather than memorizing the training data.

Table 6.1: How the data was split and how many images each split contains.

Split	# Images	Train fraction
Synthetic Data	932	0.74
Real Data	200	0.34
Synthetic and Real Data	1132	0.77
Test Data	333	
Real-World Data	533	
All	1465	

6.2 | Training

As already mentioned in the introduction of this chapter, the training of the model was done using the python library *DeepLabCut*. As this library is explicitly designed not just for computer scientists, but rather interdisciplinary research, it simplifies a lot of the steps that would otherwise be necessary to train a neural network. In earlier versions, both *TensorFlow* and *PyTorch* were supported, since the release of version 3.0 users are heavily encouraged to use the *PyTorch* backend as *TensorFlow* will be deprecated in the future. This influences especially the available model architectures and some of the training parameters, but the general workflow remains the same.

For this thesis, three different model architectures were tested. Two of them are standard *ResNet* architectures (`ResNet-50` and `ResNet-101`). The third one is a *HighResolution Network* (`HRNet-W48`), which often tends to perform better on tasks such as pose estimation, as pixel accuracy is more easily archieved if details are preserved rather than lost during the downsampling of the image. Since this task only features a single horse in the image and does not require the detection of multiple individuals, all networks are used in their default *Bottom-Up* version.

As described in the previous section 6.1.4, the architectures should be tested on different training data sets aswell to see how well the synthetic data can imitate real-world data and if the synthetic data in addition to the real-world data can improve the performance of the model. As the *ResNet-50* models performed very poorly on both synthetic and real data, they were not further used and only the deeper networks were trained on the mixed and finetuned datasets.

6.2.1 | Training Parameters

Most of the training parameters were left at their default values, as similar papers have already shown that these hyperparameters can produce good results[15].

Weight Initialization: All of the networks were initialized with weights pre-trained on *ImageNet*[24].

Epochs: The number of epochs was set to 300, as the results started to plateau around 250 to 300 epochs. This is slightly lower than the 400 epochs used in the paper by Feuser et al. [15]

Batch Size: The batch size was kept at the default value of 8.

Learning Rate: The learning rates differed between the different architectures, but were kept to the default values as well. For both *ResNets*, the learning rate was initially set to 0.0005 and decreased to 0.0001 and $1e-05$ after respectively 90 and 120 epochs. The *HRNet* was trained with a learning rate of 0.0001 initially, decreased to $1e-05$ and $1e-06$ after 160 and 190 epochs.

The loss rate of the best performing architecture can be seen in Figure 6.2.

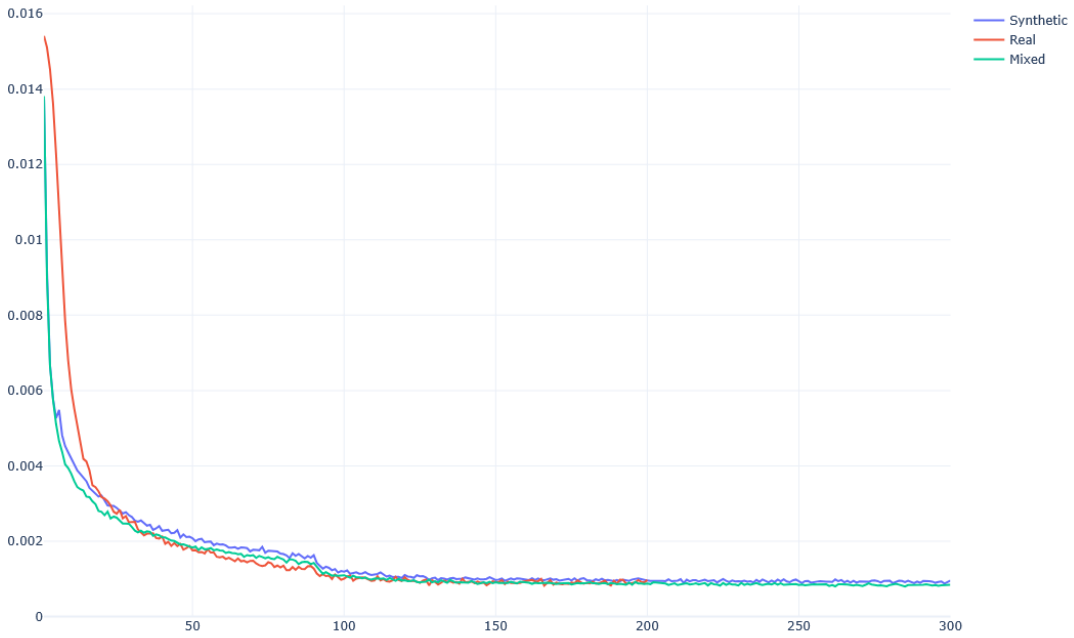


Figure 6.2: Plotted loss rates of *ResNet-101*. The other lossrates were similar.

6.2.2 | Hardware and Training Environment

The training was done on the same server as the data generation, euqipped with an *Intel(R) Core(TM) i5-7500* CPU, a *NVIDIA Titan V* GPU and *32 GB* RAM running `Ubuntu 22.04.4 LTS`. The training was done in a *Docker* container provided by the *DeepLabCut* developers. This container provides an `deeplabcut 3.0.0` installation with a `pytorch 2.5.1` backend.

The amount of time needed to train the networks varied a lot depending on the size of the network and the amount of training data. A more detailed overview is shown in Table 6.2 below.

Table 6.2: The amount of time each network needed to train on the different dataset splits.

Model	Synth data	Real data	Mixed data
ResNet-50	4:10 h	2:30 h	–
ResNet-101	5:00 h	1:40 h	6:20 h
HRNet-W48	9:30 h	3:20 h	11:0 h

The models that were trained solely on synthetic data were trained for an additional 100 epochs on real data to get another version, that can be compared against the models that were trained on mixed data from the beginning.

6.3 | Results and Evaluation

In this section the results that the networks archieved on the test data will be discussed. It is particularly interesting to compare the models with regard to three factors:

- Which model performs best overall?
- How well performs a model trained on synthetic data, compared to real and mixed data?
- How is performance of a model trained on synthetic data and later finetuned on real data, compared to a model trained on mixed data from the beginning?

The models are evaluated both quantitatively using the RMSE and subjectively by applying them to videos from the test dataset.

6.3.1 | Quantitative Analysis

The models were quantitatively evaluated in two ways: using the *RMSE* to evaluate their overall performance, and the error in pixels per keypoint. For the RMSE, the evaluation was done twice; in the first run, all keypoints were considered. However, the evaluation per keypoint and qualitative analysis showed that most errors stem from the keypoints on the legs. Therefore, a second evaluation was done that only kept the keypoints of the upper body, the elbow and the stifle. This was done because the detection of elbow and stifle is so accurate and reliable that the trajectories should be sufficient to be able to calculate the stance phase. The results of the evaluation by RMSE on all trained models can be seen in Figure 6.3. It is important to note that the reported RMSE values are calculated on all keypoints, with no regard to the confidence. If only detected keypoints above a certain threshold were considered, the RMSE dropped significantly, for the best performing models even below 10 over all keypoints. Different values for the `pcutoff` parameter were tried (0.4 / 0.5 / 0.6) but no significant difference between those was found. Since it is not possible to evaluate how many values are dropped for the calculation of the RMSE because the confidence for that keypoint was below a certain threshold. However too many frames with missing keypoints, depending on which keypoint is affected, would make the video useless for analysis. Therefore, those values were not further used for evaluation.

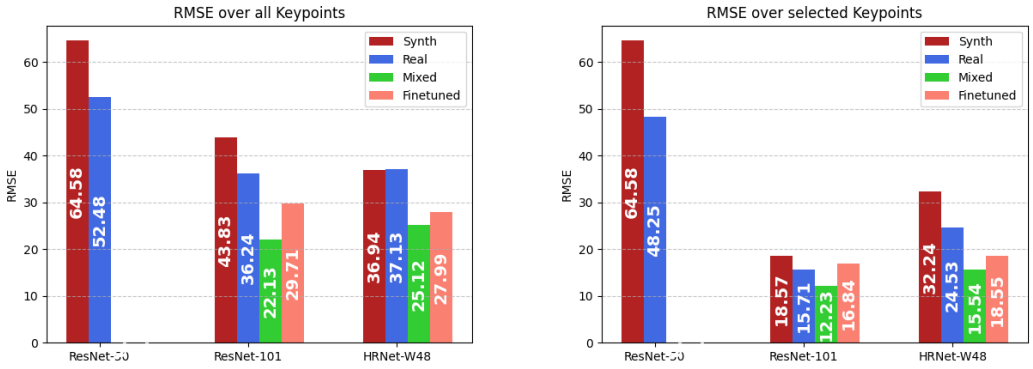


Figure 6.3: RMSE for all trained models. Left shows the error for all keypoints, right only for the selected keypoints.

To get a better understanding what causes the error, the euclidean distance in pixels between prediction and ground truth was calculated aswell. A comparison of the best two model is shown in Figure 6.4. A comparison for each architecture can be found in the Appendix A.3.1. It becomes apparent that all the models struggle with the keypoints

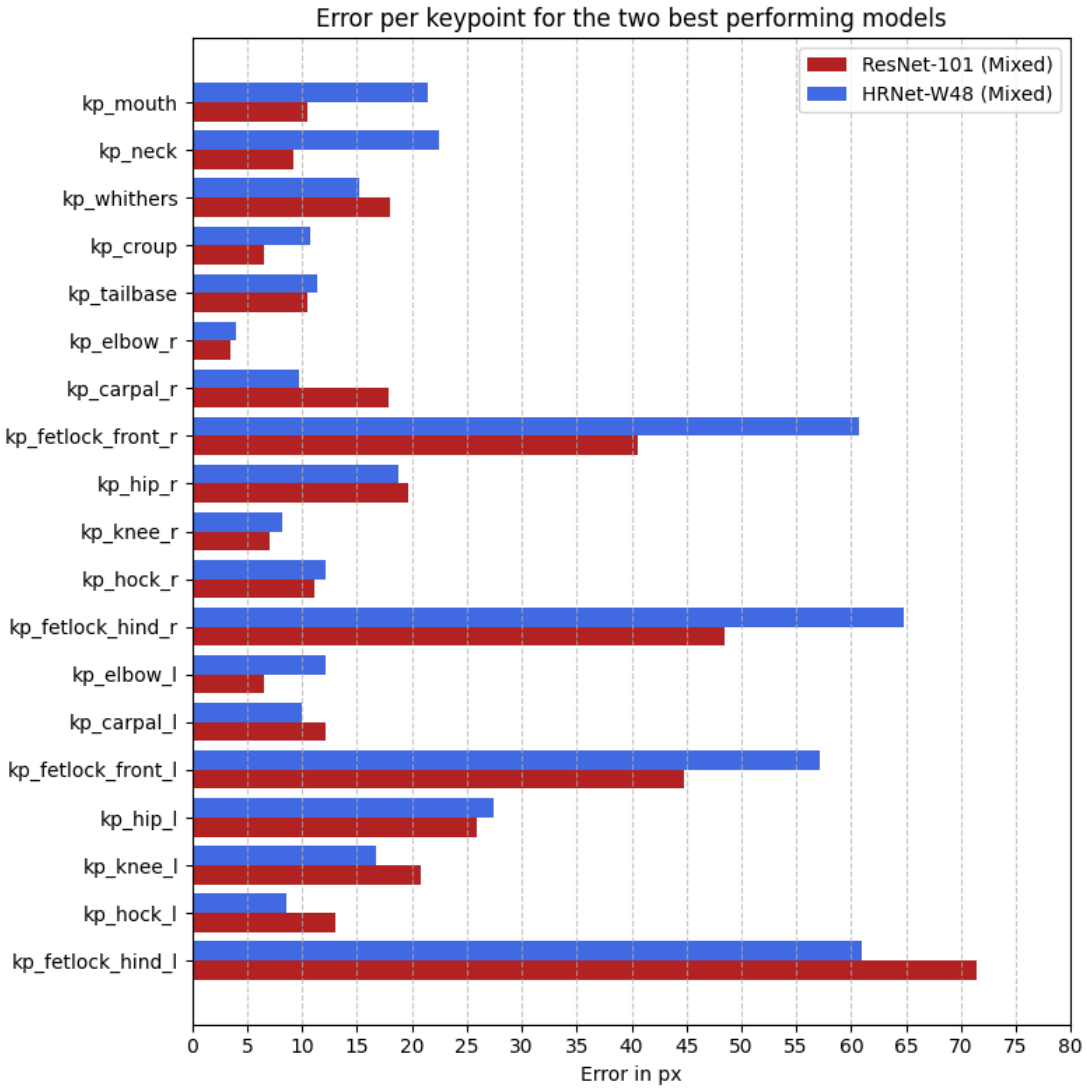


Figure 6.4: Error in pixels per Keypoint and model

on lower limbs of the horse. This is in line with the improvement of the RMSE when those keypoints are ignored for the evaluation.

Overall, the best performing models are the *ResNet-101* and *HRNet-w48* models trained on mixed datasets, with an RMSE of 22.13 and 25.12 respectively.

6.3.1.1 | Qualitative Analysis

For a qualitative analysis, the models were mainly tested by using them to create annotated videos from the testset. A keypoint was drawn into the frame with a probability

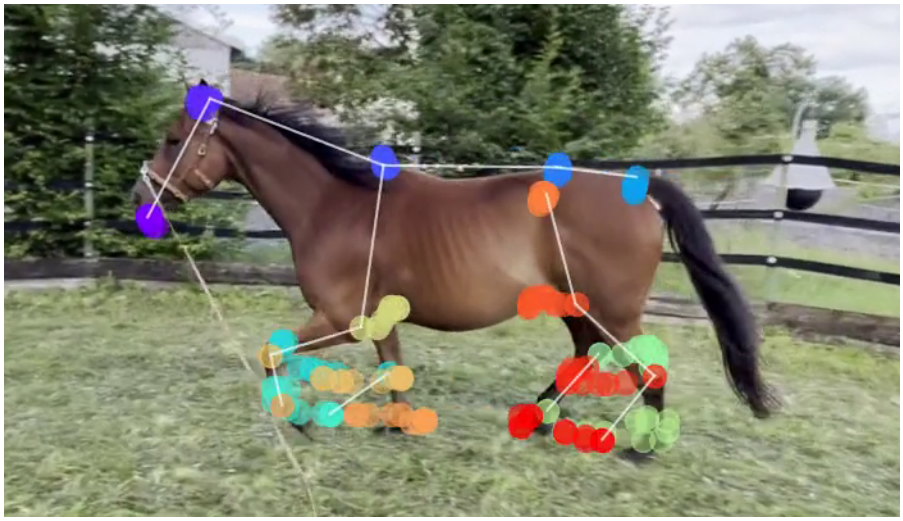


Figure 6.5: A Screenshot taken from one of the annotated videos. The annotation was set to not only show the keypoints of that frame, but create a trail of the position of each keypoint for the past 10 frames.

$p \geq 0.4$. Addiditionally, the series of keypoint positions was filtered, to remove outliers and make it a bit smoother. This did improve the consistency of the annotated keypoints. While the results produced by *ResNet-50* did not seem to deliver useful information, the deeper architectures were able to produced even if trained solely on synthetic data results, that seemed to be usable even though the keypoints tended to jump a little bit to the left/right, especially around the withers. Also the legs were difficult; while a general differentiation between left and right was not a problem, as the elbow and stifle were labeled correctly for each side, the lower joints were repeatedly put onto the same leg or somewhere else entirely. There was no noticeable difference between the *ResNet-101* and *HRNet-w48* that were trained on the mixed dataset and turned out to be the best performing models both quantitatively and qualitatively. Both models delivered results that hat a stable and reliable detection for most of the keypoints. They were even able to get the legs correct most of the time. Keypoints that were placed outside of the horse were very rare for all models.

In Figure 6.5 an example from one of those videos is shown. By showing the keypoints from the prior frames, it is possible to better track the consistency and trajectory of a keypoint. The figure is an particularly good example because it shows both the consistent tracking and periodid vertical movement of the upper body keypoints, aswell as the problems related to the estimation of the limp position.

Addiditionally, *DeepLabCut* offers the option to get the predictions plotted onto the frames that were extracted. The difference in both confidence and accuracy in the de-

Figure 6.6: A comparison of the prediction of the worst (ResNet-50 trained on synthetic data) and the best (ResNet-101 trained on mixed data) model on the same frame (cropped) from the training data set.
$+$: Ground truth
$\cdot$: Model prediction with confidence above the pcutoff value
$\times$: Model prediction below the cutoff

tection becomes apparent in those annotated images aswell. An example can be seen in Figure 6.6.

6.4 | Discussion

Considering a RMSE and pixel error per keypoint that seems to be quite high, the pose estimation on videos works surprisingly well. When comparing error values with other studies, it is important to consider the resolution of the videos: an error of 10 pixels in a 720p video is less significant than in videos with significantly lower resolution. It is possible that this is also related to the poor performance of the ResNet-50 architecture in this work: the network may possibly not be deep enough, or the receptive field at the end of the network may be too small to "see" the entire image in high-resolution videos and view the image areas in their bigger context. Higher-resolution videos offer the benefit that small movements can be detected, which otherwise would be lost. If performance is not a concern (within reasonable limits), than a deeper networks should be the prefered choice. It should also be taken into consideration that the dataset was not randomly shuffled, but seperated by location and horses. It is reasonable to assume that a random split of all images, or at least the occurrence of a location in both data sets, would have improved the metrics in the quantitative evaluation.

When calculating the RMSE on a subset of keypoints, the HRNet's error was almost twice as high as that of the ResNet. One possible explanation for this could be the key feature of HRNets: the preservation of high-resolution features even at deeper levels could mean that the domain gap between synthetic and real data has a stronger influence than with a ResNet, as the difference between the synthetic images and the real images will be most noticable up close. The results of comparing models trained on different training sets are as expected and are also consistent with other research such as [45]: Synthetic data alone does not achieve the same performance as a training set of real data. However, it does lead to improved results when both data sets are mixed and used in training. One possible explanation for this could be that labels annotated by humans are unconsciously set differently than in synthetic data. While this gap is virtually 'lost' in the mixed data set, the network must first relearn this difference. Perhaps a longer training period on the real data would have led to an improvement, and 100 epochs were too short.

Remarkable is the performance of the models trained solely on synthetic data; while models trained like this will often perform poorly on real data and are not able to bridge the domain gap by themselves, in this project it worked surprisingly well. Even if they were outperformed by the other models, the results were still acceptable the performance gap compared to a model trained on real data is almost negligible. This strongly suggests that effort spend on creating high quality 3D assets is not wasted, but will increase the value of synthetic data by a large margin, especially if real data is hard to come by. Carefully crafted synthetic data sets with a focus on realism can be a helpful addition to a dataset for pose estimation.

Future research should evaluate the usability of the trained models for actual lameness evaluations; for this, it is necessary to find methods to deal with shaky camera movements or changes in distance between the horse and the camera and seperate them from the trajectory of the keypoints.

Overall, the results are satisfying and the goals of this thesis were archieved: The successfull training of deep neural networks for pose estimation on horses filmed from the center of a circle with a predominantly good performance and the creation of a syntehtic dataset which is realistic enough to not only improve the model performance when added to real-world dataset, but also produces acceptable or even good results when the model is trained solely on synthetic data.

7

Conclusion

Objective lameness evaluation is a very highly relevant topic and good solutions in this area could lead to a significant improvement in equine wellfare. Video-based markerless methods are particularly suitable for making the application of these methods as simple and accessible as possible. This area of research is an interesting interdisciplinary field between computer science and veterinary research. Therefore fundamental knowledge for both fields was presented in the early chapters. Returning to the research question posed in the introduction, it can be said with certainty that synthetic data is a useful addition to the training of neural networks for pose estimation in horses, as it boosts the reliability and accuracy of the model. For the creation of realistic synthetic data, a new framework was developed in form of a python package, that is not limited to be used in this project, but is adaptable enough to be a suitable solution for other projects. With this framework and manually created assets with a focus on realism, a dataset of over 900 images was generated. Additionally, a dataset of real videos was collected which could be used for evaluation and comparison of various training strategies. These two datasets were used to train a neural networks in different configurations in *DeepLabCut*. The evaluation of the results those networks delivered suggests that deeper architectures can be beneficial to be able to use higher resolution videos for the analysis, which leads to a decreased loss on details and small changes. The best performing model is a *ResNet-101* trained on a mix of synthetic and real data. However, there was no noticeable difference to a *HRNet-w48* trained on the same dataset when inspecting videos annotated by them.

Further research should focus on methods how extracted trajectories can be stabilized, even for videos that were filmed without a tripod, and options to postprocess the trajectories to reduce noise and make them more usable for downstream biomechanical analysis. This could include smoothing algorithms, filtering techniques, or model-based motion reconstruction approaches. By improving the stability and precision of the derived kinematic data, these systems could become robust enough for deployment in clinical settings, ultimately contributing to more accurate and accessible objective lameness evaluation for horses.

A

Figures

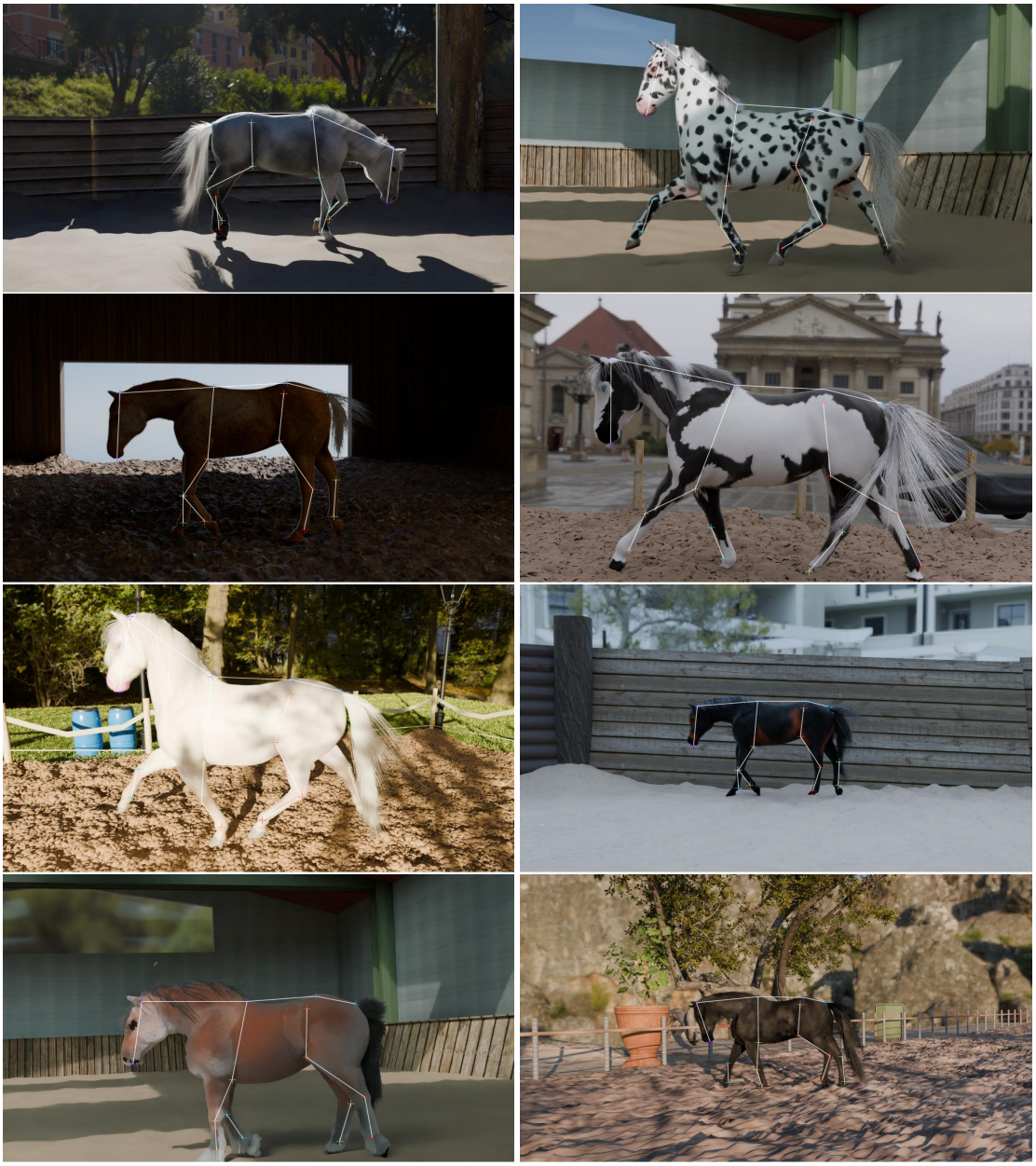
A.1 | Examples from the Synthetic Dataset

A.1.1 | First iterations of the dataset

The first images that were rendered looked a lot less realistic than later iterations. Some examples of those "first tries" can be found here.

A.1.2 | Dataset examples

Here are some examples of images rendered with *PoseCraft* for the dataset. The images show horses in different poses and lighting conditions. The annotated labels and a skeleton where plotted onto them with *DeepLabCut*.

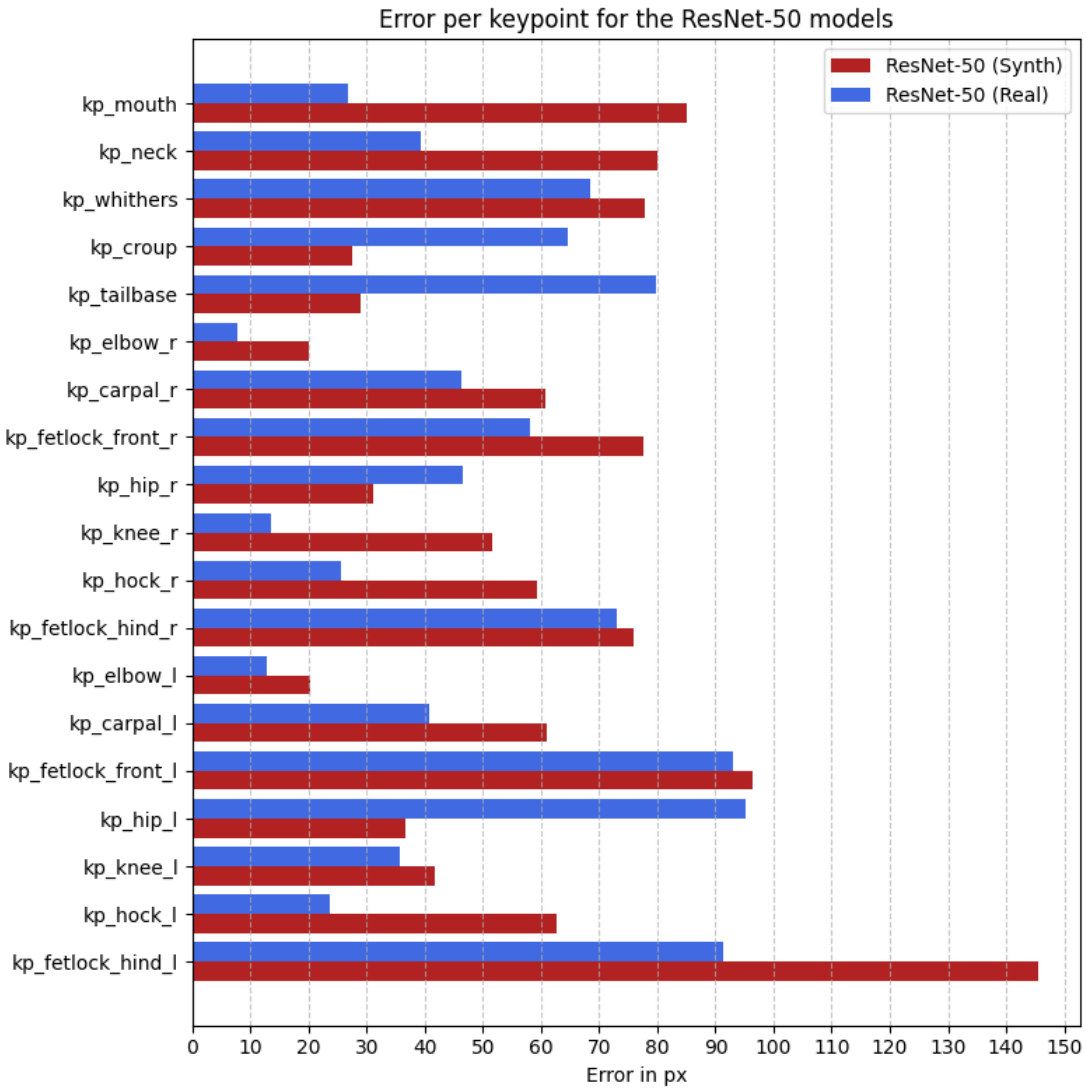


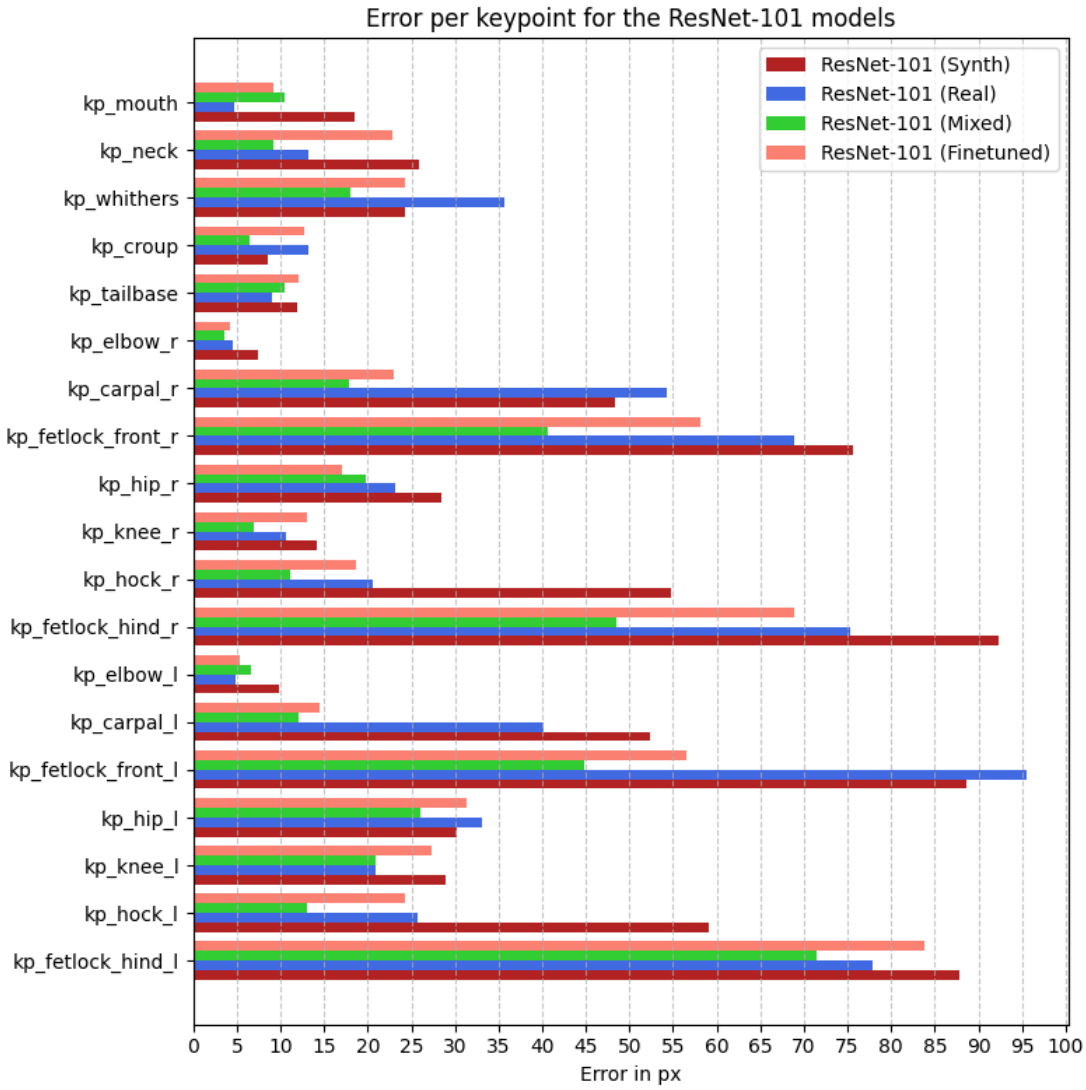
A.2 | Exaples from the Real-World Dataset

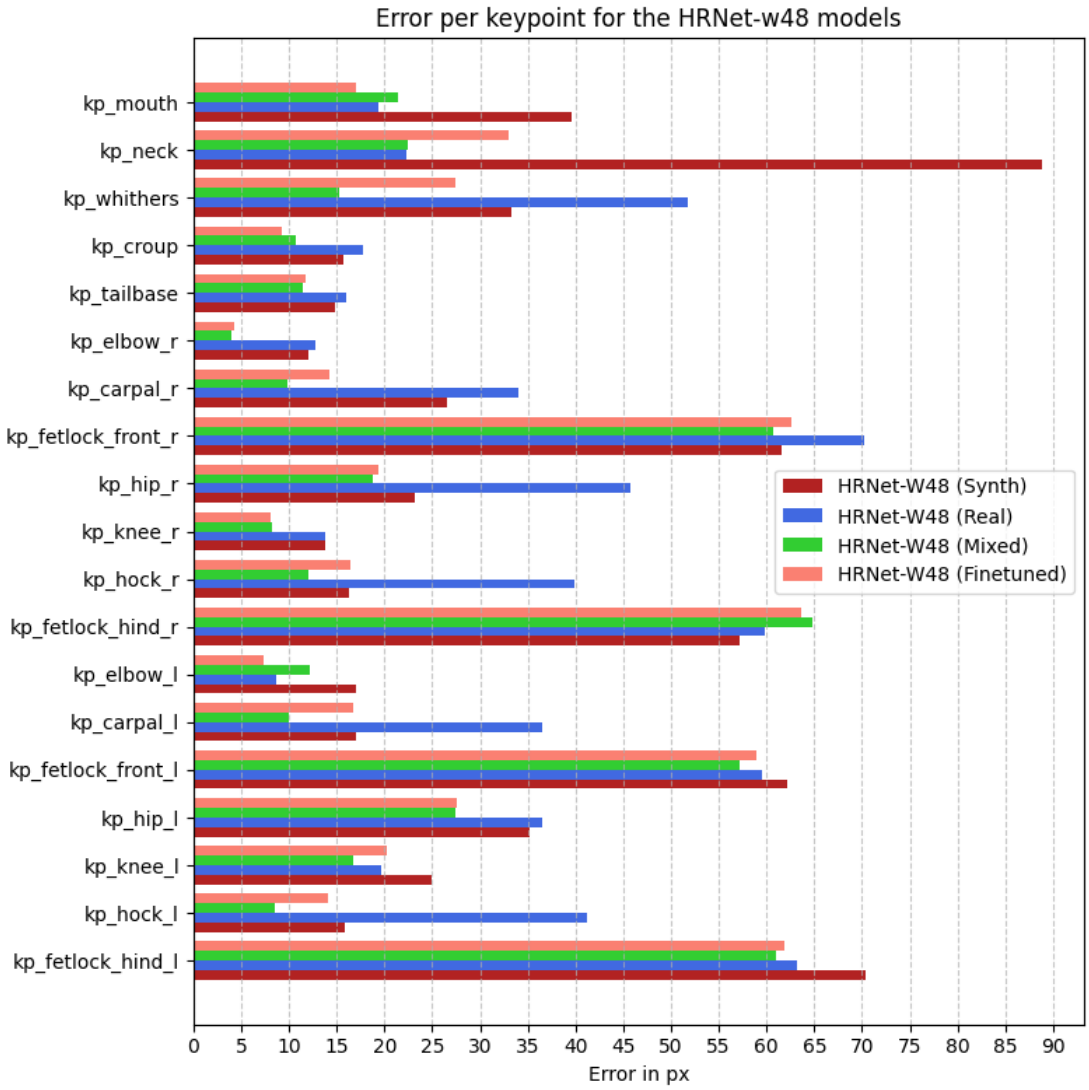
Here are some examples of frames extracted from the real-world videos. The keypoints are annotated with *DeepLabCut* and plotted onto the images.

A.3 | Evaluation results

A.3.1 | Error per Keypoint grouped by Architecture




Error per keypoint for the ResNet-101 models
ResNet-101 (Synth)
ResNet-101 (Real)
ResNet-101 (Mixed)
ResNet-101 (Finetuned)
kp_mouth
kp_neck
kp_whithers
kp_croup
kp_tailbase
kp_elbow_r
kp_carpal_r
kp_fetlock_front_r
kp_hip_r
kp_knee_r
kp_hock_r
kp_fetlock_hind_r
kp_elbow_l
kp_carpal_l
kp_fetlock_front_l
kp_hip_l
kp_knee_l
kp_hock_l
kp_fetlock_hind_l
0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100
Error in px


Error per keypoint for the HRNet-w48 models
kp_mouth
kp_neck
kp_whithers
kp_croup
kp_tailbase
kp_elbow_r
kp_carpal_r
kp_fetlock_front_r
kp_hip_r
kp_knee_r
kp_hock_r
kp_fetlock_hind_r
kp_elbow_l
kp_carpal_l
kp_fetlock_front_l
kp_hip_l
kp_knee_l
kp_hock_l
kp_fetlock_hind_l
HRNet-W48 (Synth)
HRNet-W48 (Real)
HRNet-W48 (Mixed)
HRNet-W48 (Finetuned)
0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90
Error in px

B

Code listings

B.1 | PoseCraft

Using PoseCraft in practice

```
# import the posecraft package
import posecraft
from posecraft.create_render_sequence import create_render_sequence

list_of_combinations = create_render_sequence(n=NUMBER_OF_COMBINATIONS)

# Iterate over the list of combinations of the object of interest, the main file and lighting conditions
for combination in list_of_combinations:

    # dataframe to save the keypoint coordinates into
    keypoint_df = utils.create_keypoint_df(PROJECT_SETTINGS['keypoints'])

    # the Scene Builder object assists with the combinations of the different Blender files into a single scene
    scene_builder = sb.SceneBuilder(scene_name=combination.SCENE_PATH, project_path=PROJECT_PATH, world_file_name=combination.WORLD_PATH, keypoint_object_name=combination.KEYPOINT_OBJECT_PATH, scene_id=combination.ID)

    # load the file under NAME_OF_SCENE containing the environment as mainfile
    scene_builder.load_blend_file()
    # add a World and all Light objects from the file WORLD_FILE_NAME
    scene_builder.load_scene_lighting()
    # create EMPTY-Objects in their own collection, at random locations on the surface of an object named "PlacementArea" in the mainfile
    scene_builder.create_coordinates()
    # load the mesh, armature, textures and curves (hair) from
```

```
    KEYPOINT_OBJECT_PATH
23  scene_builder.load_keypoint_object()
24
25  # The Scene renderer object assists with any changes within the created
    scene and renders and saves the image
26  scene_renderer = sr.SceneRenderer(project_settings=PROJECT_SETTINGS,
    scene_builder=scene_builder)
27
28  # Use the objects created by scene_builder.create_coordinates() to place
    the Armature-Object at random locations within certain constraints
29  for coordinate in scene_builder.coordinate_collection.objects:
30      # move the object to a new location
31      scene_renderer.move_object(coordinate.location)
32      # adjust the orientation of the camera, so it faces the object (an
    offset can be given)
33      scene_renderer.adjust_orientation()
34      # pose the object into a randomly chosen position
35      scene_renderer.pose_object()
36      # save the scene to allow for easier debugging if something goes
    wrong
37      scene_builder.save_scene()
38      # render the image; the function will return the filepath of the
    image
39      filepath = scene_renderer.render_image()
40      # add the keypoints calculated for that pose to the dataframe
41      keypoint_df.loc[filepath] = scene_renderer.get_keypoints(
    PROJECT_SETTINGS['keypoints'])
42  # save the dataframe for this scene
43  keypoint_df.to_csv(f"{PROJECT_PATH}/generated_images/{combination.ID}/
    keypoints.csv")
```

Listing B.1: An example how the PoseCraft package can be used in a script. It is a slightly simplified version where some parameters that are currently necessary are ommitted.

GPU Setting in `bpy`

```
1 bpy.context.preferences.addons["cycles"].preferences.compute_device_type = "
    CUDA"
2   bpy.context.preferences.addons["cycles"].preferences.get_devices()
3   if len(bpy.context.preferences.addons["cycles"].preferences.devices) > 0:
4       for device in bpy.context.preferences.addons["cycles"].preferences.
    devices:
5           if device.type == 'CPU':
6               device.use = False
7           else:
```

```
                device.use = True

    bpy.ops.wm.save_userpref()
```

Listing B.2: Activate GPU in bpy

Copying data objects between Blender files

```
with bpy.data.libraries.load(self.world_file_path) as (data_from, data_to):
    data_to.worlds = data_from.worlds
```

Listing B.3: Import the world from one Blender file to another

```
with bpy.data.libraries.load(self.world_file_path) as (data_from, data_to):
    lights = [name for name in data_from.lights]
    for light in lights:
        bpy.ops.wm.append(
            filepath=light_dir + light,
            directory=light_dir,
            filename=light)
```

Listing B.4: Import lights and other objects

Converting PoseCraft data to DeepLabCut

```
import numpy as np

from posecraft.util import dlc_converter
import pandas as pd

dlc_converter.collect_files(project_path=PATH_TO_POSECRAFT_CONFIG,
                  dlc_path=PATH_TO_DLC_PROJECT_CONFIG,
                  scorer='aylu')

# read csv
path = PATH_TO_COLLECTED_DATA_CSV_DLC

df = pd.read_csv(path, header=[0, 1, 2], index_col=0)

for idx in df.index:
    # find out if the horse is facing left or right
    left_rein = df.loc[idx, ('aylu', 'kp_neck', 'x')] < df.loc[idx, ('aylu',
    'kp_tailbase', 'x')]
    if left_rein:
        df.loc[idx, ('aylu', 'kp_elbow_r')] = np.nan
        df.loc[idx, ('aylu', 'kp_knee_r')] = np.nan
        df.loc[idx, ('aylu', 'kp_hip_r')] = np.nan
```

```
    else:
        df.loc[idx, ('aylu', 'kp_elbow_l')] = np.nan
        df.loc[idx, ('aylu', 'kp_knee_l')] = np.nan
        df.loc[idx, ('aylu', 'kp_hip_l')] = np.nan

df.to_csv(path)
```

Listing B.5: Export PoseCraft data to DeepLabCut

B.2 | Working with DeepLabCut

Dataset preparation

Training of the networks

```
import deeplabcut

deeplabcut.create_training_dataset(
    CONFIG_PATH,
    net_type="resnet_101",
    Shuffles=[2,3,4],
    trainIndices=[synthDataIndices, realDataIndices, realDataIndices+
    synthDataIndices],
    testIndices=[testIndices,testIndices, testIndices],
    userfeedback=False,
)

deeplabcut.train_network(
    CONFIG_PATH,
    shuffle=2,
    epochs=300,
    trainingsetindex=0,
)

deeplabcut.train_network(
    CONFIG_PATH,
    shuffle=3,
    epochs=300,
    trainingsetindex=1,
)

deeplabcut.train_network(
    CONFIG_PATH,
    shuffle=4,
    epochs=300,
```

```
    trainingsetindex=2,
)
```

Listing B.6: Training a model

Analyzing new videos

```
import deeplabcut as dlc

config_path = $PATH_TO_DLC_PROJECT_CONFIG
shuffle_i = $SHUFFLE_ID

dlc.analyze_videos(
    config_path,
    [args.videodir],
    save_as_csv=True,
    shuffle=shuffle_i,
)

dlc.filterpredictions(
    config_path,
    [args.videodir],
    shuffle = shuffle_i
)

dlc.create_labeled_video(
    config_path,
    [args.videodir],
    videotype=".mp4",
    shuffle = shuffle_i,
    save_frames = False,
    filtered=True,
    draw_skeleton=True,
    overwrite=True,
    fastmode=False
)
```

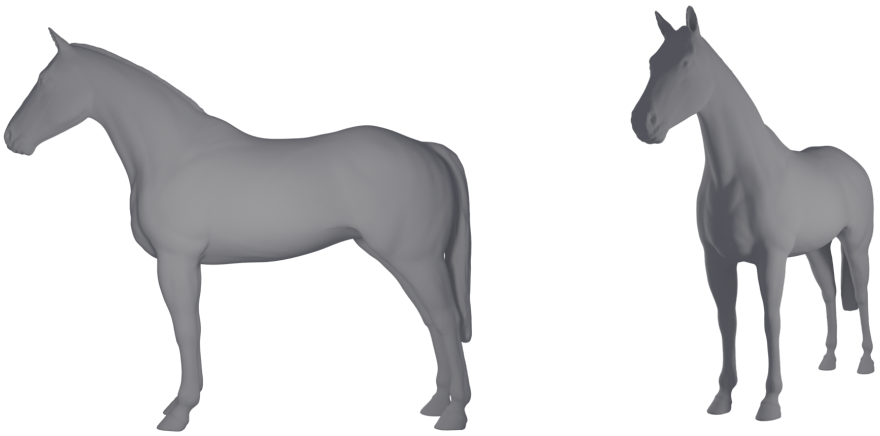
Listing B.7: Analyzing a video with EquiNeT

C

3D assets

C.1 | Horses

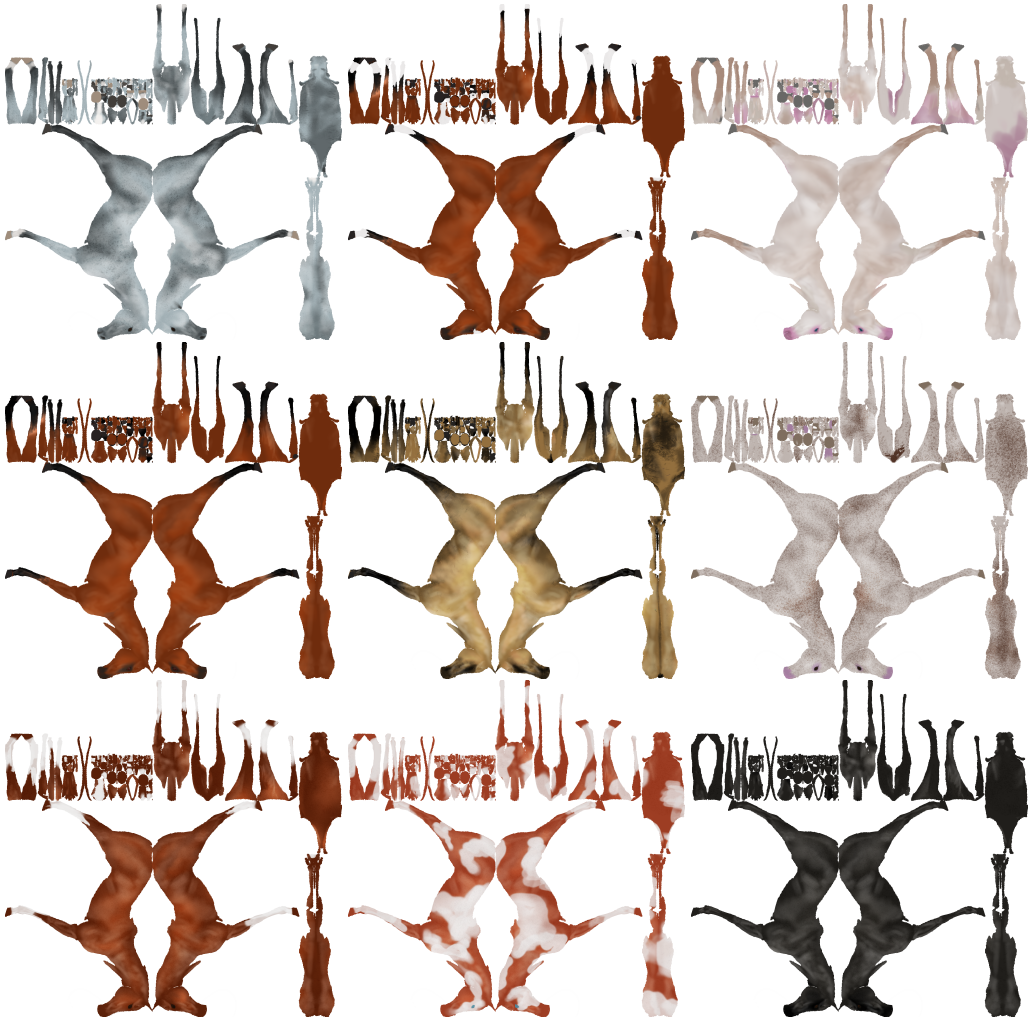
Basemodel



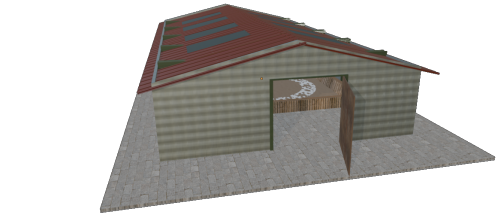
Base Mesh	
Name	American Paint Horse Nuetral V1
Creator	ultramitente
Lizenz	CC-0
URL	https://cults3d.com/en/3d-model/art/caballo-americano-nuetral

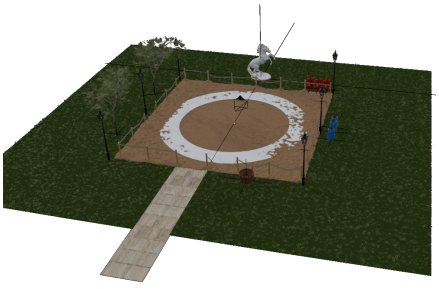
3D Models

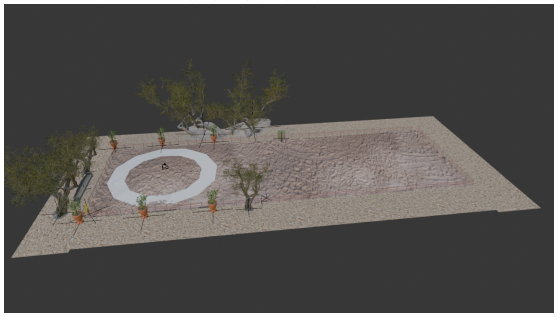
Textures

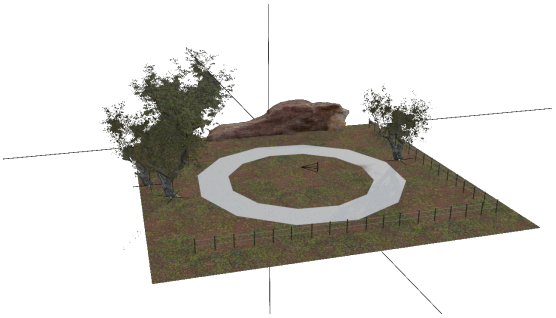


C.2 | 3D scenes









C.3 | Assets from *Poly Haven*

For the creation of the environment, multiple materials (including textures), meshes and 360° HDRIs were imported into Blender using the Poly Haven plug in.

About Poly Haven

Poly Haven is a platform that provides high-quality assets for 3D renders such as meshes, textures/materials and HDRIs. All their assets are available under the *CC0* license, which allows for free use in any project, commercial or non-commercial, without the need for attribution. For suporters they also offer a *Blender* Plugin, that allow for easy acces to their assets by directly being able to drag-and-drop them into the scene. With exception of the Horse model, all assets in this thesis that were not created by the author, were downloaded from *Poly Haven*.

More information on the project can be found on polyhaven.com

Bibliography

[1] P. Baraneedharan, A. Nithyasri, and P. Keerthana. "Revolutionizing Image Recognition and Beyond with Deep Residual Networks". In: *Communication and Intelligent Systems*. Ed. by Harish Sharma et al. Singapore: Springer Nature, 2024, pp. 441–448. ISBN: 978-981-97-2053-8. DOI: 10.1007/978-981-97-2053-8_33.

[2] Gary M. Baxter, Ted S. Stashak, and Kevin G. Keegan. "Examination for Lameness". In: *Adams and Stashak's Lameness in Horses*. 7th ed. John Wiley & Sons, Ltd, 2020, pp. 67–188. ISBN: 978-1-119-27671-5. DOI: 10.1002/9781119276715.ch2. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781119276715.ch2. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119276715.ch2 (visited on 05/15/2025).

[3] Lokesh Borawar and Ravinder Kaur. "ResNet: Solving Vanishing Gradient in Deep Networks". In: *Proceedings of International Conference on Recent Trends in Computing*. Ed. by Rajendra Prasad Mahapatra et al. Singapore: Springer Nature, 2023, pp. 235–247. ISBN: 978-981-19-8825-7. DOI: 10.1007/978-981-19-8825-7_21.

[4] Ravikumar Ch et al. "A Comprehensive Analysis for Advancements and Challenges in Deep Learning Models for Image Processing". In: *Proceedings of the 5th International Conference on Data Science, Machine Learning and Applications; Volume 1*. Ed. by Amit Kumar et al. Singapore: Springer Nature, 2025, pp. 229–234. ISBN: 978-981-97-8031-0. DOI: 10.1007/978-981-97-8031-0_24.

[5] Hilary M. Clayton. "Cinematographic Analysis of the Gait of Lame Horses". In: *Journal of Equine Veterinary Science* 6.2 (Jan. 1, 1986), pp. 70–78. ISSN: 0737-0806. DOI: 10.1016/S0737-0806(86)80037-5. URL: https://www.sciencedirect.com/science/article/pii/S0737080686800375 (visited on 05/23/2025).

[6] Cristian Mihaita Crecan et al. "Development of a Novel Approach for Detection of Equine Lameness Based on Inertial Sensors: A Preliminary Study". In: *Sensors* 22.18 (Jan. 2022), p. 7082. ISSN: 1424-8220. DOI: 10.3390/s22187082. URL: https://www.mdpi.com/1424-8220/22/18/7082 (visited on 08/13/2025).

[7] Elizabeth J. Davidson. "Lameness Evaluation of the Athletic Horse". In: *The Veterinary Clinics of North America. Equine Practice* 34.2 (Aug. 2018), pp. 181–191. ISSN: 1558-4224. DOI: 10.1016/j.cveq.2018.04.013. PMID: 30007446.

[8] Jia Deng et al. "ImageNet: A Large-Scale Hierarchical Image Database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009 IEEE Conference on Computer Vision and Pattern Recognition. June 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848. URL: https://ieeexplore.ieee.org/document/5206848 (visited on 08/11/2025).

[9] J. R. Donnell et al. "Comparison of Subjective Lameness Evaluation, Force Platforms and an Inertial-Sensor System to Identify Mild Lameness in an Equine Osteoarthritis Model". In: *The Veterinary Journal* 206.2 (Nov. 1, 2015), pp. 136–142. ISSN: 1090-0233. DOI: 10.1016/j.tvjl.2015.08.004. URL: https://www.sciencedirect.com/science/article/pii/S1090023315003287 (visited on 10/06/2023).

[10] Jörg Drechsler and Anna-Carolina Haensch. "30 Years of Synthetic Data". In: *Statistical Science* 39.2 (May 1, 2024). ISSN: 0883-4237. DOI: 10.1214/24-STS927. URL: https://projecteuclid.org/journals/statistical-science/volume-39/issue-2/30-Years-of-Synthetic-Data/10.1214/24-STS927.full (visited on 05/02/2025).

[11] Samantha Elmhurst. *The Horse in Motion: The Anatomy and Physiology of Equine Locomotion*. United States: Wiley, 2013. ISBN: 978-1-118-70236-9.

[12] Khaled Emam. *Accelerating AI with Synthetic Data*. 1st ed. O'Reilly Media, Inc, 2020. ISBN: 978-1-4920-4599-1. URL: https://learning.oreilly.com/library/view/~/9781492045991/?ar (visited on 04/22/2025).

[13] EU. *Verordnung (EU) 2016/679 des Europäischen Parlaments und des Rates vom 27. April 2016 zum Schutz natürlicher Personen bei der Verarbeitung personenbezogener*

Daten, zum freien Datenverkehr und zur Aufhebung der Richtlinie 95/46/EG. May 25, 2018.

[14] Abassin Sourou Fangbemi et al. *ZooBuilder: 2D and 3D Pose Estimation for Quadrupeds Using Synthetic Data*. Sept. 1, 2020. DOI: 10.48550/arXiv.2009.05389. arXiv: 2009.05389 [cs]. URL: http://arxiv.org/abs/2009.05389 (visited on 05/24/2024). Pre-published.

[15] Ann-Kristin Feuser et al. "Artificial Intelligence for Lameness Detection in Horses—A Preliminary Study". In: *Animals* 12.20 (Oct. 17, 2022), p. 2804. ISSN: 2076-2615. DOI: 10.3390/ani12202804. URL: https://www.mdpi.com/2076-2615/12/20/2804 (visited on 10/06/2023).

[16] Free Software Foundation, Inc. *What Is Free Software? - GNU Project - Free Software Foundation*. gnu.org. 2025. URL: https://www.gnu.org/philosophy/free-sw.html.en (visited on 08/13/2025).

[17] M. Hammarberg et al. "Rater Agreement of Visual Lameness Assessment in Horses during Lungeing". In: *Equine Veterinary Journal* 48.1 (Jan. 2016), pp. 78–82. ISSN: 2042-3306. DOI: 10.1111/evj.12385. PMID: 25399722.

[18] A. M. Hardeman et al. "Variation in Gait Parameters Used for Objective Lameness Assessment in Sound Horses at the Trot on the Straight Line and the Lunge". In: *Equine Veterinary Journal* 51.6 (Nov. 2019), pp. 831–839. ISSN: 0425-1644, 2042-3306. DOI: 10.1111/evj.13075. URL: https://beva.onlinelibrary.wiley.com/doi/10.1111/evj.13075 (visited on 05/31/2024).

[19] Aagje M. Hardeman et al. "Visual Lameness Assessment in Comparison to Quantitative Gait Analysis Data in Horses". In: *Equine Veterinary Journal* 54.6 (2022), pp. 1076–1085. ISSN: 2042-3306. DOI: 10.1111/evj.13545. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/evj.13545 (visited on 05/31/2024).

[20] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas, NV, USA: IEEE, June 2016, pp. 770–778. ISBN: 978-1-4673-8851-1. DOI: 10.1109/CVPR.

2016.90. URL: http://ieeexplore.ieee.org/document/7780459/ (visited on 08/11/2025).

[21] Sarah Hobbs et al. "Motion Analysis and Its Use in Equine Practice and Research". In: *Wiener tierärztliche Monatsschrift* 97 (Jan. 1, 2010), pp. 55–64.

[22] Sarah Jeune and James Jones. "Prospective Study on the Correlation of Positive Acupuncture Scans and Lameness in 102 Performance Horses". In: *American Journal of Traditional Chinese Veterinary Medicine* (Aug. 1, 2014). DOI: 10.59565/001c.83237.

[23] Le Jiang et al. "Animal Pose Estimation: A Closer Look at the State-of-the-Art, Existing Gaps and Opportunities". In: *Computer Vision and Image Understanding* 222 (Sept. 1, 2022), p. 103483. ISSN: 1077-3142. DOI: 10.1016/j.cviu.2022.103483. URL: https://www.sciencedirect.com/science/article/pii/S1077314222000893 (visited on 05/24/2024).

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Communications of the ACM* 60.6 (May 24, 2017), pp. 84–90. ISSN: 0001-0782, 1557-7317. DOI: 10.1145/3065386. URL: https://dl.acm.org/doi/10.1145/3065386 (visited on 08/11/2025).

[25] Akshay Kulkarni, Adarsha Shivananda, and Nitin Ranjan Sharma. "Pose Estimation". In: *Computer Vision Projects with PyTorch: Design and Develop Production-Grade Models*. Ed. by Akshay Kulkarni, Adarsha Shivananda, and Nitin Ranjan Sharma. Berkeley, CA: Apress, 2022, pp. 199–227. ISBN: 978-1-4842-8273-1. DOI: 10.1007/978-1-4842-8273-1_6. URL: https://doi.org/10.1007/978-1-4842-8273-1_6 (visited on 08/12/2025).

[26] Felix Järemo Lawin et al. "Is Markerless More or Less? Comparing a Smartphone Computer Vision Method for Equine Lameness Assessment to Multi-Camera Motion Capture". In: *Animals* 13.3 (Jan. 2023), p. 390. ISSN: 2076-2615. DOI: 10.3390/ani13030390. URL: https://www.mdpi.com/2076-2615/13/3/390 (visited on 10/06/2023).

[27] Hang Li. "Introduction to Machine Learning and Supervised Learning". In: *Machine Learning Methods*. Ed. by Hang Li. Singapore: Springer Nature, 2024, pp. 1–37. ISBN: 978-981-99-3917-6. DOI: 10.1007/978-981-99-3917-6_1. URL:

https://doi.org/10.1007/978-981-99-3917-6_1 (visited on 08/13/2025).

[28] Tsung-Yi Lin et al. "Microsoft COCO: Common Objects in Context". In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Cham: Springer International Publishing, 2014, pp. 740–755. ISBN: 978-3-319-10602-1. DOI: 10.1007/978-3-319-10602-1_48.

[29] George F. Luger. "An Introduction to Neural Networks". In: *Artificial Intelligence: Principles and Practice*. Ed. by George F. Luger. Cham: Springer Nature Switzerland, 2025, pp. 345–361. ISBN: 978-3-031-57437-5. DOI: 10.1007/978-3-031-57437-5_15. URL: https://doi.org/10.1007/978-3-031-57437-5_15 (visited on 08/12/2025).

[30] George F. Luger. "Deep Learning: Introduction and Representations". In: *Artificial Intelligence: Principles and Practice*. Ed. by George F. Luger. Cham: Springer Nature Switzerland, 2025, pp. 383–408. ISBN: 978-3-031-57437-5. DOI: 10.1007/978-3-031-57437-5_17. URL: https://doi.org/10.1007/978-3-031-57437-5_17 (visited on 08/12/2025).

[31] George F. Luger. "The Delta Rule, Backpropagation, and Matrix Representations". In: *Artificial Intelligence: Principles and Practice*. Ed. by George F. Luger. Cham: Springer Nature Switzerland, 2025, pp. 363–382. ISBN: 978-3-031-57437-5. DOI: 10.1007/978-3-031-57437-5_16. URL: https://doi.org/10.1007/978-3-031-57437-5_16 (visited on 08/12/2025).

[32] Madhumati C. Mailarada et al. "Pose Estimation: Human Keypoint Prediction". In: *Emerging Trends and Technologies on Intelligent Systems*. Ed. by Arti Noor et al. Singapore: Springer Nature, 2025, pp. 535–548. ISBN: 978-981-97-5703-9. DOI: 10.1007/978-981-97-5703-9_44.

[33] Abdul Majeed and Seong Oun Hwang. "Synthetic Data: A New Frontier for Democratizing Artificial Intelligence and Data Access". In: *Computer* 58.2 (Feb. 2025), pp. 106–114. ISSN: 1558-0814. DOI: 10.1109/MC.2024.3515412. URL: https://ieeexplore.ieee.org/document/10857849 (visited on 04/22/2025).

[34] Alexander Mathis et al. "DeepLabCut: Markerless Pose Estimation of User-Defined Body Parts with Deep Learning". In: *Nature Neuroscience* 21.9 (9 Sept. 2018), pp. 1281–

1289. ISSN: 1546-1726. DOI: 10.1038/s41593-018-0209-y. URL: https://www.nature.com/articles/s41593-018-0209-y (visited on 10/24/2023).

[35] Paweł Nadachowski et al. "Classification of Glacial and Fluvioglacial Landforms by Convolutional Neural Networks Using a Digital Elevation Model". In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* PP (Jan. 1, 2024), pp. 1–17. DOI: 10.1109/JSTARS.2024.3470253.

[36] Sergey I. Nikolenko. *Synthetic Data for Deep Learning*. Vol. 174. Springer Optimization and Its Applications. Cham: Springer International Publishing, 2021. ISBN: 978-3-030-75177-7 978-3-030-75178-4. DOI: 10.1007/978-3-030-75178-4. URL: https://link.springer.com/10.1007/978-3-030-75178-4 (visited on 04/08/2025).

[37] Sinno Jialin Pan and Qiang Yang. "A Survey on Transfer Learning". In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (Oct. 2010), pp. 1345–1359. ISSN: 1041-4347. DOI: 10.1109/TKDE.2009.191. URL: http://ieeexplore.ieee.org/document/5288526/ (visited on 08/11/2025).

[38] T. Pfau et al. "Lungeing on Hard and Soft Surfaces: Movement Symmetry of Trotting Horses Considered Sound by Their Owners". In: *Equine Veterinary Journal* 48.1 (2016), pp. 83–89. ISSN: 2042-3306. DOI: 10.1111/evj.12374. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/evj.12374 (visited on 10/24/2023).

[39] Thilo Pfau et al. "Effect of Trotting Speed and Circle Radius on Movement Symmetry in Horses during Lunging on a Soft Surface". In: *American Journal of Veterinary Research* 73.12 (Dec. 1, 2012), pp. 1890–1899. DOI: 10.2460/ajvr.73.12.1890. URL: https://avmajournals.avma.org/view/journals/ajvr/73/12/ajvr.73.12.1890.xml (visited on 10/24/2023).

[40] M. Rhodin et al. "Effect of Lungeing on Head and Pelvic Movement Asymmetry in Horses with Induced Lameness". In: *The Veterinary Journal*. E-Supplement: 7th International Conference on Canine and Equine Locomotion 198 (Dec. 1, 2013), e39–e45. ISSN: 1090-0233. DOI: 10.1016/j.tvjl.2013.09.031. URL: https://www.sciencedirect.com/science/article/pii/S1090023313004590 (visited on 10/24/2023).

[41] M. Rhodin et al. "Head and Pelvic Movement Asymmetry during Lungeing in Horses with Symmetrical Movement on the Straight". In: *Equine Veterinary Journal* 48.3 (May 2016), pp. 315–320. ISSN: 0425-1644, 2042-3306. DOI: 10.1111/evj.12446. URL: https://beva.onlinelibrary.wiley.com/doi/10.1111/evj.12446 (visited on 10/24/2023).

[42] Donald Bruce Rubin. "Discussion: Statistical Disclosure Limitation". In: *Journal of Official Statistics* 9.2 (1993), pp. 462–468. URL: https://www.scb.se/contentassets/ca21efb41fee47d293bbee5bf7be7fb3/discussion-statistical-disclosure-limitation2.pdf?utm_source=techmap.io&utm_medium=search-engine&cid=techmap.io (visited on 05/02/2025).

[43] Donald Bruce Rubin. "Multiple Imputations in Sample Surveys-a Phenomenological Bayesian Approach to Nonresponse". In: *Proceedings of the Survey Research Methods Section of the American Statistical Association*. Vol. 1. VA, USA: American Statistical Association Alexandria, 1978, pp. 20–34. URL: http://www.asasrms.org/Proceedings/papers/1978_004.pdf (visited on 05/02/2025).

[44] F. M. Serra Bragança, M. Rhodin, and P. R. van Weeren. "On the Brink of Daily Clinical Application of Objective Gait Analysis: What Evidence Do We Have so Far from Studies Using an Induced Lameness Model?" In: *The Veterinary Journal* 234 (Apr. 1, 2018), pp. 11–23. ISSN: 1090-0233. DOI: 10.1016/j.tvjl.2018.01.006. URL: https://www.sciencedirect.com/science/article/pii/S1090023318300066 (visited on 05/19/2025).

[45] Moira Shooter, Charles Malleson, and Adrian Hilton. *SyDog: A Synthetic Dog Dataset for Improved 2D Pose Estimation*. July 31, 2021. DOI: 10.48550/arXiv.2108.00249. arXiv: 2108.00249 [cs]. URL: http://arxiv.org/abs/2108.00249 (visited on 08/13/2025). Pre-published.

[46] Sandra D. Starke and Stephen A. May. "Veterinary Student Competence in Equine Lameness Recognition and Assessment: A Mixed Methods Study". In: *Veterinary Record* 181.7 (2017), pp. 168–168. ISSN: 2042-7670. DOI: 10.1136/vr.104245. URL: https://onlinelibrary.wiley.com/doi/abs/10.1136/vr.104245 (visited on 10/06/2023).

[47] Sandra D. Starke and Maarten Oosterlinck. "Reliability of Equine Visual Lameness Classification as a Function of Expertise, Lameness Severity and Rater Con-

fidence". In: *Veterinary Record* 184.2 (2019), pp. 63–63. ISSN: 2042-7670. DOI: 10.1136/vr.105058. URL: https://onlinelibrary.wiley.com/doi/abs/10.1136/vr.105058 (visited on 06/05/2024).

[48] Yiqi Wang et al. "Identifying Lameness in Horses through Deep Learning". In: SAC '21: The 36th ACM/SIGAPP Symposium on Applied Computing. Virtual Event Republic of Korea: ACM, Mar. 22, 2021, pp. 976–985. ISBN: 978-1-4503-8104-8. DOI: 10.1145/3412841.3441973. URL: https://dl.acm.org/doi/10.1145/3412841.3441973 (visited on 10/06/2023).

[49] R. Weller et al. "Reliability of Conformational Measurements in the Horse Using a Three-Dimensional Motion Analysis System". In: *Equine Veterinary Journal* 38.7 (Nov. 2006), pp. 610–615. ISSN: 0425-1644. DOI: 10.2746/042516406x150367. PMID: 17228574.

[50] Barret Zoph et al. "Rethinking Pre-training and Self-training". In: ().

Software

[51] *Blender*. Version 4.2. Blender Foundation, Apr. 16, 2024. URL: https://www.blender.org/ (visited on 05/31/2024).

[52] Blender Authors. *Blender Python API*. URL: https://docs.blender.org/api/current/index.html (visited on 07/27/2024).

[53] Blender Foundation. *Rigify — Blender Manual*. Jan. 15, 2024. URL: https://docs.blender.org/manual/de/4.0/addons/rigging/rigify/introduction.html (visited on 07/16/2025).

[54] Maximilian Denninger et al. "BlenderProc2: A Procedural Pipeline for Photorealistic Rendering". In: *Journal of Open Source Software* 8.82 (2023), p. 4901. DOI: 10.21105/joss.04901. URL: https://doi.org/10.21105/joss.04901.

[55] Epic Games, Inc. *Das leistungsstärkste Werkzeug für 3D-Echtzeit-Entwicklung*. Unreal Engine. URL: https://www.unrealengine.com/de/home (visited on 08/13/2025).

[56] Sleip AI AB. *Sleip – Scientifically Validated Equine Gait Analysis App*. 2025. URL: https://sleip.com (visited on 08/12/2025).

[57] Unity Technologies. *Echtzeit-Entwicklungsplattform von Unity | 3D, 2D, VR- und AR-Engine*. Unity. URL: https://unity.com (visited on 08/13/2025).