Hochschule Darmstadt

Department of Natural Sciences and Mathematics

&

Information Science and Computer Science

Individual identification of patterned solitary species based on unlabeled video data

Thesis for the award of the academic degree

Master of Science (M. Sc.)

in the study program Data Science

submitted by:

Vanessa Süßle

First supervisor:

Prof. Dr. Elke Hergenröther, Department of Computer Sciences

Second supervisor:

Prof. Dr. Jutta Groos, Department of Mathematics

Issue date: 01/07/2021

Submission date: 16/12/2021

Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen

entnommen sind, sind als solche kenntlich gemacht. Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden

Quellennachweis versehen. Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Mörfelden, den 16.12.2021 _____

ABSTRACT

The manual processing and analysis of videos from camera traps is a time consuming process and includes several steps from the filtering of false triggered footage to the identification and reidentification of individuals.

The program developed in the course of this thesis provides a tool to automatically process videos from camera traps without the need of manual interaction. The recognition of individuals in the animal kingdom differs between species and is not possible across all species based on visual characteristics. The work described here addresses the identification of animals with unique fur patterns. The system can only be applied to animal species that meet specific requirements. The species of interest must be uniquely identifiable based on visual characteristics, similar to the human fingerprint. The inspected species should have a mainly solitary behavior, which is required for the basic assumption. It is assumed that only the same individual can be seen throughout one triggered video sequence.

The program includes components of analytical computer vision as well as Deep Learning methods like Convolution Neural Networks. The assignment of individuals is based on the SIFT algorithm. In addition, other components were implemented to substitute the otherwise necessary interaction of a human. Based on similarity between frames from the video material, calculated by the Feature Detector, clusters are formed. A cluster represents an individual and groups the available data to this individual. It is not required to know the single individuals prior to the analysis.

The program was tested on the leopard dataset from the PanAfrican Programme and achieved a rate of over 85% for correct assignments between previously unknown individuals.

ABSTRACT GERMAN

Die manuelle Auswertung von Videos aus Kamerafallen vom Herausfiltern von Fehlauslösern bis hin zur Erkennung von Individuen ist ein zeitintensiver Prozess.

Das hier entwickelte Programm stellt ein Werkzeug dar, mit welchem Videomaterial aus Kamerafallen automatisch ausgewertet werden kann, ohne die Notwendigkeit manueller Interaktion durch den Anwender. Die Wiedererkennung von Individuen im Tierreich unterscheidet sich zwischen den Tierarten und ist nicht für jede Spezies basierend auf visuellen Merkmalen möglich. Die hier beschriebene Arbeit befasst sich mit der Identifizierung von Tieren mit eindeutigen Fellzeichnungen. Das System kann nur für Tierarten angewendet werden, die spezifische Voraussetzungen erfüllen. Die zu identifizierende Tierart muss anhand von visuellen Merkmalen eindeutig erkennbar sein, ähnlich dem menschlichen Fingerabdruck. Des Weiteren sollte die betrachtete Spezies hauptsächlich ein Einzelgänger Verhalten aufweisen. Daraus ergibt sich für die Bildauswertung die grundlegende Annahme, dass nur ein Individuum in einer Videosequenz zu sehen ist.

Das Programm beinhaltet Komponenten der analytischen Computer Vision als auch Deep Learning Verfahren wie Convolution Neural Networks. Die Zuordnung von Individuen basiert auf dem SIFT Algorithmus. Zusätzlich wurden weitere Komponenten, implementiert, um die sonst notwendige Interaktion eines Menschen zu ersetzen. Anhand der Matching Scores zwischen Frames aus dem Videomaterial, berechnet durch den Feature Detector, werden Cluster gebildet. Ein Cluster repräsentiert ein Individuum und gruppiert die vorhandenen Daten zu diesem Individuum. Vor Beginn muss keines der Individuen bereits bekannt sein.

Das Program wurde auf dem Leoparden Datensatz des PanAfrican Programmes getestet und erreichte eine Quote von über 85% für korrekte Zuordnungen zwischen zuvor unbekannten Individuen.

ACKNOWLEDGEMENTS

Firstly, I would like to thank professor Elke Hergenröther for sparking my interest in computer vision.

Next, I especially thank Mimi Arandjelovic from the PanAfrican Programme for her support. In an inspiring dialog we discussed from which automated processes ecologists would profit the most and where it overlaps with computer vision. In this way I could design the definition of my research issue. The PanAfrican Programm provided me with their leopard dataset, without which I could not have accomplished the work for this thesis. Furthermore, I want to thank Anja Landsmann for the support with her expertise in leopards.

I am grateful to my parents who have always encouraged me during my studies and for labeling almost 3000 cheetah images for this thesis.

In general I want to thank the growing community that brings together people for the conversation of nature from the biological and computer science point of view, especially in machine learning. I met very kind and open-minded people that took their time and introduced me to scientists and specialists working in the relevant area. I felt very welcomed from the first moment.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

"We only know a tiny proportion about the complexity of the natural world. Wherever you look, there are still things we don't know about and don't understand. [...] There are always new things to find out if you go looking for them."
— Sir David Attenborough

1 INTRODUCTION

Efficient and reliable monitoring of wild animals in their natural habitats is essential for wildlife conservatory, but a complex and time intense task for ecologists. It is a crucial step to answer hypotheses on incidences, behavior, habitat ranges, social relationships, potential conflicts with humans and livestock as well as their general way of life. Performing population census on a certain species gives scientists insights on the specie's endangerment, their impact on the ecosystem they are part of and helps to achieve conservation objectives to adequately protect the population.

Individual identification is an often used method to estimate a population. Over the past years camera traps became an increasingly popular tool for wildlife monitoring. Camera traps are equipped with motion and/or infrared sensors and are therefore suitable to unobtrusively monitor wildlife. The low acquisition costs and maintenance requirements make camera traps a cost-effective way to collect large volumes of data without invading the habitat and disrupting animal's natural behaviors. The bottleneck in this process is the researcher's time consuming and monotonous task to analyze the enormous volume of data. First to filter the empty false-triggered videos and images and later to establish connections between the appearing individuals.

Leopards (*Panthera pardus*) are disappearing at an alarming rate and were categorized as vulnerable on the IUCN Red List. Leopards represent an important conservation icon due to their prominent fur pattern and general popularity for people and children. They are distributed from the tropical forest area to semi-deserts. Often confused with jaguars (*Panthera onca*) due to their similar pelage, leopards are only indigenous to Africa and Asia and do not have a common living space with jaguars from South America. As a large carnivore they have a crucial impact on ecosystems, wherefore it is of big concern to ecologists to estimate their population in an ecosystem. Like many other large carnivors, especially big cats (*Felidae*), leopards are difficult to monitor and rarely seen by humans. Camera traps offer a non-invasive solution to gather video data on leopards. Due to the fact that leopards have large territories and a low population density very few of the motion-triggered videos are likely to contain footage on leopards besides many other species.

The target of this thesis is to address the problem of the time consuming identification task of individual animals by developing a program that automatically processes the data taken from camera traps with computer vision methods, eliminating the manual tagging conducted by ecologists. The developed system visually identifies individuals by means

of their distinctive body marks or fur patterns using tools from the analytical computer vision and is therefore only applicable for species with those characteristics. Throughout this thesis the system will be demonstrated on a dataset of leopards collected by the PanAfrican Programme.

Making use of the fact that a leopard's coat pattern has the same characteristic as a human's fingerprint to uniquely identify an individual, it is desired to label each individual seen in the camera trap data with a unique name ID, if possible. This task can be challenging, because the data was collected in the wild and may differ in lighting and quality, especially for partly nocturnal animals like leopards. Major obstacles for the identification are pose variations of the animals and the varying viewpoints. Unlike zebras for instance, leopards are a solitary species, which offers the opportunity for the core assumption for this thesis. It is assumed that within one motion-triggered video the exact same individual is seen throughout the frame sequences.

The analysis of the data includes several steps from extracting the frames from the video to the final identification of the individual by a feature detection and matching algorithm. For a more robust system interim steps built on deep learning methods are implemented to detect the animal within each frame and to check for the viewpoint are implemented to overcome the versatility of the data caused by the animal's natural habitat and behavior. The frames of one video are automatically tagged with the same name ID and are queried against the database containing frames of other videos. If a match is detected the name IDs of the videos are merged for all frames in the database. Finally, all images that contain the same individual are clustered in the database.

2 DESIGN AND DEVELOPMENT OF THE RESEARCH ISSUE

This chapter highlights my approach previous to the actual data science task. I decided to approach wildlife conservation organizations for a collaboration. My wish is to help bring wildlife conversation forward by taking part in the development of tools that support an ecologist's work to protect ecosystems all over the planet.

> *"To restore stability to our planet, therefore, we must restore its biodiversity, the very thing we have removed. It is the only way out of this crisis that we ourselves have created. We must rewild the world!"*
> **— Sir David Attenborough**

2.1 The PanAfrican Programme

The PanAfrican Programme project was initiated by the Max Planck Institute for Evolutionary Anthropology in Leipzig with many local partners in Africa. The full name of the project is "The PanAfrican Programme : The Cultured Chimpanzee". The main target of the project is to collect data on chimpanzees in order to get insights into their behavior. Therefore, the setup for the data collection was specifically designed to collect footage on chimpanzees. Data taken of other species is used for other projects as well. All data is captured by automatically triggered camera traps. The PanAfrican Programme hosts research sites in central and west Africa.

The main idea of this thesis is to support ecologists in their work to conserve wildlife. In a dialog with one of the program's scientists, Dr. Mimi Arandjelovic, we modeled my research issue. First ideas included to develop a system for automated identification of individuals for chimpanzees. But chimpanzees have a monochromatic fur and cannot be identified by a coat pattern. The individual identification for monkeys in general is mostly done on their faces or sometimes buttocks. Deep Learning methods for identification require a large amount of labeled data, which seemed hard to get for chimpanzee's faces. Further brainstorming lead to the idea that, if not enough labeled data for deep learning approaches is available classical computer vision methods might be the solution. For animals with a patterned coat those methods can be applied. The dataset of the PanAfrican Programme includes a subset of footage triggered by leopards, which fulfill the requirement of individual coat patterns.

2.2 Leopards

The identification of individuals in the animal kingdom can only be automated for species with distinctive visual properties. Those distinguishing features can be body marks, fur patterns or the shape of recognizable body parts. For species without fur like rays, sharks and whales, body marks like the spots on their skin or the contour of flukes and fins can be distinctive features to reidentify an animal. Many species of the *Felidae*, from both *Pantherinae* (big cats) and *Felinae* (small cats), wear patterned coats. The colloquial terms for *Felinae* and *Pantherinae* are misleading, because the distinction

between them is not defined by their size. *Pantherinaes* with a body pattern are jaguars (*Panthera onca*), leopards (*Panthera pardus*), tigers (*Panthera tigris*), snow leopards (*Uncia uncia*), clouded leopards (*Neofelis nebulosa*), whereas the patterned cheetahs (*Acinonyx jubatus*), lynxes (Lynx lynx), ocelots (*Leopardus pardalis*) and servals (*Leptailurus serval*) belong to the fur-patterned *Felinae*. Individual identification based on fur patterns does not work for all species of the family. Lions (*Panthera leo*), caracals (*Caracal caracal*) and cougars (*Puma concolor*) among others do not have a coated fur pattern with characteristics for an individual identification. For the species mentioned in the beginning the pattern for each individual is unique and therefore the individuals can be uniquely identified and reidentified [1].

Most species of the *Felidae* family are solitary and territorial animals for the most time, exceptions are the mating season and rearing of the cubs.

Leopards (*Panthera pardus*) belong to the family of *Felidae* and the sub-family *Pantherinae*, have a patterned coat and are solitary. Leopards are classified as "Vulnerable" by the IUCN Red List of Threatened Species [2].

Leopards can be divided in eight subspecies based on their fur coloring [3], [4]. They inhabit mostly the savanna and rainforests, but are also seen in the sub-Saharan area as well as Asian coniferous forests [5]. Their territories range from 5-30 km² for females and 16-96 km² for males according to a study conducted by and in the Kruger National Park. The territories of females in adjoining territories can overlap as well as the territory of a male with multiple female individuals [6]. Therefore, different leopards can be caught by a fixed camera trap location. The rearing of the cubs can take up to 13-18 months. For the first months the mother hides the cubs in a den and hunts alone [7], [8]. Only during rearing phase, mating or in territorial fights more than one leopard could be caught by a camera trap in a scene. By the time the cubs become independent male leopards usually leave and aim for territories far away while females may stay close and could be caught in camera traps nearby.

The method described in this thesis is only applicable for solitary animals with patterned fur coats, demonstrated on a dataset containing African leopards (*Panthera pardus pardus*). The spots on a leopard's fur are called rosettes and are mostly concentrated on the back, the flanks, the hind limbs and the upper tail. The pattern on an individual's coating is very distinctive so that a match of an individual in different images can sometimes be verified by as few as a single rosette or a combination of three rosettes. The underbelly, paws and throat are free of rosettes, but spotted. On the basis of those parts it is harder to identify an individual, but it must also be mentioned that those parts are less likely caught on camera based on the natural movement of a leopard. For the method developed in this work it is important to mention that the left and right side of a leopard do not have the same pattern. The patterns on the flanks are totally independent and do not display mirrored images or other relations [9].

2.3 Core assumptions and scheme

The best basis for the research on leopards in an area would be to have images of the individuals from diverse perspectives, knowing that it is the same individual. Making use of the video data instead of only images, it could be possible to extract that information from different frames, if the collected data allows. For the planned concept the following assumptions must be made:

1. The inspected animal species has a solitary behavior.
2. Within one triggered video the same individual is seen throughout the frame sequences.
3. The inspected species is uniquely identifiable by coat or body marks.

Assumption two arises from assumption one.

For animals with herd-behavior like zebras this will mostly not be given. Within one video several animals might be seen in every frame changing their position to each other or occluding individuals, making it nearly impossible to automatically collect detections of one individual from different viewpoints. The fact that most solitary species do not have a solitary behavior during the rearing phase and obviously not during mating is neglected for the course of this thesis. It is assumed that the majority of the videos show single individuals.

The third mandatory prerequisite is a technical necessity for the use of computer vision methods. The species to be analyzed must be uniquely identifiable by visual features. A feature recognition can only take place, if distinctive features are visible on the animal. Leopards fulfill the above mentioned assumptions. For the leopard the third requirement is met by its pelage with the spots and rosettes pattern. Not all *Felidae* are uniquely identifiable by just a small part of their fur coat. The system would not work on cougars for instance, because of the missing coat pattern. And even less for lions, which miss the distinctive coat pattern for individuals and live in packs. For species that do not meet the required assumptions the system is not applicable. With the above mentioned independency of the animal's flanks in mind, it must be ensured that individuals are not counted twice into the population or right flanks are compared to left flanks in general.

Video data in which the leopard might be shown from different viewpoints are beneficial. Here the second core assumption comes back in. If for the most cases only one leopard is displayed in one video and if it is the same individual throughout the frame sequence, there is a chance to get footage of the leopard from different viewpoints. Initially all frames of one video containing a leopard are labeled with the same name ID for that individual.

The scheme in Figure 2.1 illustrates the concept on a pictographic example. *Video 1, Video 2* and *Video 3* were taken at different camera spots A, B and C. Initially we tag every frame of *Video 1* with the name ID *Leo 1* and the one in *Video 2* with *Leo 2* and analogue for V*ideo 3*. For simplicity only one prominent distinctive feature is highlighted

in each frame as a star. Stars of the same color display the same feature. In *Video 2* and *Video 3* the animal is shown from opposite sides. If only *Video 2* and *Video 3* are available the animals are incomparable. But, if the dataset offers a video with the characteristics in *Video 1*, the three videos become comparable. The first frames of *Video 1* can be matched with *Video 3* by means of the red star feature, whereas the last frames of *Video 1* match *Video 2* by the blue star feature. This implies that the captured image of the leopards in *Video 2* and *Video 3* belong to the same individual. Based on this knowledge the leopard in all three videos can be identified as the same individual.

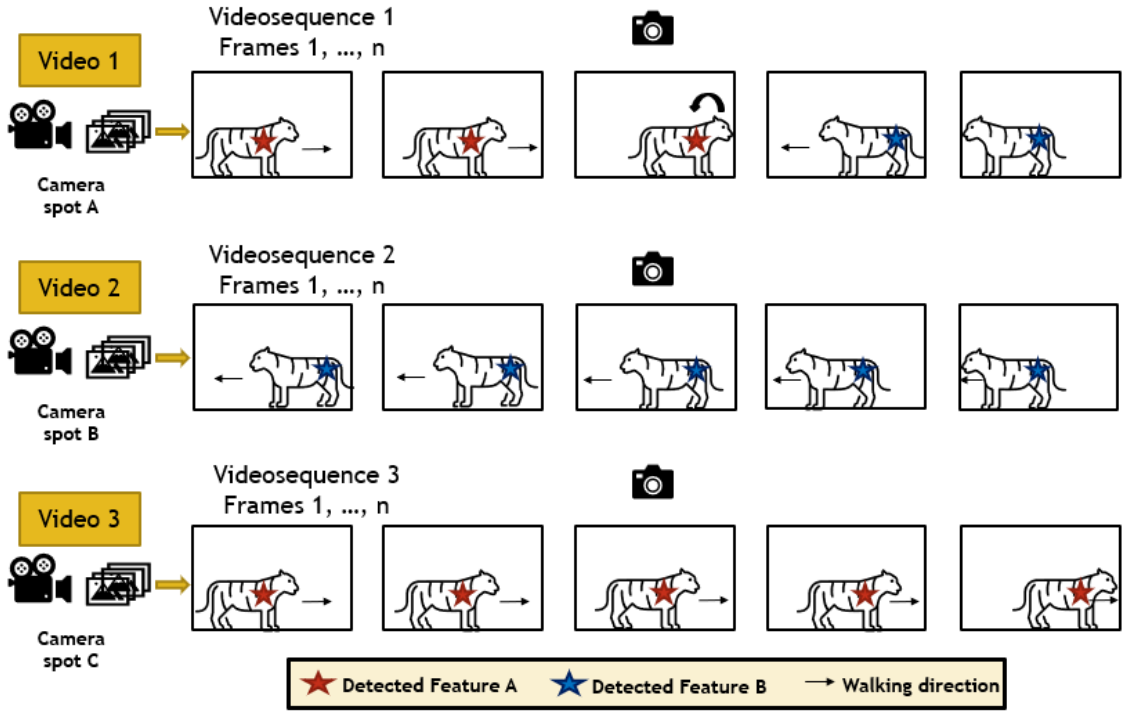


Figure 2.1 **Pictographic illustration of the concept**
The animal is only seen from one side in two of the videos. The third video shows the animal on both sides and enables a matching of the three videos, if the same individual is shown.

2.4 Definition of research issue

The objective of this thesis is to develop a program that automatically identifies and reidentifies individual animals with computer vision tools, provided the species fulfil the requirements explained in the last chapter. The program will be implemented in Python and demonstrated on the leopard dataset at hand. To minimize manual interaction, the program should not require labeling of the dataset of interest. Additionally it is aimed to use and combine algorithms that already have proven themselves for the use on wildlife data and, if necessary, to implement additional components that eliminate the otherwise needed manual interaction of a human.

3 DATA

The data foundation which is used to demonstrate the system in this thesis was provided by the PanAfrican Programme. The PanAfrican Programme hosts 39 temporary and collaborative research sites. For 29 sites the data collection is completed. Up to now, almost 400 000 video clips have been taken and 21 000 organic samples were collected by scientists. During the whole project the members mastered more than 3000km for recces, transects and maintenance visits [10].

The videos were taken by camera traps in different regions in central and west Africa across several countries, including the Democratic Republic of the Congo, Guinea, Ivory Coast, Liberia, Gabon, Senegal, Uganda, Congo and Guinea-Bissau. Figure 3.1 shows the regions in which the research sites are located and Figure 3.2gives a closer insight on the sites. The main intention of the project was to collect systematic ecological, social, demographic and behavioral data on 35 to 40 chimpanzee populations.

Figure 3.1 **Research regions**
The wider research regions in which the data collections sites are located [11].

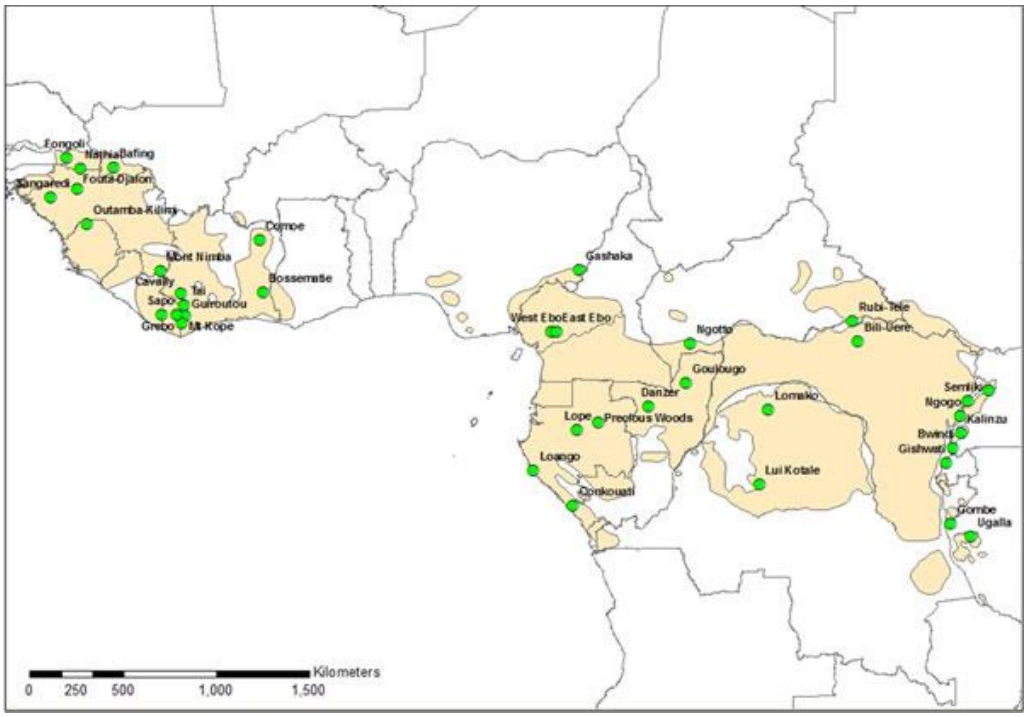


Figure 3.2 **Research sites**
Research sites with real names. On Zooniverse fake names are used [12].

To get the best results from a conservation research point of view systematic sampling was conducted. Systematic sampling implies that the camera traps are placed in predefined grids. Random sampling, in contrast to systematic sampling, implies that camera traps are placed at random locations. Randomly placed cameras rarely take useful images, because most species do not use their habitat randomly. Instead, they have spots they visit regularly, and travel paths they use to move in their territory. So far there is no official research available for leopards, but in comparison for chimpanzees it is assumed that a minimum of 50-80 good videos are required to estimate the chimpanzee density in an area. Camera trap studies at different sites have revealed chimpanzee visitation rates of about 1-3 events per camera and month, depending on local chimpanzee density. Thus 20 cameras at a study site should provide about 20 or more chimpanzee videos per month. However, only about 30% of chimpanzee footages are usable when it comes to the identification of individuals, which is roughly 6-7 videos/month [13], [14].

The data is captured in a predefined data collection zones of 20 to 100km² with a grid of cells of size 1x1 km each. The systematic camera placement requires one camera to be placed per grid cell (see

Figure 3.3). The available hardware includes 20 Bushnell cameras for every data collection zone. If the data collection zone covers more than 20 grid cells the systematical placement should be retained by placing them in every second cell. In addition to the systematic design, remaining cameras are non-systematically placed in promising high-activity area.

The Bushnell cameras are placed in Tupperware boxes and sealed with cling film for protection against wildlife damage or insects (see

Figure 3.3 left). Previous experience also shows that the high humidity and rain in the rainforest affects the hardware. The cameras are equipped with a motion and infrared sensor.

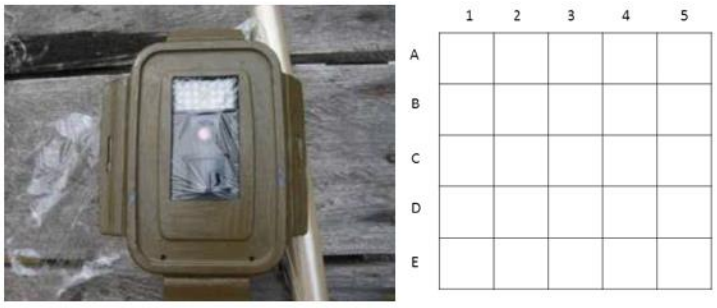


Figure 3.3 **Camera trap and grid cell**
Left: The camera traps are sealed and placed in a box for protection.
Right: The grid cell for camera locations [13].

Camera spots should be open enough to ensure that individuals can be filmed clearly without being blocked. Travel paths in general work well, especially at intersections. Elephant paths often suit the requirements well, but also bear the risk of the cameras getting damaged or moved, if an elephant rubs on a tree or tears it down. Therefore bigger trees for the mounting of the camera are preferable. Locations like water holes

or natural bridges are often visited by many different species. Natural bridges are fallen trees or other obstacles that build a path over a body of water. Natural bridges are an ideal location to collect data for identification tasks. Two cameras are placed facing each other at either end of the bridge. In this way it is likely to get footage from two viewpoints of one individual.

Due to the fact that the main target of the project was to collect data on chimpanzees the cameras were set in their favor. Scientists hiked the area looking for evidences on the recent presence of chimpanzees. Indications on their recent presence can be faeces, fur, food remains or sticks and stones that have indications like dents and scratches, showing that they were likely used as tools by monkeys. Open fields without trees are avoided by chimpanzees. Therefore no cameras were placed in those cells and no statement of the presence of leopards can be made. Leopards or carnivores in general may prefer different locations as chimpanzees.

For the most cases it can be said that the jungle is a complex application area to take image and video data. The images may have high differences in quality. Many objects can cause occlusions or false triggers of the camera traps and animals can be hidden very well in the footage. The cameras can be triggered any time during daylight and in the night with large variations in illumination. The missing light during night complicates the situation and the videos can be of a lower quality. The snapshots on the animal or animals itself can vary in distance, be blurred or very close up. Touches of animals to the camera are common [13], [15].

The triggered video clips have a duration of one minute each. While collection a large amount of automatically taken data it is important to have a systematical way to save it and to hold the entire metadata. Metadata includes the maintenance time, the GPS location in UTM of the camera spot, the cell in the grid and the camera in that cell, a site code and the site name.

Loango\Loa_d12\d12_Loa_cam38_012345_012345_20170912\09180056 .mp4*

Sitename\grid cell\camera in that cell_ geodata in UTM_maintenance date of camera\videoname.mp4

**geo data changed to hide real location and protect wildlife*

For the conduct of this thesis a subset of the dataset collected by the PanAfrican Programme is used. The videos of the subset are preselected videos including leopards tagged by volunteers and experts on Zooniverse [16]. The seen individuals were partly tagged with a unique name by experts, if the data allowed. The information on the individuals will only be used for the evaluation, not for the process itself. For secrecy reasons the site names were changed to fake site names and the GPS information not included for the publications on the Zooniverse platform.The subset encompasses 40 GB of footage from the time between 2012 and 2018 from over 250 camera spots and a total of around 800 videos.

4 FUNDAMENTALS OF COMPUTER VISION

The chapter outlines methods in the area of computer vision that are needed to visually identify and reidentify objects in images or videos. Methods from the analytical computer vision and machine learning approaches are considered.

4.1 Feature detection and feature description

A feature is a distinctive characteristic in an image. In the field of computer vision features are small areas that should be as identifiable as possible. Distinctive characteristics can be contrasts or edges for example. Feature detection is mostly applied as part of feature tracking tasks, meaning that a program tries to find the same object or feature in another image. Objects are collections of pixels that can move or can be changed in conjunction throughout images or in a video. Humans perform this task by intuitively scanning the image with their eyes for the object or the feature. For a human it seems easy to find the same object, also in another shape or pose. For example, a person or an animal that has moved to another pose or position. For computers this task is less intuitive.

A previous approaches in feature detection was template matching, in which an area was searched having the highest accordance with the template. But this method is limited in terms of scaling and rotation. To overcome this problems the same template was used in different scales and rotations, causing many combinations to the detriment of computing time. The issues caused by rotation, scaling and distortions can be solved with markers. A marker is an object known to the algorithm that was visibly placed in the scene during the data collection. With the marker as a reference the algorithm can calculate scaling, rotation and distortions. But the use of markers brings limitations. For some research cases it is not feasible to place markers on the objects of interest previous to the collection of data. This also applies for the use case investigated in this thesis.

Newer algorithms for feature detection and tracking are scale- and orientation invariant. The main components of feature tracking are usually the following: first a feature detector that identifies distinctive characteristics in an image (the features), second a descriptor to convert the characteristics into a vector and lastly a matching part that finds similar characteristics in another image by repeating the first steps on the new image and comparing the feature descriptors. Areas with similar characteristics lead to vectors with a small metric distance in the feature space, whereas dissimilar areas lead to large distances. Thus, the task of feature matching is to find a feature with a vector that is close to the feature vector of the region of interest that is tracked.

The following three algorithms belong to the most known in this are: ‘Scale Invariant Feature Transform’ (SIFT), ‘Speeded Up Robust Features’ (SURF) and ‘Oriented FAST and Rotated BRIEF’ (ORB). For this thesis the SIFT algorithm will play a central role and therefore it will be further explained in the next chapter [17], [18].

4.2 SIFT algorithm

The Scale Invariant Feature Transform algorithm, for simplicity from now on called SIFT, was developed by David G. Lowe in 2004 [19]. The British Columbia University holds the US-patent. Other than template matching algorithms, the SIFT algorithm can deal with rotation, distortion, changes in illumination and occlusions to a certain extend. Instead of searching for a whole object the SIFT algorithm searches for a few very descriptive unique features (keypoints), which can be reliably matched between different views of objects or scenes.

The first component of the algorithm is its feature detector. The features are detected by the following steps. First of all, the image is stacked up to a pyramid, where each level has a different scaling. Scaling down towards the top of the pyramid each level halves the resolution. For each level the image is copied multiple times and each copy is Gaussian blurred with different factors (standard deviations σ). The Gaussian blur eliminates some noise in the image. Inside one level of the pyramid the Difference of Gaussian (DOG) between two adjacent images is calculated (see Figure 4.1 left). The DOG offers an approximation to the scale-normalized Laplacian of Gaussian that is needed to handle scale invariance as stated in the research of Lindeberg and is low in computing time [20]. Still within the level the pixels with the maximum and minimum value are selected by comparing each pixel to its eight neighbors in the same copy of the image and respectively to the 9 pixels in the image above and below as illustrated in Figure 4.1 on the right. The pixel selected during this step are candidates for keypoints. In this way only the pixel that are generally unique and differ their neighbors enough in terms of intensity are selected. Additionally, a filter is implemented that disqualifies the candidate that are located on an edge. The reason is that pixels along an edge are too similar. The remaining points are the SIFT keypoints. As soon as the keypoints are detected the effect of scale can be removed by normalization with the known sigmas.

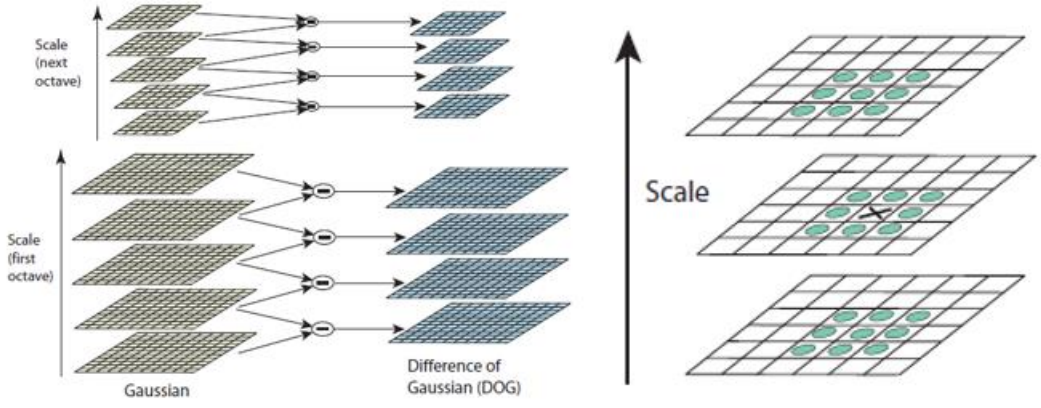


Figure 4.1 **Difference of Gaussian for pyramid levels & keypoint candidate selection**
Left: Two levels of the stacked pyramid for feature detection. Within each level the Difference of Gaussian is calculated for adjacent images of different Gaussian blurs. Right: Selection of pixels with minimum and maximum values as keypoint candidates [19].

The next step strengthens the algorithm against rotations of features in different images. To cope with rotation Lowe represents the keypoint relatively to its orientation. A grid is placed over the area containing the keypoint (see Figure 4.2 left, blue circle). For every pixel in that grid the gradient is calculated returning the magnitude and orientation between the pixels. The gradient visualizes the surface of the gray values 3D plotted on the (x,y)-plane. For every quadrant shown in Figure 4.2 the gradients of every grid cell are sorted into an orientation histogram consisting of 8 bins for the directions shown in Figure 4.2 on the right. Normalizing the length of the gradient, the magnitude, strengthens the algorithm towards changes in illumination. The largest bin of the orientation histogram states the principal orientation of the keypoint. The feature descriptor is the vector concatenating the values of all the entries in the orientation histograms. For the simplified depiction illustrated in Figure 4.2 a 2x2 array of orientation histograms on a region of the pixel size 8x8 is used. Most implementations of the SIFT algorithm work with 4x4 descriptors and an area of 16x16 pixel, resulting in a feature vector of the length 4x4x8=128, where 8 is the number of bins in the orientation histogram.

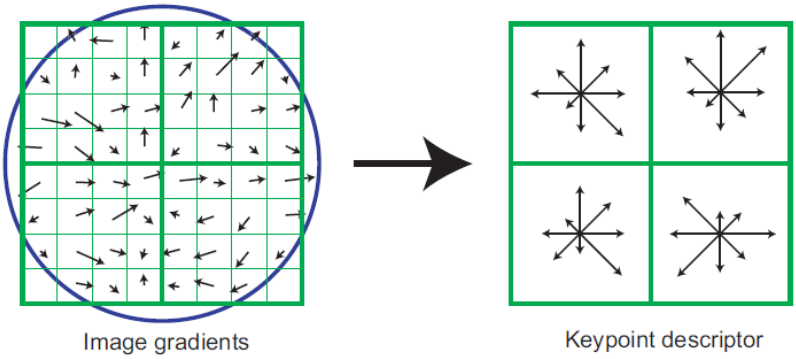


Figure 4.2 **Gradients of SIFT algorithm**
Left: Grid over the area containing the keypoints. Right: Directions of the orientation histogram.
2x2 descriptor on a feature region of size 8x8. Most implementations of the SIFT algorithm work with 4x4 descriptors and an area of 16x16 pixels. The histogram for each has 8 bins resulting in a feature vector of the length 4x4x8 = 128 [19]

After the feature vectors are learned they can be used for matching. If a new unseen image is given the algorithm starts to calculate the feature descriptors for the keypoints in the same manner. By comparing the new feature descriptors to the known ones in the database potential matches can be found. The best candidate for a match is most likely the nearest neighbor in the database. The nearest neighbor is defined as the keypoint with the minimum Euclidean distance between the two feature vectors mapped in the feature space [18], [19], [21], [22].

4.3 Convolutional neural networks

Convolutional neural networks, for the ease of reading from now on called CNNs, are a subclass of Deep Learning, a field in artificial intelligence. Deep learning algorithms are

extremely powerful machine learning algorithms and can learn to differentiate between arbitrary categories on images, if they were trained on those categories. Machine learning systems are generally composed of a dataset, model, loss function and optimization algorithm [23]. The model learns the relationship between input and output on its own based on the dataset. Neural networks are comprised of many connected layers consisting of nodes, inspired by biological neural networks.

Convolutional neural networks are a specialization of classic neural networks and a popular tool for computer vision tasks like image classification and object detection. The vast majority of applications of convolutional neural networks focus on image data. Convolutional neural networks have historically been the most successful of all types of neural networks. In 2016 it was proven that CNNs can outperform humans in the task of image classification [24], [25].

Analogue to classic neural networks the calculation of convolutional neural networks can be divided into the fundamental components of feedforward and backpropagation. During those steps the later described weights of the network are iteratively updated to minimize the loss function, which in turn measures a distance between a prediction and the ground truth. The detailed explanation on the fundamental theory of feedforward and backpropagation exceeds the scope of this thesis and are not further explained.

Even though deep learning algorithms, especially CNNs, are extremely powerful there is a downside. A large amount of training data with ground truth information is necessary, which usually requires some kind of labeling.

The specialty of CNNs is their ability to process image data and to detect patterns in the images. The architecture of a CNN is a stack of the three main types of layers, namely convolutional layers, pooling layers and fully-connected layers. Each output of a layer is the input for the next layer.

As an input it consumes images, which are usually preprocessed to be of the same size and a label for each image stating the CNN the ground truth. The labels are the catalogue of outputs the CNN learns, for example the different classes for a classifier. The input of the image is 3-dimensional, height and width of the resized image and the depth, which depends on whether the image is in gray scales in this case the depth is one or in color channels like RGB leading to a depth of three.

Within each convolution layer a convolution operation is conducted by a predefined number of 3-dimenesional learnable filters (in some literature also called kernels), able to learn and detect patterns. A filter is a matrix (tensor) of a fixed size. Its values are the weights. The filter can be visually imagined as a small window that is slid over the image row by row and calculates the dot product from the values covered by the window at each step. The sliding movement of the filter is referred to as convolving, giving the model its name. The resulting dot products form the feature map of the previous layer are then fed into the next layer. This procedure is illustrated in Figure 4.3. For the position on the bottom left the filter calculates the dot product as follows:

$$5 * 1 + 8 * 0 + 1 * 1 + 8 * 1 + 0 * 0 + 1 * 0 + 6 * 0 + 4 * 0 + 1 * 2 = 16.$$

A padding can be added around the image to enlarge the edges of the image so that every pixel is consumed equally and the edges and corners are not neglected.

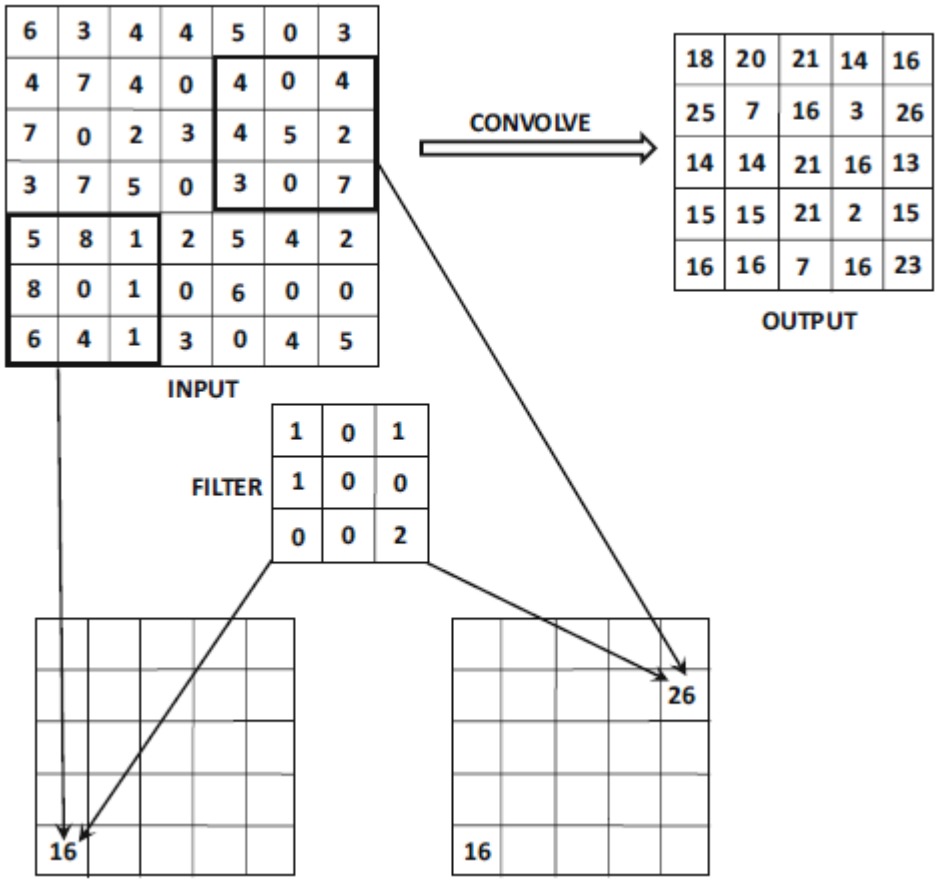


Figure 4.3 **Convolution operation in a convolution layer**
A filter matrix with weights convolves over the matrix of the image and calculates the dot product [24].

The first layers of the network recognize primitive shapes like edges, corners and circles. The deeper the convolutional layer into the network, the more sophisticated features can be recognized [24]. Depending on the use case and the input images those features can be fur or feathers for animals or wheels and other components for cars. Figure 4.5 visualizes which filters detect which patterns. The example shows a handwritten 7 from the MNIST dataset (see Figure 4.4) [26]. The values in the matrixes can be decoded by color with -1 being black, 1 white and 0 as grey. The abstracted versions of the original image of a handwritten seven shows the output after a convolution operation with the according filter (see Figure 4.5). The image of a handwritten 7 is a good visual example for how the filter detects the horizontal and vertical edges of the relevant parts.

Figure 4.4 **Handwritten 7 from the MNIST dataset**
[26]

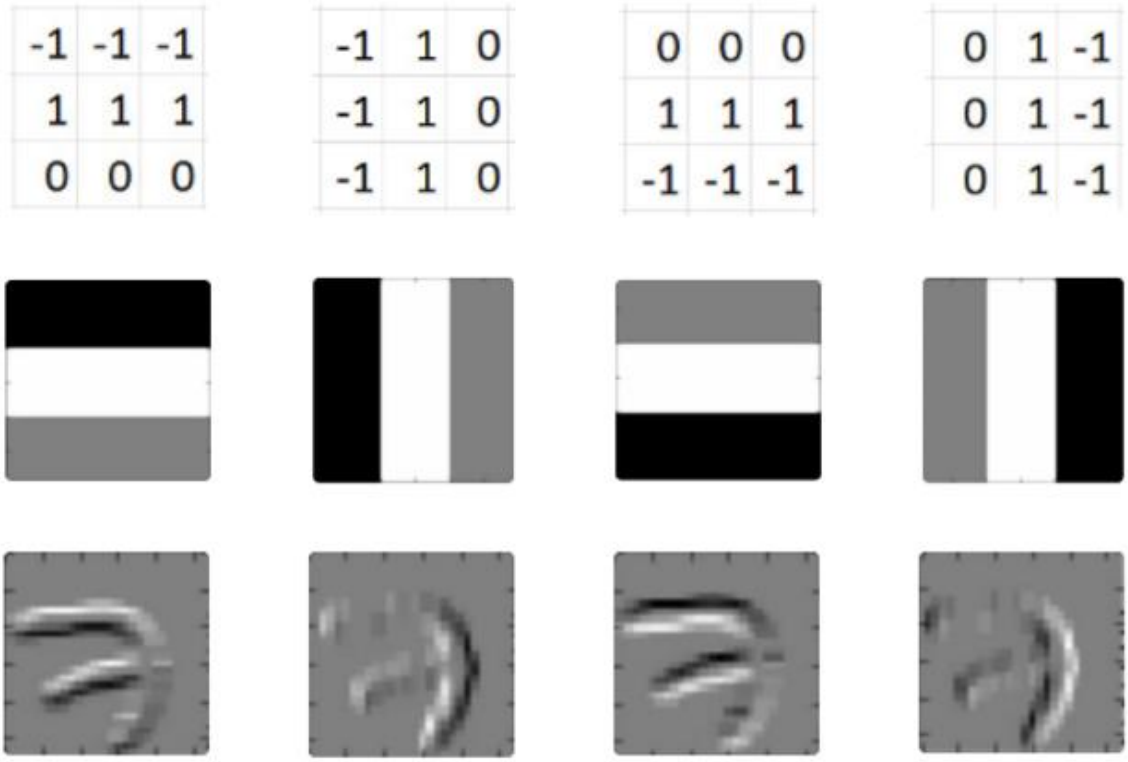


Figure 4.5 **Different filters in a convolutional layer**
Different filters and the patterns they detect. The filters detect horizontal and vertical edges [27].

In a CNN architecture pooling layers are periodically inserted between the convolution layers. The operation of a pooling layer downsamples along width and height of the filters. The most common type of pooling is the max pooling. Here a filter of a predefined size with a predefined step width convolves over the image and returns the maximum of the observed window at each stop. In this way the pooling layer progressively reduces the spatial size of the original input data through the layers also in favor of computing power and to limit the risk of overfitting. The original input image gets transformed through the layers into a final score leading to a classification. Overfitting occurs if the network is too specialized on the training data and does not perform well on new unseen data. Overfitting is often caused by too limited training data. An example of a pooling layer operation is demonstrated in Figure 4.6.

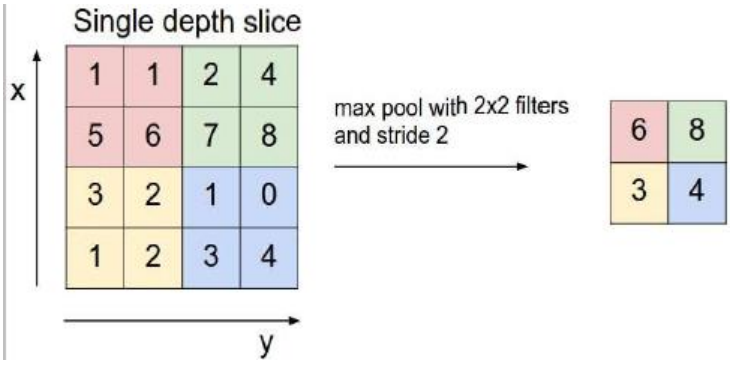


Figure 4.6 **Pooling layer of a CNN**
Pooling reduces the special size of the data selecting the maximum at each stop of the convolution [28].

The last layers of the network are fully connected layers. The spatial layout of the pixels in multiple dimensions from the hidden convolutional layers is no further needed. To transform the spatial data into a one-dimensional vector it is flattened:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \xrightarrow{\text{Flattening}} (1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \quad 9)^T$$

The fully connected layers are of the same type than in a classic neural network. The very last layer consists of nodes representing the classes. When new data, meaning an unseen image, is fed into the CNN a predicted likelihood for each class is returned.

When a CNN is built from scratch all initial weights are random. Transfer learning can improve the performance of a CNN and saves time on training. The idea of transfer learning is to use a pretrained network and to retrain the existing weights. The weights are already calibrated on previously learned simple and general features and patterns. Either all weights are further retrained, the weights from the last fully connected layers or just the last layer are retrained. All other weights are frozen. This procedure saves computing time and requires less data to train. Nevertheless, the weights for the final classification must be trained on the classes of the dataset at hand to match the desired classes for the application case.

The performance of a CNN can be improved by hyperparameter optimization. Hyperparameters manage the trainings process by means of batch size, epochs or the chosen optimizer. Furthermore, the technique on how training and test data is split for example cross validation or data augmentation can enhance the performance. Dropout layers can be added to prevent overfitting. Those topics exceed the scope of this thesis and are no further discussed here [23].

Currently the most popular libraries for CNNs or neural networks in general for Python are Keras, TensorFlow and PyTorch [29]–[31]. In the implementation for the leopard identification Keras and TensorFlow are used.

4.4 Pose estimation

Pose detection and estimation means to capture the posture of a human or animal being and is a branch of the computer vision field. The ability to track body movements improved many fields of application including healthcare, fitness and robotics.

The algorithms have to deal with a large number of visual appearances, because humans can exhibit a wide range of variations on poses. For different animals this number of variations is even larger. Back in 1973 it was already researched that only a few keypoints moving as a unit are sufficient to detect human motions. Motion of living beings can be abstracted as a geometric structure formed by several pendulum-like motions of the extremities relative to a joint [32]. As figurative imagination: the knee when dangling legs.

Keypoints are the points of interest, for example body parts, and are represented by coordinates in an image space. Motion capture systems aim to detect and track those keypoints from videos. For pose estimation of living beings body parts like knees, snout, elbows or eyes can be relevant keypoints. What is defined as a keypoint tremendously depends on the application case and the object of interest. Pose estimation on humans is further developed than on animals. For humans much more training data exists due to the fact that they are usually more cooperative than animals in the data collecting process.

To assist the tracking of objects in computer vision tasks, markers can be attached to the objects of interest. Those markers display a known feature to the computer-based tracking system that can be rediscovered. Another option for marker-based systems is to highlight the parts of interest by colors or light. In any case this method requires preparation that must be considered before collecting the data. Depending on the specific application the object of interest may not be cooperative or even known before caught on camera. For instance, when collecting data in the wild it is impossible to make preparations on the animals that might cross the path of the camera.

For use cases in which no preparation was done or cannot be done, markerless pose estimation algorithms were developed. For this approach no preparation on the object of interest is necessarily required, instead ground truth labeled trainings data is needed. The advantage is that this can even be done after collecting the data, but the downside is that this procedure generally requires manual labeling of the keypoints in the data.

Pose estimation algorithms are basically object detection algorithms. On the one hand there is an encoder that extracts visual key features from the data, learning poses. And on the other hand is a prediction model, a decoder, that predicts the locations of the learned keypoints in new unseen data. A convolutional neural network (see chapter 4.3) can learn pose estimation [17].

For the loss function, the pose prediction can be evaluated by the distance between the ground truth keypoints and the keypoints in the prediction [33], [34]. Alternatively, the

percentage of correct predicted keypoints can be evaluated, counting the keypoints that are within a defined distance to the ground truth keypoints [35], [36].

Several open source packages for pose estimation exist for Python with different pretrained weights on different large datasets. Open pose, DeepPose, DeepCut and DeeperCut or modules part of the OpenCV library are the most prominent ones [37]–[41]. But no matter how large the training datasets are even the best architectures fails to generalize to “atypical” postures that are not displayed in the trainings dataset as illustrated in the error collection on yoga poses predicted by OpenPose [42]. The challenge in pose estimation on animals is complex due to the large amount of poses an animal can take.

5 TOOLS FOR AUTOMATED IMAGE PROCESSING OF ANIMALS

The chapter presents tools that have been developed for image processing on animals. The tools are chosen to be embedded into the program, because they have stand the test in other case studies. The MegaDetector and IBEIS have been successfully used on data taken in the wild.

5.1 MegaDetector – Detector

Conservation biologists invest a huge amount of time reviewing camera trap images. Not only for identification, reidentification or classification, much time is spent on sorting out empty images. The MegaDetector aims to accelerate this process. The MegaDetector is part of Microsoft's AI for Earth program that supports groups working to solve global environmental challenges with AI tools and computing power [43].

Previous work has shown good results on classifiers for species with camera trap data, but the problem is that those models do not adapt well to new geographical regions and other species, because they were trained on specific species from that area. If the classifier is used in an ecosystem with species unknown to the system, those cannot be classified. But also for known species different backgrounds due to another region are hard to handle for the systems. The aim of the MegaDetector is to find the lowest common denominator for researches with camera traps. One task that has to be done by all applications is sorting out the empty images caused by a false trigger. This number strongly depends on the project, but is estimated to be around 70% on average. The MegaDetector was trained to be a generalizable detector that can be applied in new regions. The system detects the object and classifies it into one of the three classes: animal, human, vehicle. Cameras can be also triggered by humans or vehicles, depending on the project this data is of very high interest (i.e. to fight poaching) or irrelevant for the study (i.e. count of species). The MegaDetector only detects animals, but does not classify them by species. If an animal is detected the four values of the bounding box around the animal are returned. A visualization of detected bounding boxes is depicted in Figure 5.1. All annotations can be exported in a JSON file. If the results are not satisfactory, it is possible to label a small dataset for this region and feed it back into the system for fine-tuning [44], [45].

The first version of the MegaDetector was released in 2018. Since then it was constantly improved. The current model, the MegaDetector v4.1 was released in April 2020. It is based on Faster-RCNN with an InceptionResNetv2 base network, and was trained with the TensorFlow Object Detection API. The network was trained on millions of pictures from different ecosystems from over 23 datasets. Across all datasets approximately 75% percent of the images are labeled as animals, 5% as vehicles and 20% as humans. Additional data sources are incorporated for re-training. Two of the largest datasets are the Caltech Camera Traps dataset and Snapshot Serengeti. The Caltech Camera Traps dataset contains 243,000 images from 140 camera locations in the United States. The

dataset from the Snapshot Serengeti dataset contains an incredible amount of over 7 million images from different sites. In this dataset approximately 76% of the images are labeled as empty. The dataset includes 150,000 bounding box annotations for 78,000 images [46], [47]. Thanks to the diverse input data the MegaDetector is able to detect several individuals in one image. Furthermore, the team of MegaDetector collaborates with LILA BC (Labeled Information Library of Alexandria: Biology and Conservation). LILA BC is a repository for data sets related to biology and conservation, intended to bring together people with machine learning expertise and people or organizations that have the data and would provid from automatized processing [48].

The code is freely available in a GitHub repository [47]. The model is well documented in terms of use, but no technical documentation on the model itself is available. The project focuses more on the user side for ecologists and conservation biologists with detailed manuals and videos. For the first steps an easy to use web-based demo is available. A few images can be quickly analyzed by drag-and-drop. For larger datasets only limited programming skills are required. A template for the application of the model is offered in Google Colab. Running the analysis in Google Colab has the advantage of using Google's GPUs [45], [49], [50].

MegaDetector's infrastrutcture is extremly efficient on millions of images by dividing them over multiple nodes. To accelerate the process MegaDetector can be run on GPU, but having a GPU is not a requirement. The general estimation for a laptop from the mid-range segment without GPU that is sold in 2021 is 8-20 seconds per image, adding up to 4,000 to 10,000 images a day. In comparison for a GPU from the mid-range segment the MegaDetector need 0.3 to 0.5 seconds per image adding up to 200,000 and 250,000 a day. For a human it would probably take several weeks to review the amount of images. The above mentioned statistics make the MegaDetector also suitable for little research laboratories, wildlife reservoirs or private users.

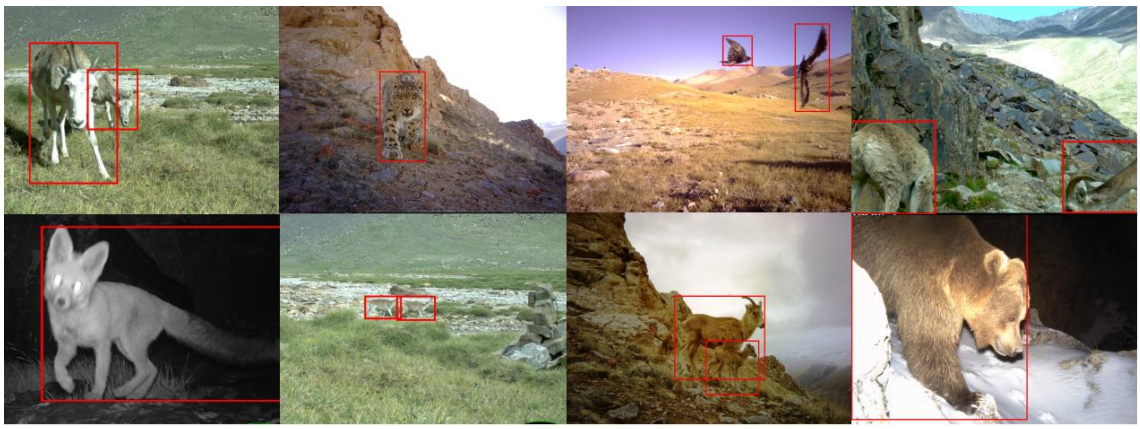


Figure 5.1 **Detections by the MegaDetector**
The MegaDetector detects animals in frames and put bounding boxes around the location. Multiple animals can be detected in one frame [44].

By the time this thesis is written the MegaDetector works on a classifier prototype, but it is not available and not known yet for which species it will work.

The MegaDetector is used in the Wildlife Protection Solutions project in real-time to support anti-poaching scenarios. As soon as a vehicle or a person is detected in certain areas an alarm is triggered, enabling the organization at site to protect threatened animals and arrest poachers[51].

5.2 DeepLabCut – Pose estimator

DeepLabCut is a freely available software package for Python 3.6.x and higher developed at Harvard University and the Mathis Group & Mathis Lab at EPFL [52], [53]. It offers a deep neural network system for pose estimation making use of transfer learning. The newest version includes a graphical user interface and is compatible with Linux Ubuntu, Windows 10 and MacOS. A Python environment with tensor flow is required.

The DeepLabCut algorithm can track user-predefined body parts in multiple species, extracting the pose of an animal with limited training data. DeepLabCut builds on a state-of-the-art human pose estimation algorithm, DeepCut. The software package of DeepLabCut comes with an out of the box tool that allows the user to create his own project where the user can build an own training dataset by labeling own data. A new individual network can be created by retraining the given neural network from DeepLabCut with its weights. In this way the user can define the keypoints and decide which parts of the animal are of interest for each project. Thanks to transfer learning only limited annotated data is required from the user. Only a few hundred annotations are sufficient to achieve good results for different pose estimations.

DeepLabCut is is a deep convolutional neural network (CNN). The network consists of a variant of DeepCut and its successor, which in turn is based on ResNet. With the difference that ResNet's last layer, the classification layer, is replaced by deconvolutional layers, which up-sample the visual information and and produces spatial probability densities, which shows the probability density of a body part benig present in a location. ResNet itself is a concolutional neural network that was trained on the ImageNet dataset. The dataset consists of over 14 million labeled images and many subsets. It is used for many other computer vision tasks and is often used as a benchmark for object recognition [54]. DeepCut is a state-of-the-art algorithm for human pose estimation meeting all the benchmarks on the popular pose estimation datasets [55]. DeepLabCut utilizes DeeperCut's feature detectors consisting of either 50 or 101 layers and its deconvolutional layers [56].

Thanks to the deep learning-based pose estimation, DeepLabCut can deal with varying and busy backgrounds as well as inconsistent illumination or distortions. Consequently no simplifiying of the data collection area is required like blank monochromatic backgrounds and high contrasts. The algorithm is more robust against occlusion or motion blurs, because no consistency is required for the features over a frame sequence. As soon as the feature is visible it can be detected. Other methods, like the Lucas-Kanade method for instance, rely on the information gathered over a sequence of images [57], [58].

In a nutshell, DeepLabCut is a specialized version of DeepCut or rather DeeperCut on animals. From this starting point the user can fine-tune an own network for an individual project by labeling a subset of frames from the own data with the keypoints of interest and retrain the weights of the basic network. It is recommended by the developers of DeepLabCut to label maximally diverse images (i.e., different poses, different individuals, luminance conditions and backgrounds), if applicable. After this step is completed the trained convolutional neural network can be used to estimate poses on unseen frames. The predicted labels can be manually corrected for further fine-tuning, if wished, adding an active-learning loop. Nevertheless, no system is perfect and every deep learning algorithm is only as good as its input and labeling. The pitfall of minimal data labeling is that labeling errors can severely disadvantage the performance. Another issue that has been experienced with pose estimation are flipped labels of right and left of the same body part.

For the evaluation of the predicitons the manual labeled data is assumed as a ground truth. In other words the DeepLabCut method aims to match human labeling accuracy. The preciseness of the pose prediction is quantified by the root mean square error (RMSE) to the human labeling.

DeepLabCuts first application case were mice following odor trails. They extracted 1080 frames from multiple videos and manually labeled the keypoints: snout, left ear, right ear and tail base. The frames included 7 different mice captured from two camera spots. Nice surprise was that, although the network has only been trained with images displaying a single mouse, it reacted well on unseen images with multiple mice interacting [34].

DeepLabCut offers pretrained weights on different species in its Modelzoo. At this time the weights for the following species are freely available: cats, dogs, primate faces, macaques, cheetahs, humans, horses, rodents and pupils of mice [59]. For this thesis the weights of the net for cheetahs will become relevant due to their similar body proportions to leopards.

With the DeepLabCut system not only the main body parts like legs, head and tail can be tracked. Everything can be tracked, if it can be labeled in the trainings data [60], [61]. Preceding projects focused on tracking of pupils on mice. Pupil tracking is of great importance for visual neuroscience. A study used head-fixed cameras and DeepLabCut for pupil tracking to reveal two distinct types of coupling between eye and head movements. The International Brain Lab implemented DeepLabCut for pupil tracking for the research on decision-making on mice [17], [62], [63].

Another enhancement that has been researched for DeepLabCut is 3D pose estimation with multiple cameras. One network is trained on a combination of cameras, which generalizes across the views. DeepLabCut was coupled with standard camera calibration techniques for 3D locations resolving [41], [64]. The 3D pose estimation was independently investigated on a cheetahs in the wild dataset and Drosophila, known as fruit flies, in a 3D behavioral chamber [34], [65]. At present the 3D pose estimation is

not relevant for the work of this thesis, but could be worth trying for a future enhancement.

An advantage to mention from the technical point of view is that DeepLabCut can run on GPU (Graphics processing unit), if the hardware is available. The processing on GPU speeds up the process by a factor of 10-100 [64]. Another technical benefit is that DeepLabCut does not require images to have a fixed frame size, as the feature automatic rescale during training.

If a user does not have a suitable GPU it can be either run on CPU, but much slower, or on Google Colab using Google's GPUs to speed up the process [66].

As many other research groups, DeepLabCut uses the support of volunteers for labeling and developed an app in this scope [67].

5.3 IBEIS and HotSpotter – Individual identifier for animals

IBEIS (Image Based Ecological Information System) and HotSpotter are computer programs for the individual identification of animals. The underlying algorithm uses the SIFT algorithm and a nearest neighbor approach.

IBEIS is the further developed version, but still includes the HotSpotter program as its centerpiece. Both were developed by Jonathan Crall in the course of his dissertation [68]. Further development and maintenance of IBEIS are taken over by the WildMe organization [69]. WildMe has set itself the target to fight extinction by building open source software based on machine learning and artificial intelligence techniques. WildMe hosts several datasets on multiple species, the WildBook.

The original HotSpotter application is an out of the box version with a Windows installer [70]. For the identification process the user must import the images and define a region of interest and the orientation of the animal within that region by putting a rectangular bounding box around the animals (see Figure 5.2). Each region surrounded by a bounding box is added as an annotation to the database. The whole process takes place in a simple graphical user interface (GUI). The successor IBEIS is available as a Python module with an open Github respository, but only in a Linux environment [71]. IBEIS also comes with a GUI. The GUI offers a few more functions than the one from the simple HotSpotter. The core matching algorithm remains the HotSpotter algorithm. The newer IBEIS program works with the same type of annotations as described above for the HotSpotter. The target is to label each annotation with a name that uniquely identifies the displayed individual and collect them in a database grouping the annotations that show the same individual under one name.

For each annotation keypoints are located. The features of the keypoints are extracted and described with the SIFT algorithm. In order to speed up the matching of the feature descriptors from the annotations in the database their IDs are indexed for a nearest neighbor search based on the feature descriptors.

When a new annotation is added to the database it can be queried against existing annotations to figure out whether the individual in the annotation is a new unseen one or already known to the database. Again, the viewpoint is a challenge. Two annotations are only comparable, if they both include a region with distinctive patterns visible in both annotations. If not, no statement on similarity can be made.

Figure 5.2 **HotSpotter input image**
Input image with bounding box and orientation

The upgraded IBEIS version offers an additional function. The imported images can be grouped into *occurrences* — a term for biodiversity data standards defined by the Darwin Core. An occurrence groups images that were taken within a small time frame and in a near location. It is assumed that only a small numbers of individuals is seen in an occurrence [72]. In favor of computing time it can be chosen to query the annotations within one occurrence against each other first, because it is more likely to see the same animal within one occurrence. Subsequent to the intra occurrence query the annotations can be queried against the master database to determine, if an annotation shows an already known individual to the database or an unseen individual.

In general, the output of the query is a list of matching candidates ranked by a matching score. A high score is an indication that the annotations queried against each other have a high likelihood of displaying the same individual. If the highest ranked annotation has a low score it is likely that the animal in the queried annotation is unknown to the database. The user of the software has to make the final decision which annotations should be matched.

From the technical point of view: the ranking score is aggregated from different components. First, a nearest neighbor algorithm calculates the correspondences between the queried annotation and the annotations in the database, based on the L2-distance of the feature descriptors. The so won scores of the annotations, describing the similarities between the single annotations, are aggregated into a name score. The name score for that name ID in the database is the highest correspondences between the queried annotation and all the annotations belonging to this name ID. Resulting in a single score for each name in the data base and the queried annotation. Further mathematical components contribute to the final name score measuring distinctiveness compared to the database and a value for the probability of the feature belonging to

the foreground and not the background. The further theoretical construct of the HotSpotters matching algorithm can be read in the dissertation of J. Crall .

The IBEIS system was used in a huge case study called the ‘Great Zebra Count’ and took place at the Nairobi National Park on March 1st and 2nd, 2015. The goal was to estimate the number of plain zebras (*Equus quagga*) and masai giraffes (*Giraffa tippelskirchi*) in that area. To make wildlife conservation more popular volunteers were included in the data collection. They equipped the volunteers with GPS dongles and assigned them routes through the park. On this routes the volunteers tried to take as many good images of the zebras. To overcome the challenge of opposing sides the photographers were instructed to only take pictures from the left side. During the event 9406 useful images were taken, resulting in a dataset of 48 GB [73].

Other application areas of the IBEIS algorithm included Grévy zebras (*Equus grevyi*) and even humpback whales (*Megaptera novaeangliae*) or manta rays (*Mobula birostris*) and seadragons (*Phyllopteryx taeniolatus*) [74].

6 RELATED WORK

This chapter outlines works and case studies related to the work of this thesis. The methods mentioned here severed as a base for the concept and pipeline or state reasons why such a method is probably unsuitable for my application task, even if they were used in similar tasks.

6.1 Computer Vision in animal detection and classification

Camera traps are a cheap and popular tool for ecologists, because large amounts of data can be produced with manageable preparation time. Over the past years an immense amount of videos and images was collected, which exceeds the processing workload that can manually be accomplished by experts. The overlap of computer science and wildlife conservatory is a growing field with many platforms and communities arising in the Internet. Large well-known companies like Google and Microsoft support projects with their expertise and hardware [43], [75].

Non-prefiltered datasets taken by motion triggered camera traps in the wild include many empty frames and different species. An automated detector and classifier is vital for ecologists to save time to cope with the large amount of data given. While the field of re-identification of individuals for animals is still in its infancy, the classification of different species has been investigated over the past years [76]. Automatically taken datasets usually include a spectrum of many different species living in the ecosystem where the camera traps were set up. Often it is of interest to analyze only one specific species, even though from the biological point of view it is very important to have an overview of the different species and their prevalence in an area. In both cases some kind of classification has to be done, either for the species or just dividing into the classes ‘animal’ and ‘empty’. Manual classification is time-consuming [47]. For datasets with thousands or even millions of records the processing of the data lasts years [46].

6.2 Citizen science

Platforms like Zooniverse [16] offer scientists to publish their data and include volunteers in the process. The volunteers become so called citizen scientists. With this approach many organizations and projects can process the data much faster with a positive side effect to draw the public’s attention on wildlife conservatory.

The Snapshot Safari project with its flagship Snapshot Serengeti was the first Zooniverse camera trap project in 2012 and is one of the world’s largest camera trapping initiatives on a continent-wide scale, bringing together conservation organizations across many countries. More than 1500 cameras are deployed in over 40 protected areas in Africa. Together they generate millions of images per year. The sheer amount of data is not manageable for the researchers, therefore volunteers were included in the labeling process. The labeling process automatically evaluates itself. A label is only valid, if 10-25 volunteers labeled it into the same class and 5-10 volunteer labels are required for the

simpler task to classify the image as ‘empty’ or ‘non-empty’. From 2013 to 2020 over 138,000 volunteers across the globe have labeled more than nine million images in the Snapshot Safari project [46], [77], [78].

The idea of citizen science is to generate a data basis for the training of machine learning algorithms to support wildlife conservatory. The labeled data is freely available on LILA. The LILA platform hosts many labeled datasets. The label mostly include species classifications, but very few datasets have annotations for the individuals (zebra, giraffes, amur tigers and whale sharks) [48].

Solely on Zooniverse, over 635 million classifications were made by almost 2.4 million volunteers in different projects so far [16]. Other large citizen science projects are the ‘Great Zebra Count’ containing images of plains zebra (*Equus quagga*) and masai giraffe (*Giraffa tippelskirchi*) with individual identifications and bounding boxes [48], [79].

Some projects remove sensitive information for privacy reasons for humans or to keep the location of endangered species a secret from poachers. In general citizen science brings benefits and problems. Volunteers usually do not have the knowledge of experts with many years of experience, which can result in inaccurate labeled data. Especially for rare, not well-known species. But the time-saving fact for experts far outweigh this drawback[80]. Most of the platforms including citizen scientists in their process provide comprehensive guides on how to classify the animals to limit mistakes.

The idea of adding volunteers to the process was taken to the next level in other projects concentrating on the monitoring of animals in specific regions. A reservoir in South Africa places camera traps throughout its field and uploads the data near-realtime on a website. When volunteers see a vehicle or a suspect human on the images they can trigger a warning. With this method the reservoir gets warned for poachers early and can react to poachers before they become active killing wild animals. With volunteers from all over the world the night bearing the most risk is covered as well. Besides triggering warnings the user can categorize the seen animals in an image by a given dictionary of animals. This data is later used to train a CNN as an animal classifier given the different species in that area.

Not only the data for labeling is shared online for other scientists, citizen scientist or computer vision enthusiasts. Conversely, platforms exist that offer a ready-to-use system in the internet browser to classify own data. On ‘ZambaCloud - Computer vision for wildlife conservation’ a user can upload an own dataset and have it classified for the given species [76].

Another effort that was made to get experts and people interested in machine learning on board are public challenges. Usually a dataset is provided and the relating research issue stated. After the deadline the best team is selected. Some challenges offer an award for the winner. A popular platform for machine learning challenges across many application areas is kaggle. This year they hosted a challenge on counting the numbers of individuals in camera trap data[81]. DrivenData is a provider that hosts challenges in

which a global community of data scientists competes. They hosted the Hakuna Ma-data challenge for the identification of wildlife had over 800 participants and a prize of $20,000 from Microsoft AI for Earth [43], [82].

6.3 Convolutional Neural Networks for Image Classification

Deep-learning methods have revolutionized our ability to train computers in the computer vision field of object recognition and classification, including many use cases like human face recognition, support in autonomous driving and wildlife conservatory. Deep learning methods can reduce human effort substantially.

The open source library MML Spark combines the Microsoft Cognitive Toolkit with the distributed computing framework Apache Spark. Microsoft additionally integrated OpenCV for image-based deep learning applications. As a prove of work they tested the system on the snow leopard dataset of the Snow Leopard Trust [83] and trained a deep neural network based on the ResNet50 taking advantage of transfer learning. They provided the organization with an end-to-end solution program for the automated classification of their camera trap data to identify snow leopards [84].

A group of students from the Sri Ramakrishna Engineering College in India trained a Convolutional Neural Network to classify 48 different species including cheetahs and elephants. The CNN was trained on data pre-labeled by volunteers. They reached an accuracy of 93.8% on the Snapshot Serengeti dataset [77], [85]. Another research team used a long-term database of more than 3 million labelled pictures to train a CNN to automatically recognize 48 African animal species [86]. The Zamba classifier mentioned in the previous chapter is based on a CNN as well [76].

Another team succeeded for identifying the three common species taken in a habitat in Australia comparing the Convolutional Neural Network architectures AlexNet, VGG-16 and ResNet-50 reaching similar accuracies for the different architectures. Before conduction the classification itself they used another CNN to classify into classes: *'animal'* versus *'non-animal'*, which in this case replaces the detector. The downside of this procedure is that the prediction only states whether a detection is expected in the image and not where it is located [87]. In another study the researchers focused on limiting the pre-labeled data that is needed to train Deep Learning algorithms by using transfer-learning methods. They demonstrated how trained models can be applied to new unknown data in combination with a low number of human classified images. The few needed manual annotations were also won in citizen science projects. For the training of the convolutional neural network the weights of the convolutional layers were copied from previous projects. Just the fully connected layer was retrained with the little available labeled data at hand [80].

The drawback of CNNs and often a problem in the use with wildlife data, is the openset problem. A traditional classifier can only sort into a dictionary of known-classes it was trained on. The classifier will always pick the class that fits the most, even if none of the classes fits from a human perspective. Studies on the open-set class try to overcome this

issue, but have not been used on wildlife data to my knowledge at the point this thesis was written. Open-set classifiers are able to classify into known classes but also to detect outliers in the dataset and identify them as unknown [88]. For the rare case of capturing an unknown species it would be sorted into the class of the best matching species.

For wildlife conservatory the rare species are often of particular interest. The problem is that for those species the least data for training is available [89].

6.4 Pose estimation

Nature's beauty in the versatility in shape and motion of living beings poses a problem for the automatized procession of camera trap data. Computer vision methods for rigid objects and steered movements in industry machines are often not prepared for the nearly infinite amount of poses and movements from living creatures. Research has previously been done for pose estimations for humans and animals. The insights of those studies can be of value for the economy, e.g. for robotics and healthcare and also for the analysis of the orientation and pose of an animal in an image [90].

In general, due to the fact that the generation of training data on humans is easier to generate, the machine learning methods for humans are more developed than for animals. Techniques in this application field already reach estimations in realtime. The most popular system for multi-person 2D pose detection is OpenPose. The open-source software localizes anatomical keypoints for the whole body, hands and faces of humans for video and image data.

For the application on data containing human beings, work exists to not only predict single poses, but of multi-persons and their interactions. Unstructured pairwise relationships between body parts of different people are encoded by a set of flow fields, called part affinity fields [37], [91].

A few studies exist that applied and further developed the knowledge won in researches with human data. A dataset including 30 horses of different breeds was generated. 22 body parts were manually labeled in 8114 frames to train a convolutional neural network. Different models of neural networks, including MobileNet, ResNet and EfficientNet, were tested on the 30 known as well as unknown horses. All models performed better on the known horses, but reached accuracies of up to 88% und unseen horses [92].

Interestingly a large amount of the researches in the field of pose estimation for animals focus on 3D estimations. Three-D Safari used a neural network to predict 3D poses and shapes of zebras using images taken in the wild and digitally generated data. For humans it was tried to translate 2D pose estimations from frame sequences to 3D pose estimations without having ground truth information on real 3D data [93].

To conquer the problem that animals are less cooperative than humans when creating training data, a group generated training data with 3D scans of toy figures of four-legged

mammals to train their model. Another similar approach was explored by generating the training data with a FBX animation file of a cougar from which the 2D and 3D coordinates of the skeleton were extracted and trained into the OpenPose model [94], [95].

Further ideas were explored for a cross-domain adaption. It is impossible to have a labeled dataset on poses for every existing species. Animals of different species may still share similarities in proportion and size, e.g. limb proportion for animals walking on 4 limbs versus animals with an upright posture like some monkeys [96].

The research supported by the South African National Research Foundation dealt with the 3D pose estimation of running cheetahs. The dataset for their 3D training, namely 'AcinoSet', is freely available and includes 7588 images with ground truth information on 20 fixed body part keypoints of the cheetah, e.g.; neck, head, tail, right knee front [65], [97], [98].

6.5 Reidentification and recognition of individuals

The reidentification and recognition of individuals differs to the above task of species classification. In the above mentioned classification tasks an animal is classified as a species, while for individual recognition this very specific individual is recognized being labeled with a name ID.

6.5.1 Applications on humans: Person re-identification

Computer vision, pattern recognition and machine learning are a widespread use for person reidentification with many application areas ranging from security and surveillance to retail and healthcare. Person reidentification methods often relying upon conventional biometrics such as face recognition. For face recognition high resolution images are required and depending on the captured data are not always given.

CNNs are a go-to solution for many works targeting person recognition. A large amount of CNNs were trained on that purpose [99]. Many open labeled datasets are available in the internet including humans [100]–[103]. But nevertheless, the use and training of deep neural networks for reidentifiation of individuals is tricky, because a deep neural network requires a large labeled dataset [104]–[106]. For the application on images with humans, approaches exist where trained CNNs are used for person reidentification in unseen domains and minimal fine-tuning of a pre-trained CNN is done [107], [108]. For animals this is hard to apply, because the pre-trained CNNs that could analogously be fine-tuned, would have to exist for every species or work cross-species, making the task even more difficult [104], [109]–[112].

In the application area of monitoring and surveillance the identification of people across non-overlapping camera views with different viewpoints and lighting conditions is a challenge. In those cases face recognition is often not applicable. Alternatives features for reidentification in surveillance applications can be the coloring and type of the

person's clothing or body parts [113], [114]. The application of surveillance of people is close to the task of monitoring wildlife for conservation reasons. In both cases the data collection is non-invasive and non-cooperative with the individual of interest [115].

Some works go one step further and built end-to-end systems for non-cooperatively taken datasets. In those approaches a detection algorithm was combined and added previous to a reidentification method to isolate the region of interest, for example studies on detection and reidentification used a dataset of persons in the 'wild' including pedestrians or the detection and reidentification of persons in data from photo albums [116], [117].

The above mentioned approaches consume images, contrary a few studies concentrated on the processing of video input data for reidentification tasks for person and vehicles taking advantage of additional information from metadata on time and location [118]–[120]. Same as for image-based tasks, the training of deep learning networks for videos requires a large amount of labeled data. Data labeling in videos is even more time intense than for images. In many cases the single frames must be labeled individually.

6.5.2 Applications on animals

The community of computer vision scientists and conservation ecologists has grown markedly over the last few years. The technical progress over the past decades enables less invasive research methods on animals including the use of camera traps and the identification by natural body marks instead of artificial marks tagged to animals for mark-recapture techniques [121]–[123],[122].

The first application of computer vision in the field of animal re-identification was introduced in 1990. On scanned images of sperm whale flukes (P*hyseter microcephalus*) noticeable characteristics were manually tagged to a database. Similarity was measured by the maximum sum of similarities of the descriptors [123]–[125]. A more independent system automatically extracted the features of unique body marks on fins of different marine mammals [126]. A different approach was the identification on their trailing edge of their fins that were represented as integral curvatures [127]. A rough timeline of computer assisted approaches for animal reidentification until 2017 is illustrated in Figure 6.1. It shows a few milestones in that area, but is far from completeness stating all the work in that field.

The first ever fully automated estimation of a population was performed on African penguins (*Spheniscus demersus*). With penguins the advantage is that they follow certain paths every day on which the cameras can be placed. Sadly, this will not work for every species [123], [128].

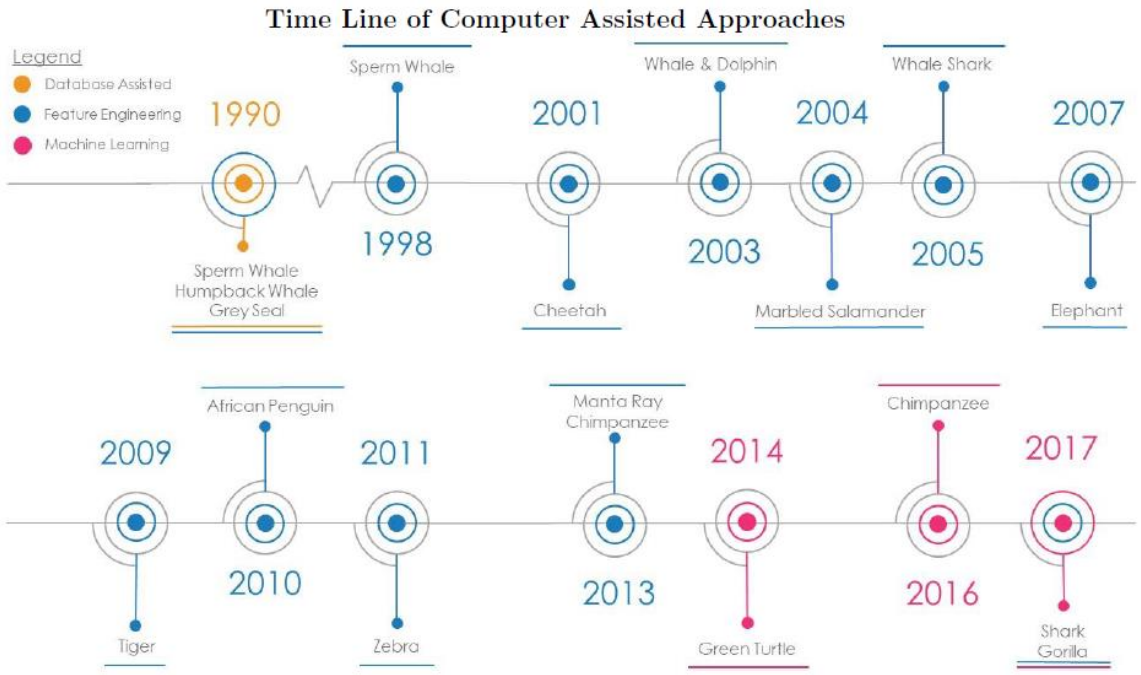


Figure 6.1 **Timeline individual identification on animals**
Timeline of computer assisted approaches on the individual identification of animals [123].

In recent years deep learning methods emerged in the field of animal individual identification of ringed seals [129], whales [130], snow leopards [84], [123], small birds [131] and others. Neural networks can extract features from animals with distinctive body marks. The southern Great Barrier Reef green turtle (Chelonia mydas) population is monitored based on a neural network system that extracts shell patterns [123], [132]. The above described works have all in common that either a labeled dataset was available, the data was collected under non-wild conditions or in a short period of time or the interaction of an expert or a human in general is necessary to make decisions or draw bounding boxes.

While citizen science as described in chapter 6.2 can be a great support for the classification and detection of species, it requires experts to label individuals of a species. Even for experts the identification of individuals in wild environments can be challenging [133], [134].

A group from the Shanghai Jiao Tong University tackled the issue of missing labeled data for animal applications. Together with the WWF they generated a dataset of 92 Amur tigers in wild zoos containing 8072 high-resolution video clips. The dataset is labeled with ground truth information on the tiger's identity, pose keypoints, bounding boxes and the viewpoint on the animal. Just as for leopards the individual identification of Amur tigers is based on their stripe patterns. Furthermore, they tested common re-identification methods that are said to generalize well to new tasks on the new dataset. Compared to the application on vehicles and persons the methods struggled with the tiger dataset, due to the varying poses of tigers and the perspective distortions of the stripe pattern. They added a component that align the distortions of the pattern based on the pose of the animal that outperformed the previously tested models. The downside of this approach is that a very accurate pose estimation is required. In their work they did not conquer the challenge of opposite flanks of tigers. Different sides were

treated as different entities. In the worst case scenario they would double the number of the population, which is significant on a dataset including 92 individuals and in general it is estimated that only around 600 Amur tigers remain in the wild. For such small populations this would affect the population census tremendously [135].

Another approach to cope with the distortion of the stripes was made by modeling a 3-dimensional surface model of the animal. Here manual labeling of certain keypoints in the images are necessary. The aspiration of this research was to have a database of living tigers caught by camera traps and be able to match the tigers with confiscated tiger skins to fight the trade [136].

Non-deep learning researches for animal re-identification are based on the classic analytical computer vision for pattern recognition and matching, including algorithms like WildID or HotSpotter for animals with strong patterns like zebras, jaguars and giraffes [137], [138]. The HotSpotter algorithm described in chapter 5.3 is a fast cross-species algorithm using the SIFT algorithm and nearest neighbor search to identify individuals against a labeled database of known individuals. The application area included several application cases with the biggest one being the 'Great Zebra Count' [68], [137].

The basic approaches for the identification and reidentification of species with unique patterns are similar and for the most part use either CNNs or the SIFT algorithm. Besides mammals and birds with fur or feather patterns applications for other body marked animals exist, too. Manta rays were identified by unique spot patterns on their ventral surface and humpback whales by their pigmentations and scars on the flukes. Inspired by deep learning for face-identification on humans they adapted a face reidentification CNN to learn features for the body marks. In their work they tackled the challenges of unseen individuals and robustness to changing viewpoints. But ultimately, the final decision for matching was always made by an expert manually interacting [139].

For species that lack unique markings other methods were tested. With the same concept as for humans bears, primates, pandas and pigs were identified by facial recognition [140]–[151].

The program BearID detects a face and keypoints in it, reorientates and cropps the face based on the keypoints and classifies it to a known database of 132 individual brown bears (*Ursus arctos*) with a deep CNN ResNet architecture. The database was generated for this research and consists of 4674 images taken by photographers and staff in National Parks in Canada and Alaska with ecotourism where the bears are used to humans being in the vicinity, therefore the images are of a higher quality and better viewpoints than the ones automatically taken from camera traps[152]. Another approach for animal species lacking unique markings was to use spline curve matching techniques to surround elephant individuals in images [153]. But it seems, that this technique was not established in the area of animal individual identification, probably due to the vast amount of poses animals are seen in.

The difficult obstacle with facial recognition are the required datasets. The above mentioned studies had pre-labeled data on the population of interest at hand in mostly optimal conditions. For applications in the wild it is extremely difficult to sort out appropriable data in good quality from camera traps, if any.

7 CONCEPT

The objective of this thesis is to develop a program that automatically identifies and re-identifies individual animals, but does not require the input of previously labeled data, minimizing the manual interaction needed from ecologists. It is aimed to use algorithms that already have proven themselves for the use on wildlife data, but to implement additional components that substitute the otherwise needed manual interaction.

The research in this thesis is based on the leopard dataset from the PanAfrican Programme explained in chapter 2.2. With leopards being classified as vulnerable and a vital part of ecosystems this thesis aims to address the problem of identifying individual animals by automatically processing the data taken from camera traps to give ecologists a good tool to estimate a population census on the leopard species in the area of interest. Individual identification and their territorial behavior are crucial steps to answer questions on how to protect leopards and their habitats.

The system is only applicable for species that fulfil the three assumptions from chapter 2.3:

1. The inspected animal species has a solitary behavior
2. Within one triggered video the same individual is seen throughout the frame sequences
3. The inspected species is uniquely identifiable by coat or body marks

Based on this assumption it is presumed that one video clip, triggered by a motion, contains the same individual in all frames. Except the animal moves out of the area captured by the camera leaving part of the video empty during the one minute of recording.

The other challenge that has to be kept in mind is the incomparableness of the opposite flanks. It has to be prevented that individuals are counted twice into the population.

To tackle the challenges and requirements the system is built up of several components explained in more detail in the upcoming subchapters. For a more robust system interim steps are planned to cope with the versatility of the data caused by the animal's natural habitat and behavior.

Due to the independent flanks of a leopard it is important to have a mechanism that prevents the comparison of opposite sides. The planned steps to overcome that issue is to sort the frames of right flanks and left flanks into different databases, only comparing flanks within one database at a later point.

7.1 Object detection – The MegaDetector

The first step is intended to find out, if a leopard is shown in the frame and where it is located. The frames are extracted from the video and are treated as single frames. Starting the process, the frames of the videos are fed into the MegaDetector. The MegaDetector returns the likelihood with which it detects an object and the location of it via a rectangular box that surrounds the animal, if an animal is found. The area inside the rectangular is from now on the region of interest, in which the animal is presumed, cutting off most of the background for that frame (see Figure 7.1 top). The region of interest is called an annotation and is surrounded by the rectangular bounding box.

It is assumed that there is no animal in the frame, if the algorithm does not detect an object with the likelihood exceeding a preselected threshold. Empty frames are not further analyzed in the process.

For the rare case mentioned in chapter 2.2 that more than one individual at the same time is seen in the image, the MegaDetector will return two rectangular boxes. The video the frame was derived from can be treated separately. In rare cases that show two individuals within one video, ecologists probably want to look at the videos manually anyhow, because much lessons can be learned on their behavior when interacting with conspecifics or other living beings.

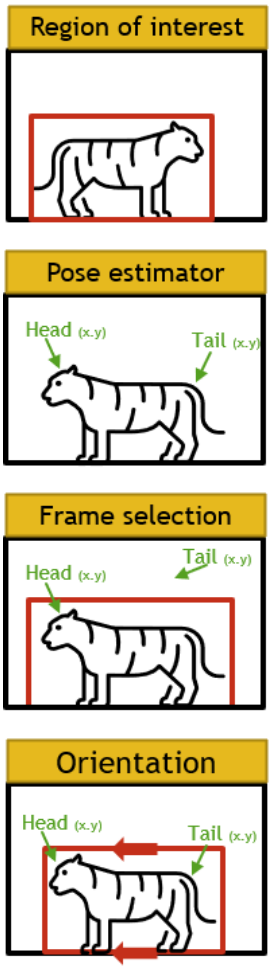


Figure 7.1
Preparation for feature detection

7.2 Pose estimator – DeepLabCut

The raw video is passed to the DeepLabCut pose estimator. Depending on the species a set of body parts is defined on which the algorithm was trained, like: right front knee, left back foot, head, tail and others. When a new unseen video is processed the algorithm returns a prediction for all the body parts of the set with the predicted location as (x,y) coordinates and the probability. For simplicity only two keypoints are shown in the pictogram (Figure 7.1).

7.3 Frame selection

The frame selection component acts as a backup solution and indirectly verifies the body part predictions with the result of the MegaDetector. In the rain forest the background can be noisy. If the pose estimator predicts a body part outside of the annotation, it is assumed to be a wrong estimation (see tail prediction in Figure 7.1). The prediction for this specific body part is dropped for the further process.

7.4 Side predictor

The newly acquired knowledge from the pose estimator on body part predictions feeds the side predictor. The side predictor is a pre-trained classification algorithm using a neural network. It gets the predicted (x,y)-coordinates and the likelihood from the DeepLabCut body part estimations for each frame as an input and predicts the viewpoint (left, right) on the animal in that frame. The class left means that the animal's left flank is visible.

The known viewpoint will later on help to prevent that two flanks of opposite side will be compared. Based on the side prediction the frames are split into different databases.

Furthermore, the information given by the side prediction is later needed for the HotSpotter to define the orientation of the animal in the frame, stating the directions of the head and tail of the animal (see Figure 7.1), which later on makes the feature detection and matching algorithm more robust.

The above mentioned side predictor is the only section in the process for which labeled data will be necessary. It will be trained on a cheetah dataset. In the course of this thesis, the only required labeling task is to sort the cheetah dataset into the classes: *'right side*' and *'left side*' (see chapter 8.2.5.1**Fehler! Verweisquelle konnte nicht gefunden werden.**). The side predictor is believed to be suitable for other animals with similar body proportions than cheetahs and can later on be used cross-species for the leopard dataset. The concept is assumed to be applicable across different species that meet the above mentioned requirements. If the underlying data quality allows, it can be applied to all solitary *Felidae* with distinctive pelage without further manual labeling.

7.5 The Databases

Once the viewpoint is identified the data is split into two databases. One contains leopards shown with their left sides and one with their right sides visible.

Initially, each frame of one video is tagged with the same name ID, assuming it is the same animal. Going back to the example from the definition of the research issue in chapter 2.4 (see Figure 7.2) each non-empty frame from 'Video 1' gets the name ID 'Leo 1' analogue for 'Video 2' and 'Video 3'. All leopards visible from the right side and therewith showing the feature marked as a red star go into the database for the right flanks named 'DB Right'. Analogue the left flanks with the blue star feature in 'DB Left' (see Figure 7.3). It is important that during the process the information on name IDs and based on that the video ID from which the frame was extracted does not get lost, here marked as 'Video 1', Video 2' and 'Video 3' in Figure 7.2. From this point on the two databases are treated independently.

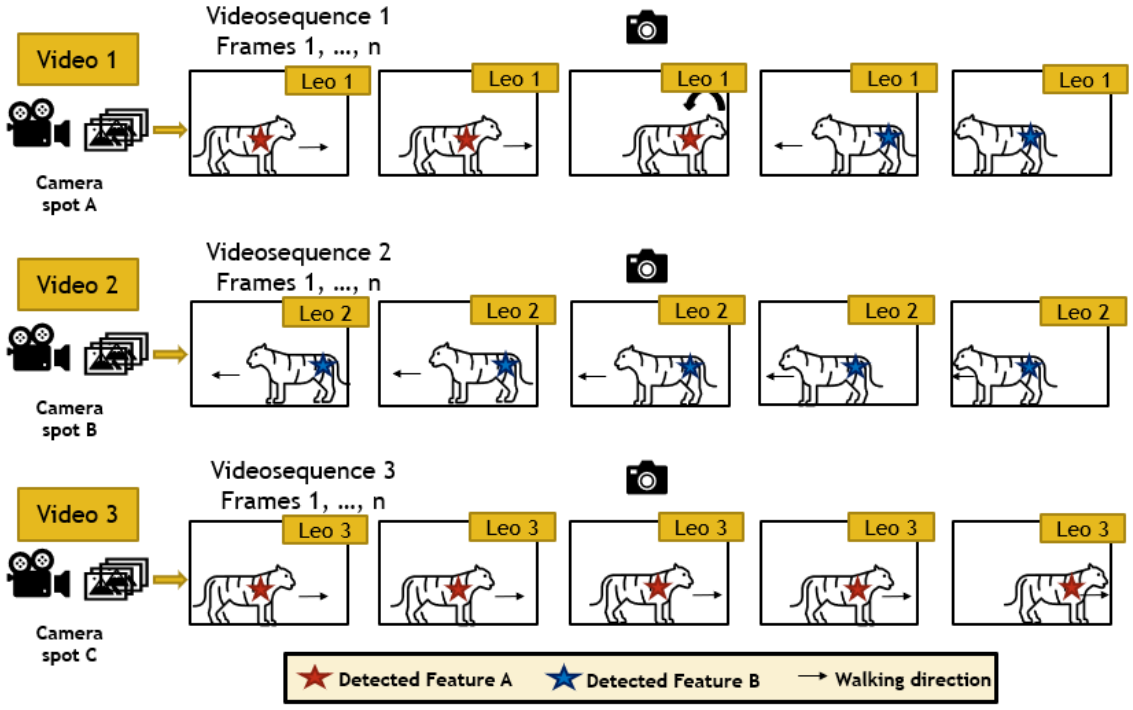


Figure 7.2 **Automatic labeling of the frames**
Each frame gets automatically labeled with a name ID related to the video it was extracted from.

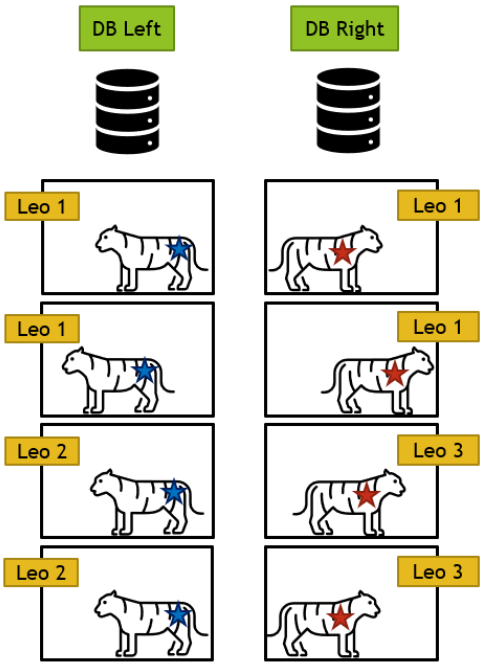


Figure 7.3 **Database split**
The frames are split into the databases based on the viewpoint on the animal.

7.6 Feature detection and matching – IBEIS with HotSpotter

The HotSpotter program seems to be a suitable component for the pipeline, not having to reinvent the wheel and build on a program that has proven itself for a use case with zebras. Together with the other components added to the pipeline and the core assumptions for solitary species the issue of manual interaction can be overcome.

The feature detection and matching process is performed on both databases individually. The IBEIS program, with the HotSpotter as its heart, gets for each frame the name ID and the related bounding box from the MegaDetector. Besides the bounding box it requires the orientation of the animal in each frame. Usually the bounding box and the orientation are manually assigned in the GUI by the user. The manual interaction for the bounding box is replaced by the MegaDetector's output. The labeling of the orientation is theoretically already circumvented by the database split. For the database containing the left flanks the animal is oriented facing left. The head will be most likely located farther left than the tail and vice versa for the right flanks. With this approach it would only pose a problem, if the animal is seen upside down in the image. For leopards or *Felidae* in general this is an unusual scenario. Most of the data taken by camera traps capture animals walking by. A possible, but rare scenario, could be a leopard rolling on its back right in front of the camera. In this way the head would face left, but the right flank would be visible. No problems are expected to originate from such rare scenarios. Nevertheless, if it appears to become a problem an additional neural network could be trained similar to the side predictor on the classes stating the orientation left or right, with a special respect to those rare scenarios.

As soon as the algorithm got all the required input the spots and rosettes of a leopard will be analyzed by a feature detector. An annotation can then be queried against other annotations in the database. The algorithm calculates a score between the queried annotation and each annotation in the database, based on their feature descriptors.

For each name ID in the database, meaning each known individual, the score from its annotation reaching the highest score is taken. The top ranked name ID score states the best matching individual of the database to the queried image. If the score exceeds a previously specified threshold it is a match and the name IDs are merged, meaning that all annotation of the name ID the queried annotation originated from are believed to show the same known individual as the annotations belonging to the name ID that had the highest ranked score. For a simpler explanation the nearest neighbor indexing as described in chapter 5.3 is ignored in this concept.

7.7 Merging of databases

Previously the frames of the videos were divided up into two databases, 'DB Left' and 'DB Right'. After the feature detection and matching within one database is completed the databases can be merged back together. The information for each frame from which video ID it was extracted was carried along during the process. The databases are merged back together on the video IDs. During this process additional matches can be found.

For the example here, the assumption is that in '*DB Left*' *'Leo 1'* and *'Leo 2'* were matched and in '*DB Right*' *'Leo 1*' and '*Leo 3*' were independently matched. From the initial labeling of the frames belonging to the same video it is beneficial that *'Leo 1'* is represented in both databases, because within one video both flanks were visible. Now

knowing from '*DB Left*' that '*Leo 1*' and '*Leo 2*' are the same individual, as well as *'Leo 1'* and '*Leo 3*' from '*DB Right*' and obviously the frames from *'Leo 1'* in both databases show the same individual it can be concluded that all three videos show the same leopard. Without '*Video 1*' the other two videos are incomparable.

In cases where one individual is only seen from the same side all the time, without any information on the front or back or no change in direction from the leopard's walking path, those detections from the individual sides are incomparable. But this limitation counts for manual matching by experts, too. A natural fact that cannot be overcome and heavily depends on the given data.

During the development of the concept I considered other techniques besides the ones mentioned above. The first method that usually comes to mind for reidentification tasks are convolutional neural networks (CNN). To train a neural network a large labeled training dataset is required, meaning that some kind of manual labeling is required, which infringes the demand of minimal manual interaction of the research issue. Furthermore, this network would be trained on the specific individuals from the underlying data. For the utilization in another wildlife reservoir or even cross-species additional training would be necessary. Through transfer-learning this might be limitable, but no out of the box solution for other application areas is possible. One of the initial thoughts on an approach to still be able to use neural networks and to generate the training data automatically was based on the same core assumption. For solitary species the single frames of a video could be extracted and all labeled with the same name, assuming only one individual is seen in a motion-triggered video. Those frames could have been used to train a convolutional neural network and later on predict the matching individual for animals in single frames. But due to the setup of the cameras on fixed locations this method seems to be prone for overfitting.

8 IMPLEMENTATION

This chapter delineates the implementation of the concept with its individual components in detail and presents the used hardware. The concept outlines a pipeline in which the data can be entered without previous labeling or other manual interaction. The core piece for the matching is the IBEIS feature detector. Additional components are added to the pipeline to substitute the otherwise needed interaction of a user. The results are stored and further processed in a SQL database.

All code can be viewed in the github repository [154].

8.1 Hardware Setup

This chapter lists the hardware that is used to process the data through the pipeline as a reference. The computer consists of a AMD Ryzen 5 1600 processor with a clock rate of 3.2 GHz, 32 GB RAM and a NVIDIA GeForce GTX 1660 SUPER graphic card. The main operating system is Windows 10 Pro. Additionally a virtual Linux environment was used, because not all of the pipeline's components are Windows compatible. The virtual Linux environment is Ubuntu 18.04.5.

The pivotal component is the graphic card. Some of the components are optimized for calculations on GPU, for example the MegaDetector and DeepLabCut. The availability of a GPU is not an exclusion criteria, but the calculation on CPU can take up to 10-100 times longer for the concerning components. To prepare the system for a GPU included calculation tensorflow-gpu for Python is required. Python has an interface for GPU calculations exclusively for graphic cards from NVIDIA. Additional software requirements have to be installed, for installation support please see https://www.tensorflow.org/install/gpu.

During the initial planning options like Google Colab or Azure were considered as a backup solution, if the needed computing power exceeds the one of a normal desktop computer in an acceptable runtime. With those solutions the calculations are done on cloud servers from Google or Microsoft, including GPUs and even TPUs (Tensor Processing Units). This approach requires a stable internet connection. For the leopard application case the data was entirely computed on the computer with the components listed above. For larger datasets cloud solutions should be considered.

8.2 Pipeline

The aim of the program is to identify and reidentify the leopards in the dataset from the PanAfrican Programme to estimate their population and get more insights on the behavior and territories of individuals. The pipeline consists of the following components which are explained more detailed in the following subchapters. As stated in the concepts chapter the data flows with only little formatting preparation into a pose estimator. In parallel, the images of the videos are processed by a detector which

returns the region of interest. An additional step, in which the detector verifies the post estimator's result, is inserted to filter some errors early. The remaining frames run through the side predictor and continue with the new won side information to the identification component. This chapter goes through the algorithms and programs used in the pipeline and the amendments made to the concept that must have been done due to technical and performance issues.

The pipeline includes algorithms that already have proven themselves in application cases for the use on wildlife data as well as newly trained ones to substitute the otherwise needed manual interaction.

8.2.1 Data preparation

The video clips in the leopard dataset from the PanAfrican Programme have a duration of one minute. Every time a camera is triggered a video of one minute is stored. The data contains only videos with leopards. I received the raw data and a textfile with the tags for each video. They are sorted in a folder structure as shown below and explained in detail in chapter 3:

Loango\Loa_d12\d12_Loa_cam38_012345_012345_20170912\09180056 .mp4

Sitename\grid cell\camera in that cell_ geodata in UTM_maintenance date of camera\videoname.mp4

Before starting the analysis every video was automatically renamed by a Python script to a string including all information on sitename, cell, camera spot, geodata and the time the data was collected from the storage.

Videonaming convention: d12_Loa_cam38_012345_012345_20170912__09180056 .mp4

This guarantees that all the information from metadata is carried along the whole pipeline as well as for saved files in interim steps. The uniform naming convention makes the program more robust against error when opening files.

From now until the import to the database every step is repeated for each video. The video is split into frames that are separately saved as jpeg files in a folder named accordingly, for the upcoming components of the pipeline. A frame rate k can be chosen and accordingly every k-th frame of the video is saved and further used. In the work of this thesis I chose a frame rate of $k = 20$ in favor of computing time. In the randomly spot watched videos the leopards moved slowly and did not run, therefore the animal's movement and change in position between the frames are rather small and a larger frame rate seems reasonable. A less random selection might be possible in conjunction with the detector and the pose estimator, selecting those frames with a certain change in position. But this approach does either not reduce the computing time, because the detector needs all the single frames beforehand or forfeits in accuracy, because the security rooting from the frame selection as a combination of the pose estimator and detector gets lost.

8.2.2 DeepLabCut

The task of the pose estimator component is to deliver information from which the orientation of the seen animal can be derived. For the pose estimator I decided to use the DeepLabCut program, because the code is publicly accessible and very well documented. The main reason that I chose DeepLabCut is the pre-trained network on the AcinoSet in its Model Zoo. The AcinoSet is a dataset on cheetahs in the wild from different viewpoints [52], [53], [59], [65], [155]. Although cheetahs belong to the subfamily of small cats (*Felinae*) and leopards to the big cats (*Pantherinae*) they share similar body proportions, independent from their also similar, but distinctive coat pattern. Figure 8.1 shows a cheetah (left) and a leopard (right).

Figure 8.1 **Cheetah and leopard**
Cheetahs and leopards have similar body proportions and have both patterned coats. Cheetahs have mostly spots and leopard rosettes. For both species individual are uniquely identifiable by their coat pattern [12], [97].

The network of weights I use was trained on labeled cheetah frames with the keypoints as labeled in Figure 8.2.

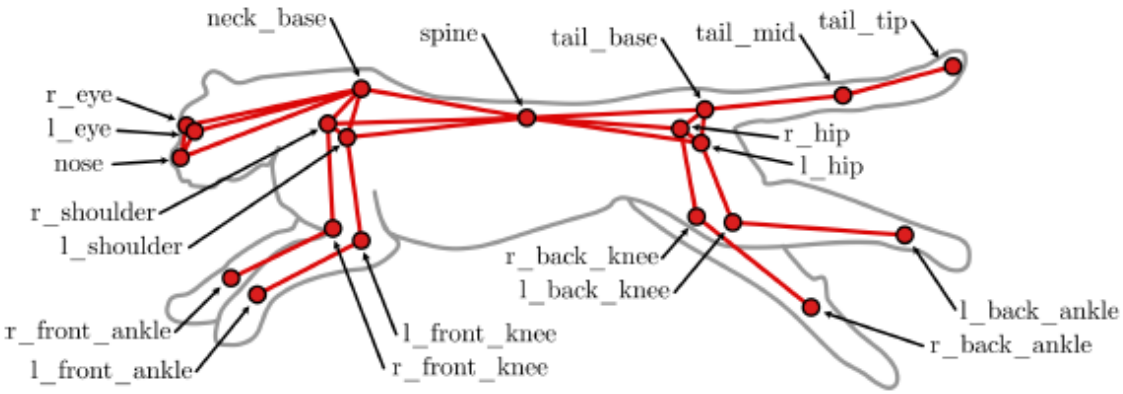


Figure 8.2 **DeepLabCut keypoints for the cheetah model**
DeepLabCut's cheetah model from the model was trained on 20 prior defined body part keypoints [65].

The developer of the DeepLabCut program developed a GUI with the intention that a user can label and train an own project from the bottom up. Having said that the aim of this thesis is to reduce manual interaction as much as possible, I will import the weights from DeepLabCut's readily trained cheetah model from the ModelZoo in the leopard memory pipeline. The labeling and training process can therefore be skipped and I solely

embedded the prediction function in the leopard memory pipeline. The function gets the raw videos as an input and returns a prediction for each frame and body part with a location in (x,y)-coordinates and the likelihood of the presence of the body part in that point. An example output from the pose estimator is shown in the Appendix 0.

DeepLabCut offers a function to plot the predicted keypoints on the original video, which is a beneficial tool for manual checking. In favor of runtime this function is not implemented in the pipeline, but can be individually called for single videos. The results are exported as csv files maintaining the naming convention from chapter 8.2.1.

In the DeepLabCut workshop it is mentioned that different basic networks for the transfer learning can lead to different results. A higher robustness is reached with ResNet50 or ResNet101, while MobileNetsV2 is faster. More accuracy and runtime affecting settings can be chosen to train the CNN in terms of hyperparameters like: Batchsize, number of epochs among others listed in chapter 4.3. As I use the pretrained cheetah network the architecture of the convolutional neural network and the training parameters were already set and the training completed. [60]

8.2.3 MegaDetector

In the next step an object detector localizes the leopard in each frame. For this component I chose Microsoft's MegaDetector, because it has proven itself for many applications and is said to adapt well to new environments. Figure 8.3 pictures the results of the MegaDetector when applied to images from the leopard dataset. It can be examined that the MegaDetector works well on images with noisy backgrounds and bad lighting conditions suitable for automatically captured records in the rain forest. The pictures show two frames from the same video of a leopard moving away from the camera. For the human eye the leopard in the later frame is hard to catch at first glance.

Figure 8.3 **MegaDetector detection**
The images are frames from the same video. On the left the animal is clearly visible on the right it is hard to detect at first sight for a human. Both detections were correctly found by the MegaDetector [12].

Similar as for DeepLabCut, the MegaDetector's detection function, that returns the coordinates for the bounding boxes around the predicted region of interest, is embedded into the pipeline. The coordinates of the bounding boxes and its height and width are exported as csv files maintaining the naming convention from chapter 8.2.1.

8.2.4 Frame Selection

The intermediary frame selection filters obvious major errors before the data is transferred to the next component of the pipeline. The results from the pose estimation and the detection are handled with basic dataframe functions from the Python module pandas.

Only the predicted body parts that are within the detection area of the detector are kept. Every body part with x or y coordinates outside of the interval given by the minimum and maximum values of the x and y coordinates from the bounding box are marked as outliers. For the further process the likelihood from the pose estimator is set to zero for that body part.

8.2.5 Side Predictor

With regard to the viewpoint problem on coat patterned animals and their independent flanks it is necessary to define which side of the individual is visible to prevent the comparison of opposite flanks. Therefore, one component of this pipeline is a side predictor.

The side predictor is basically a classifier sorting the images into viewpoint classes, i.e. ‘right’ and ‘left’ which will be explained more in detail later in this chapter. To train a side predictor labeled data is necessary. The aim is that the side predictor will be trained on another dataset and subsequently can be used for prediction on the leopard dataset. In this way the side predictor and therewith the pipeline stays generic and not specialized on the leopard dataset.

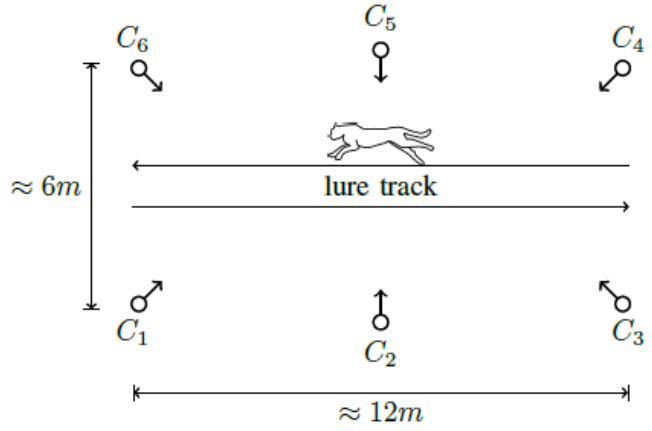


Figure 8.4 **Camera Setup for the AcinoSet data collection**
6 cameras from different viewpoints on the cheetah capture footage. The cheetah is lured with a mock prey through the middle [65]

As a labeled data fundamental I used the AcinoSet to train on additional labels to the existing labels. The existing labels are used as an input. The set consists of videos of Cheetahs in a wild-like setup. The setup included 6 cameras marked as $C_1, \ldots, C_6$ in Figure 8.4. The cheetahs were lured with a mock prey to run a certain route. In the dataset the cheetahs run either in a straight line through the middle or are encouraged to make

sudden turns by changing the track of the mock prey. During the sudden turns the cheetahs are seen in more various poses than when running a straight line (see Figure 8.5), which generates more diverse input data, which generally makes deep learning systems more robust.

Figure 8.5 **Poses of cheetahs in action**
Cheetahs are lured with a mock prey to make sudden turns to make the dataset more versatile.

The Acino dataset was initially labeled with 20 body part keypoints by a research team exploring DeepLabCut for 3D pose estimation (see chapter 5.2). The thought of taking the already labeled Acino dataset was to keep all options open for the later research. It will be possible to use the side labels and images to train a convolutional neural network or use the the body keypoints and viewpoint labels to train a classic neural network without image data.

8.2.5.1 Labeling of training data with GUI

In the work of this thesis 2824 frames of the AcinoSet with 6 different cheetahs were sorted leading to 8 classes. To speed up the labeling process

To speed up and make the monotonous labeling process more pleasant I developed a simple graphical user interface as shown in Figure 8.6. For each frame the user has to make 2 choices on the viewpoint. One concerning the sides and the other the front and back. Due to the fact that living beings do not appear as rigid objects instead they have an extremely complex locomotor systems, no clear line for the definition on the right and left viewpoint can be drawn. In this process the viewpoint is considered as ‘right’, even if the front or back of the leopard is seen, but right flank as well or at least partly, analogue for left. Excluding choices are back and front and analogue left and right. Meaning the user makes the choice for the side on ‘right, ‘left’ or ‘none’ and the second choice between ‘front’, ‘back’ and ‘none’. In this way combined classes like ‘left’ and ‘front’ are possible, if the leopard’s chest or face and part of the left flank are visible. Combinations like ‘front’ and ‘back’ are not possible. The completion of the possible combinations lead to the following 8 classes in Table 8.1. The labels are exported as a csv file.

Figure 8.6 **GUI interface for side labeling**
The user can label the shown cheetah. A combination of the tags is possible, but opposite flanks cannot be chosen and front and back are excluding choices.

The cheetahs shown in Figure 8.5 would be classified as '*front-left*' for the left image and '*front-right*' for the image on the right.

8.2.5.2 Training the Side Predictor

Models of different architectures were trained for the side predictor on the side-labeled AcinoSet. The first attempt was a classic neural network without convolutional layers. The network got the ground truth data for the body part keypoints of the cheetahs as variables for the training. As target variable the side predictor is fed with the manually labeled viewpoints on the cheetahs. The 8 classes as described in the preceding chapter are encoded as stated in Table 8.1.

The second implemented architecture is a CNN trained with the frames from the AcinoSet. To aim for a better performance a transfer learning approach was applied using ResNet50 as a basis, as well as a MobileNetV2. In different attempts exclusively the last flattened layer was retrained or the last few layers were retrained freezing the remaining weights of the net. Assuming that 8 classes could be too many the 8 classes were combined to only 3 classes, in which only left and right are excluding factors (Table 8.1 right column).

Viewpoint	Encoding Type 1	Encoding Type 2
Front	0	0
Front left	1	1
Front right	2	2
Back	3	0
Back left	4	1
Back right	5	2
Left	6	1
Right	7	2
Unknown	0	0

Table 8.1 **Encoding of viewpoint classes for side predictor**
The previous labeling process results in 8 combinations encoded from 0 to 7. Type 2 encodes the viewpoints in 3 classes based on the visibility of the flanks.

The images of the raw frames of the AcinoSet have the cheetah in a small area with a lot of background around. For comparison the MegaDetector was used on the AcinoSet and solely the cut out bounding boxes with the cheetah inside were fed to the CNN.

A few videos of the leopard dataset were labeled on Zooniverse based on the viewpoint with the tags *'frontside', 'backside', leftside',*and *'rightside'*. The tags are based on the video data and often do not apply for all frames, if the leopard moves. Nevertheless, a ResNet50 CNN architecture was trained. After training the networks are used to make predictions on the remaining frames after the frame selection. The prediction is appended to the database for each frame.

8.2.6 IBEIS and HotSpotter

For the conventional use of the HotSpotter software the programme expects to get the frames as input and thereafter a manual human interaction is required. The user must draw a bounding box, the region of interest, on each image with always the same orientation. Meaning always drawing the box from the head to the tail of the animal. The target of this thesis is to eliminate the manual interaction. Therefore, the drawing of the bounding box, needed by the HotSpotter, is automatized by harnessing the information won from the DeepLabCut algorithm and the side predictor.

The out of the box version from the HotSpotter software recommended by an ecologist and mentioned in chapter 5.3 with a Windows installer turned out to be difficult to use other than manually with its GUI, due to inaccessible code the system workes as a blackbox. Therefore it was replaced by the IBEIS program (Image Based Ecological Information System), the successor of HotSpotter, but still based on the main HotSpotter algorithm. The new improved version comes as a Python package and can be installed via pip. Unfortunately a major drawback is the fact that the Python package as a whole is exclusively available for Linux environments (for installation instructions see: [71]).

Having the agony of choice it was decided to proceed with the Linux variant with the freely available code [71]. The software explained in this chapter is applied on a virtual Linux machine with Ubuntu 18.04.5. Technically the whole pipeline can be run entirely

on the Linux virtual environment. Due to slower performance in the virtual environment compared to the host operating system the components of the pipeline are split. All that is required in both environments for the pipeline is the folder with the extracted frames and the database from IBEIS that I automatically fill.

I did not use the IBEIS GUI, because it requires manual interaction of a human. I only used the relevant functions from the module that trigger the query requests. I amended the code at several points to make it suitable for a use without the GUI.

The IBEIS software offers the option to group images into occurrences. In this way the order in which the images are queried against each other can be prioritized. An occurrence originally describes records taken in a small time frame and in a near location. I will use the occurrence grouping in the database to group all data from one study site, for example the site Loango, independently of the time the image was captured.

8.2.6.1 Database

For an easier handling of the data, I decided to have all necessary data in an SQL database to avoid working with several csv files for every video. I did not build the database from scratch, instead I use the database that is automatically generated when starting a project in IBEIS. In any case an interaction with this IBEIS database is necessary to import my own data into the IBEIS program. To have all data united in one database I continue to use the IBEIS database for the subsequent steps as well.

The information won in the previous steps of the pipeline are automatically written into the database. Information that is otherwise entered manually by the user. Import information include the initial names, the bounding boxes, the viewpoint, locations and of course the IDs and paths for the import of the images. The relevant tables are *'annotations', 'images', 'names', 'imagesets'* and *'imageset_image_relationship'*. I added the tables *'queries', 'clusters'* and *'cluster_edges'*. The tables that are newly created will be later explained at the relevant point in the pipeline.

The *'images'* table lists all frames with an image ID, the path of the image, the file type, the height and the width and optional GPS data. For the course of this thesis the GPS data is not used due to the secrecy of sensitive data on the location of an endangered species. The *'imageset'* and *'imageset_image_relationship'* table are filled with the information on the study site. Each study site gets an imageset ID in the '*imageset*' table and its name as a text column. The '*imageset_image_relationship*' links each image ID to an imageset. The '*names*' table is originally foreseen to hold the manually entered names of an individual found in the matching process. In my implementation the '*names*' table holds the names of the videos. Initially all frames from one video get the same name ID from the '*names*' table. In such manner already several images of one individual are grouped together, based on the assumption made in chapter 2.3. The most vital table for the import is the '*annotations*' table. An annotation is the region of interest in which the animal is located in the image. The table comprises information on

the location of the animal, the viewpoint and the linked name ID to name the most important columns. The location of the annotation in the image is derived from the bounding box prediction from the MegaDetector. The viewpoint is forwarded from the side predictor and the name ID is initially set as stated above. I discarded the originally plan from the concept (see chapter 7.5) of splitting the data into different databases based on the view point. Instead a column for the viewpoint is added to the *annotations* table and filled. Furthermore I chose to conduct the matching process purely based on the scores and other information written in the database and not use the implemented functions from the IBEIS software. To simply rely on filter conditions and sorting of scores makes the process and decisions more transparent. The restriction on comparing opposite flanks can be handled with simple filtering as well. The process and conditions for matching will be elucidated in a later chapter 8.2.7.

8.2.6.2 Queries

The target is to label each annotation with a name that uniquely identifies the individual, if possible. In other words, annotations of the same individual are grouped under one name ID. A query runs one annotation against several other annotations and compares the feature vectors calculating a score for each annotation pairing. The score is a measurement for the similarity between annotations aggregated from their feature correspondences.
The algorithm pairs the queried annotation with several annotations from the same name ID. From those pairings the highest score is chosen representing the name ID, with a single name score. The underlying annotation from the name ID leading to that score is the best matching annotation to the queried annotation compared to the other annotations of that name ID. This is done between the queried annotation and all the name IDs with its corresponding annotations in the database.

For a faultless triggering of the queries it must be differentiated between the following IDs: *qaid, qnid, daid, dnid*. The IDs containing *q* are the objects that will be queried against the database IDs, which begin with a *d*. The *n* implies that the ID refers to a name ID, while an ID with a *q* refers to an annotation. One *qnid* groups several *qaids*, but a *qaid* belongs to only one *qnid*. Analog for *dnid* and *daid*. Hence an annotation score measures the similarity between a *qaid* and *daid*, while a name score measures the similarity between the *qaid* and the most similar *daid* within the *dnid* or further aggregated the similarity between a *qnid* and a *dnid*. In a nutshell, a frame (*qaid*) of a video (*qnid*) is compared to the frames (*daids*) of another video (*dnid*). The decision on whether it is a match is made on name ID level and effects all annotations belonging to the name ID. For simplicity already merged videos under a *nid* were not considered in the explanation above.

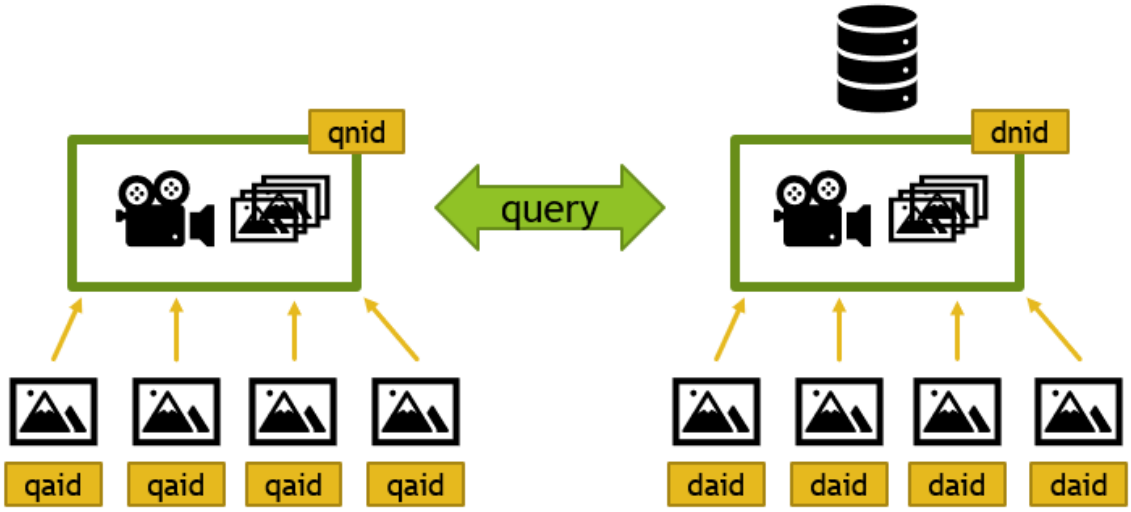


Figure 8.7 **Structure of IDs in database**
All annotation stemming from one video have the same nid (name ID), but each has its individual aid (annotation ID). The images starting with a q are the ones that are queried against the images in the database.

The next step is to decide in which order to query the annotations against each other. The queries are generally run in a way that one frame or several frames of one individual are run against the whole database. In the leopard use case the dataset has additional information on the locations of the camera spots. The locations are spread across Central Africa with large distances. Therefore, camera locations that are close to each other are run against each other first. Although leopards have large territories and can travel long paths and cross country borders it is not expect to see the same leopard several thousands of kilometers away. Making use of the grouping in occurrences as stated in chapter 8.2.6 the queries are run in chunks for each occurrence. An occurrence includes all frames from videos from the same research site. First of all the frames (*qaids*) from the videos (*qnids*) from one site are queried against each other. Within one occurrence it is more likely to find matches, but the leopards are seen across different sites as well. Therefore as a second step the annotations of one occurrence are queried against all other occurrences. The results are written to the queries table in the database including information on *qnid, qaid, dnid, daid*, the annotation score and boolean flags for matches that are automatically considered true, because they were extracted from the same video, as well as flags on whether it is an intra-occurrence query or inter-occurrence query. So far only the scores are calculated and the information collected in the database. No matching process and related amendments in the database are made yet.

The IBEIS software automatically selects a subset of exemplar annotations for each name ID in the database to represent that individual based on the number and distinctiveness of the detected features. Those annotations are indexed in a search data structure based in nearest neighbors to improve the runtime [68]. The mathematical background of the indexing can be looked up in the dissertation of J. Crall. Compared to the query order described above and the automatic selection and indexing of

representative annotations of the software, when systematically running each annotation ID against every annotation ID the computing time grew by almost a factor of 100.

8.2.7 Clustering

After the data has successfully passed through the pipeline the database facilitates the analysis of the results based on conditions and sorting via SQL queries.

The goal of the clustering step is to group the annotations that are believed to be of the same individual. No merging in the names table as primary planned will be done, because if mismatches are made and names are merged they cannot be tracked back. Instead a cluster approach that can be visualized as graphs is implemented. The nodes represent the video IDs from the *names* table in the database. Two nodes are connected with an edge if they were matched. In the illustration in Figure 8.8 each cluster represents one individual. The red sub graph states that the animal seen in the videos with the IDs 3, 4, 6 and 7 is probably the same in those videos. The same accounts for the blue subgraph. For Video ID number 9, printed in green, no match has been found. The clusters can be given real names in the *clusters* table. The width of the edges are derived from the score of the matching. A broad edge implies a high similarity between the animal shown in the frames of the videos. The distance of the edges or the distance in between the nodes and clusters do not give information on their similarity, the length is chosen to serve for a clear visual illustration.

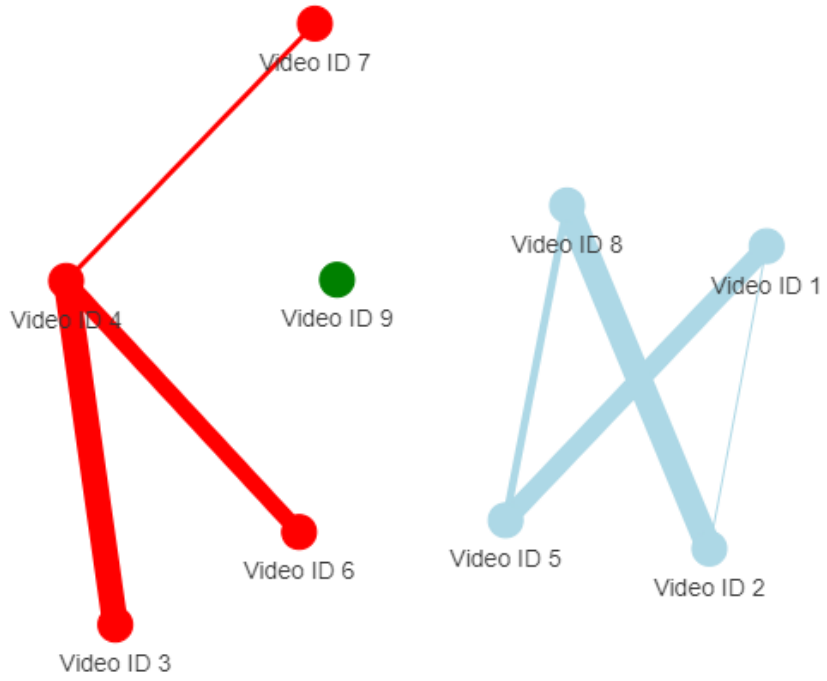


Figure 8.8 **Clustering scheme**
The video IDs are grouped in clusters representing individuals. Edges indicate a match. The width of the edge relates to the matching score value.

The clustering step begins with the extraction of the queries from the database. From the queries table all *qnid* and *dnid* pairings and their name score, the highest score over all annotation scores, is selected. Queries that do not exceed the pre-defined threshold for the score are dropped from the selection. Based on a Boolean column true matches are filtered out. Annotations that are from the same video are marked as true matches, because the same individual is expected to be seen based on the assumption made in chapter 2.3. From here on all *qnids* and for each *qnid* the *dnids* that meet the threshold requirement are looped. For each pairing of the name IDs three cases are possible:

1. Both name IDs do not belong to a cluster yet.
2. One name ID is already part of a cluster, but the other one is not.
3. Both name IDs belong to a cluster, but to different clusters, causing a conflict.

Hereinafter I will explain how the different cases are handled. For case one a new cluster record is added to the *clusters* table and the both name IDs are allocated to the new cluster in the *cluster_edges* table and an edge between them with their matching score is added. A record of the *cluster_edges* table includes information on the name IDs in the pairing, the score, the cluster affiliation and a validity flag, which will become relevant in the third case. In the second case the name ID that does not belong to a cluster and therefore does not have a link to any other nodes gets attached via an edge to the other name ID and its cluster. The third case is the most complex case due to conflicts. A conflict occurs, if both name IDs are already assigned to a cluster, but different cluster. Now the scores are decisive for the matching. If the score of the new pairing is lower than the scores that cause the link of each name ID to another node and therefor to a cluster, nothing happens. Both previously made matches are based on a higher score and hence more similar than the similarity of the inspected pairing. If the new binding score between the name IDs of the pairing is higher than both scores that attach the name IDs to other nodes, the name ID of the pairing with the lower binding score to its cluster gets detached from that node and therewith from the cluster. If the name ID has multiple edges in that cluster the highest one is used for the comparison. In case of a detachment all edges to the old clusters are terminated and a new edge to the node belonging to the other cluster is added to the database and the cluster ID is updated for the name ID. The last option for case three is that the new binding score of the pairing is higher than the connection score of one of the name IDs to other edges in its cluster, but lower than the connecting score of the other name ID. The same procedure as just mentioned is applied. The name ID with the weaker binding to its cluster and edges switches to the cluster of the other name ID. The newly added edge to the name ID and the cluster is higher than for the previous match. The individual in the frames of the video (name ID = video ID) is more similar to the frames included in the new cluster.

With this approach errors can be detected, if a better match is found during the process. Also the addition of new data to the database can be handled by running the new videos,

also meaning new name IDs, against the existing ones in the clusters. For a more intuitive use the clusters can be visualized in an interactive html graph.

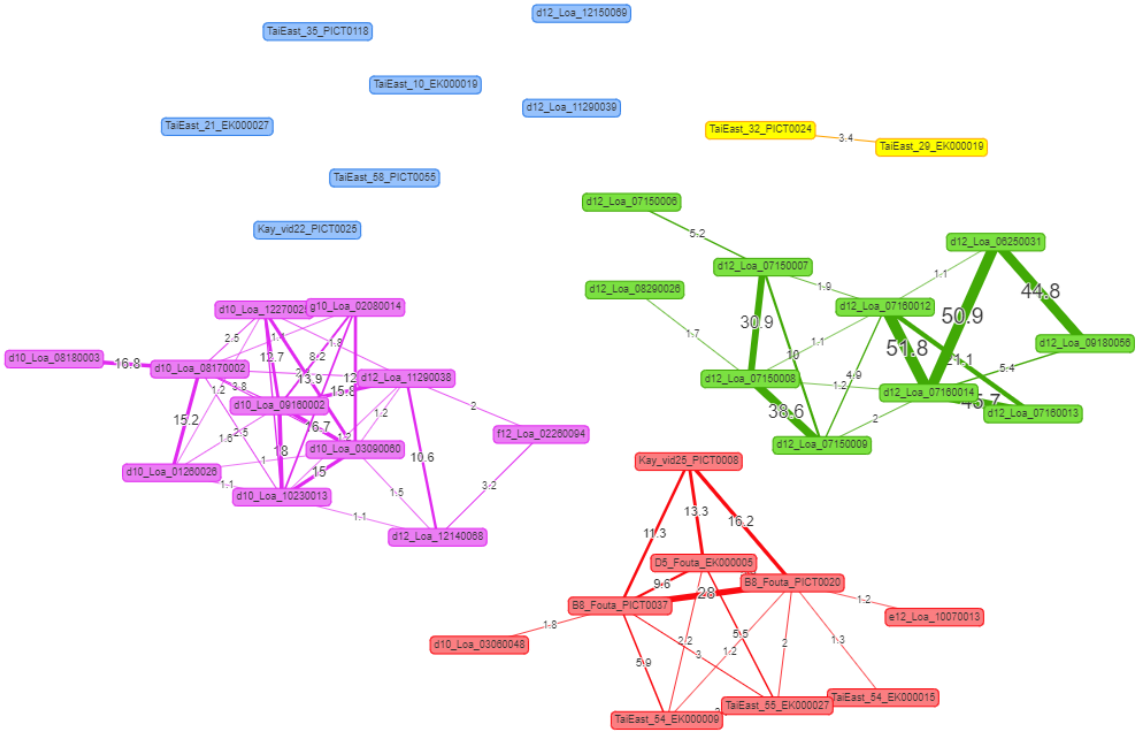


Figure 8.9 **Interactive html visualization of the clusters**
The nodes represent videos. An edge indicates a match grouping all videos assumed to belong to the same individual into a cluster.

8.3 Problems during the implementation

During the development of the leopard memory pipeline some technical issues occurred and components had to be implemented differently than planned in the concept. The problems and amendments are presented below.

8.3.1 Technical issues and modifications

The pipeline can only be executed efficiently on a conventional computer, if a GPU is present. Otherwise the calculations cannot be performed in an acceptable runtime. The used Python modules support the outsourcing of the calculations from the CPU to the GPU, but in general Python's tensorflow is exclusively compatible with GPUs from the NVIDIA brand [156]. Users with other graphic cards than NVIDIA must have this in mind.

It took some effort to align the components and the Python packages they rely on, because the different components are limited to different versions of the underlying modules, especially because of the use of the GPU. A functional combination is listed in Appendix 12.1. Other combinations may work as well, but were not tested.

Another technical issue was the operating system. As stated in chapter 8.2.6 the feature detection and matching of the IBEIS Python module were embedded in the pipeline, instead of the HotSpotter software. The developer recommends a virtual machine for

Linux due to the fact that the Windows issue could not be solved in the past years. I tried to independently install the required packages, but due to errors with the binary dependencies the attempt failed. An easier to use approach would have been to conduct the entire pipeline on the virtual machine in Linux, but the execution of the code in the virtual machine is much slower. Therefore only the feature detection and matching part takes place on the virtual machine. A further obstacle and reason against having the whole program on a virtual machine was the storage that is needed for the videos and the extracted frames. The whole project with video data and extracted frames, Python modules, models, additional data other than the leopard dataset for the training of the CNNs blocked around 160 GB, not including any backup copies and caching.

8.3.2 Adjustments to the concept

Over the course of implementation I had to make adjustments to the originally planned concept. The main amendments concern the side prediction and the related databases and the merging process.

As already mentioned in the implementation chapter 8.2.6.1 I deviated from the idea of splitting the data into different databases and instead combine all data in one database and append a column with the viewpoint to the annotations, which can be filtered by conditional SQL queries to prevent that opposite flanks are being compared. The annotation IDs that are fed into the IBEIS feature detector can be filtered on the viewpoint column. During the training of the side predictor it became clear that due to low accuracies of less than 50% that are reached, the separation of the viewpoints will do more harm than good to the overall results. The results of the side predictor and the associated consequences are presented more in detail in chapter 9.1. As an alternate solution to avoid the effect of the inaccurate side prediction on the overall result the side predictor component will be omitted from the pipeline. The body part and pose estimator that produced the input for the side predictor is correspondingly no longer needed. The effect of the loss of the two components is analyzed in the next chapter. A small consolation caused by the loss is the improved runtime, because the pose estimator is computationally expensive.

Another amendment was made to how matches from the feature matching are treated. The original idea was to merge the name IDs of a match. In early analysis it became apparent that the merging process is not fault-tolerant. If an error occurs it is not revertible. For the application case this means that if two individuals are mistakenly merged and a third annotation is found that matches one of the two all three are merged under the same name ID. It is not taken into consideration, which of the matches is more similar. To overcome this issue I constructed the clustering system from chapter 8.2.7. Each cluster represents an individual grouping all the annotations that are believed to belong to the individual. The name IDs that originate from the name of the video from which again the frames and there of the annotations were extracted, remains unchanged, enabling a backpropagation if a mismatch was made.

Besides the external developed components that have proven themselves for the use on wildlife data in other projects the functionality of the pipeline to process data without manual interaction relies on my core assumptions from chapter 2.3. Rounded up by the self-designed and self-developed clustering mechanism.

8.3.3 Other issues

Less complex, but time consuming issues were deviations from the standard naming convention in the input data. As for almost every Data Science project the data must be inspected and brought to the correct format to meet the input requirements of the program. Even though the same pipeline can be used for other datasets from other conservation projects or camera trap analysis in general, provided that the required assumptions can be made, it will always require time to prepare the data and metadata in a way and format that the program can process it properly.

A minor issue was the format in which the different components, namely DeepLabCut, MegaDetector and IBEIS, process and save location information on bounding boxes, annotations and body part estimations.

9 EXPERIMENT AND RESULTS

The pipeline described in the last chapter was applied to the leopard dataset at hand. The ground truth data for the evaluation are the individual tags for the leopards given by experts from the PanAfrican Programme. The system will be evaluated on the final output of the entire pipeline, the amount of correctly reidentified and matched leopards. The interim components can hardly be evaluated based on ground truth data, because no ground truth data for the special application is available, like body parts or bounding boxes.

9.1 Evaluation of single components

The following chapters assess the performance of individual components of the pipeline. A sound data-based evaluation cannot be conducted for all individual components due to missing ground truth data. The overall performance of the pipeline will be evaluated instead.

9.1.1 Evaluation of the MegaDetector

Due to the missing labels on bounding boxes no automated evaluation based on ground truth data can be conducted. The results from the MegaDetector were visually examined on the annotations in the IBEIS GUI. Within 192 videos with 3183 images and annotations the MegaDetector conducted wrong detections in 15 videos. But, in all but one video, the misdetections concern only a few frames of the video. In the consecutive frames, in which the leopard was visible, it was detected correctly. For two of the 15 videos it was not clearly discernible for the human eye, if it truly is a mismatch. The mismatches originated from images with bad lighting conditions or the leopard being far in the background. An incorrectly made detection in an empty image is considered to be less harmful than a missed detection, when an animal is truly present. In 14 of the 15 videos no leopard was present. Only in one case the MegaDetector chose an incorrect region of interest over the leopard that is actually seen in the image (Figure 9.1 left). The detection was correct in adjacent frames with nearly the same background scenery (Figure 9.1right)

Figure 9.1 **Misdetection MegaDetector**
Left: A detection at the bottom left is found instead of the animal on the animal in the image pointed out by the red error. Right: In a consecutive frame with nearly the same scenery the animal is detected correctly.

None of the annotations with incorrect detections met the threshold for a match in the IBEIS program. If the quality of the image is too low or the animal is too far in the background for the MegaDetector to detect the animal, then the SIFT algorithm from IBEIS is probably also not able to find features.

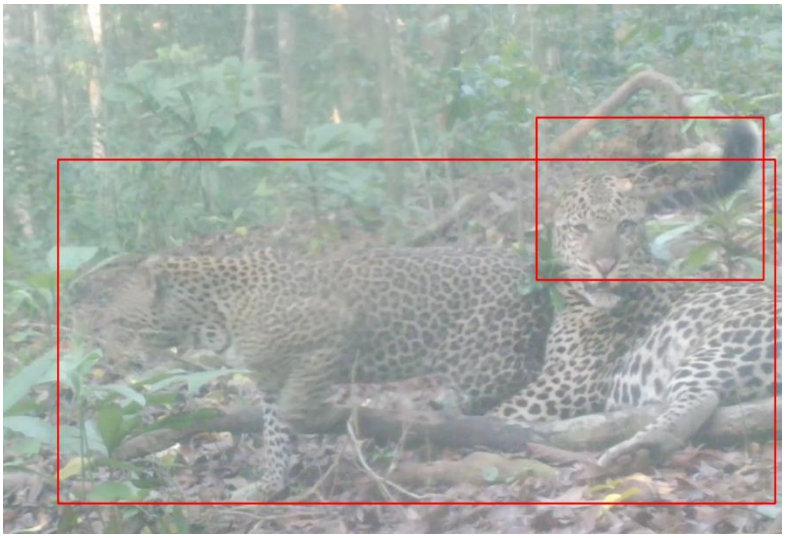


Figure 9.2 **Detection of 2 animals by the MegaDetector**
Footage of images with multiple individuals are of high interest for ecologists.

A collection of the incorrectly detected images can be viewed in the Github repository [154] for this project, as well as some impressive good examples, in which the MegaDetector detected extremely well hidden animals in images of low quality or bad lighting conditions similar to the example in chapter 8.2.3.

For the rare situations where two individuals are captured on tape the MegaDetector recognized both and returned two bounding boxes (see Figure 9.2). It can be filtered on those videos, which are of high interest for ecologists.

9.1.2 Evaluation of the Side Predictor

The side predictor turned out to be the component with the poorest performance in the pipeline. Table 9.1 summarizes the different architectures with the number of classes and the type of the input data with the resulting accuracy on the testset. The AcinoSet was split into trainingset and testset. The accuracies are measured on the AcinoSet testset.

Dataset	Input	Architecture type	Accuracy
AcinoSet	DeepLabCut body part estimations*	Neural network	0.4811
AcinoSet	Entire image	CNN – ResNet50	0.4226
AcinoSet	Bounding box cutted out	CNN – ResNet50	0.5207
AcinoSet	Bounding box cutted out	CNN – MobileNetV2	0.5293
Leopard Set	Entire image	CNN – ResNet50	0.3912

Table 9.1 **Accuracies on cheetah testset for side predictor models**
Different side predictor models trained on multiple architectures and different datasets.
*After the frame selection based on the MegaDetector results.

Each architecture was tried out with different hyperparameters and loss functions with up to 300 epochs. The performance on the foreign leopard dataset with a different ecosystem background is even lower. Based on the little promising results, no further research was done on the side predictor for the pipeline. Due to the low accuracies the side predictor component was not included in the pipeline for the application on the leopard dataset.

Chapter 9.2 will outline the few cases on which the viewpoint was an issue and led to a mismatch. By filtering the annotations on their viewpoint prior to the HotSpotter more information is lost, because annotations will not be queried against each other even though the same flank is visible, but incorrectly classified by the side predictor.

9.2 Results for the leopard application case

The pipeline presented in the previous chapter has its crucial test on the leopard dataset from the PanAfrican Programme. For the evaluation of the system a subsets was generated for which reliable tags are available. The subset contains all videos for which finally approved tags on the individuals are available and additionally videos for which temporary tags exist. The experts add temporary tags on leopard videos that are assumed to show the same individual, but are not yet verified and cannot be matched to any of the known individuals. Temporary IDs that were merged with a name are all merged under this ID for the evaluation, stating that it is the same individual. Matches within temporary IDs are considered a correct match, if it is matched to the same temporary ID. The tags labeled by the experts are treated as ground truth data.

The subset consists of 210 videos with 3183 annotations remaining after the filtering of empty frames by the MegaDetector. The videos were captured at 8 different sites in 59 cells with a total of 68 different camera spots. The cameras were triggered by 16 different known and verified leopards and supplementary 40 temporary IDs that are partly assumed to belong to one of the 16 individuals.

For the clustering process different thresholds for the score were defined to compare the results. In total 151051 annotation pairings were queried, whereof 3534 reached a score above 1.5 and 2156 pairings exceeded a score of 2. The computing time of the matching process for the subset took 2 hours and 56 minutes, not including the computing time of the preceding components. The preparation of the data of the entire unfiltered leopard dataset processing through all components until the feature matching process took around 3 days with the hardware listed in chapter 8.1.

For each name pairing the score with the best matching annotations is considered. Meaning two videos are compared by the best matching frame pair.

With a threshold of 1.5 for the matching score 128 pairings were matched and grouped into 19 clusters. 97 of the 128 pairings are correct matches according to the ground truth data. This equates to a share of 75.78%. Table 9.2 lists the % of correct matches and number of clusters for different thresholds for the score. The highest reached matching

score was 116.49 followed by the second highest with 68. 23 matches were in the score interval between 20 and 68.

Threshold for score	% of correct matches	Number of clusters
1.5	75.78 %	19
2	75.68 %	18
2.5	78.00 %	18
3	77.89 %	18
5	84.29 %	13
10	81.81 %	10
20	83.33 %	5

Table 9.2 **Application results for the application on the leopard dataset**
Summary on the percentage of correct matches out of the made matches, based on the selected threshold, and the resulting number of clusters.

Interestingly one camera spot sticks out. For 55 out of the 128 matches the location for at least one of the pairings annotation was the camera spot 38 with at Twin Oaks (fake site name). 45 of the matches are correct, resulting in a percentage of 81.81 correct matches at this location. This incident could be caused by different factors, like the number of triggered videos, the number of tags, the location itself, the leopard density and the position of the camera. The spot is shown in Figure 9.3. The camera at this location is placed with a good view on the scene and the habitat within the scene is not too dense, limiting the noise and is brightly illuminated. Furthermore, the mound at the spot seems to be a popular favorite spot for some leopards. 3 named individuals and 2 with a temporary ID tags were verified at that camera spot. For this spot a high number of 37 videos were tagged by the expert, which also drives up the number of matches. All of the above are an indication that the spot is often visited by leopards and footage of high quality is captured, leading to multiple tags from the experts.

Figure 9.3 **Popular cameraspot**
The spot seems to be popular for leopards. The camera trap at this location is often triggered by leopards and provides a well illuminated scenery resulting in a high number of tagged videos from that location.

9.2.1 Mismatches

The following chapter highlights some of the mismatches that were made and points out possible reasons. For the visualization of the matches the IBEIS GUI was used.

For all mismatches having a matching score higher than 10, the location of the annotations is the same. For the wrong matches with a score between 2 and 4 only a single mismatch stems from annotations captured at the same location. Table 9.3 shows the percentage of the mismatches that were made for pairings where the underlying footage was taken at the same location. For lower scores the percentage of mismatches from the same location decreases. For scores lower than 2 no mismatches due to the annotations being from the same camera spot are made. The large number of mismatches with a high score imply that strong non-moving features are present in the background with a higher distinctiveness than the features of the leopard.

For scores:	% of the mismatches for annotations at same location
g>10	100%
>5	90.91%
>4	78.57%
>3	52.38%
2 - 4	15.38%
1.5 - 2	0%

Table 9.3 **Mismatches for images captured at the same location**
The percentage of mismatches that can be traced back to matching of the scenery in the background for different thresholds for the matching score.

Figure 9.5 illustrates an example of a mismatch caused by matched features in the background. The red and yellow circles mark the features.

Further analysis of the mismatches revealed a general issue. Some of the footage has subtitles automatically added by the camera trap device. 9 out of the 31 mismatches made in total were matches on digits and letters as depicted in **Fehler! Verweisquelle konnte nicht gefunden werden.** 9.4. This problem can easily be solved by cutting of a thin strip at the bottom of each image.

Figure 9.4 **Mismatches caused by automatic subtitles**
Features are marked in yellow and red.

Figure 9.5 **Mismatches caused by features in the background**
Features in the background are matched for images taken at the same location.

The mismatches of two leopards stood out. The two leopards, namely Jelani and Tau shared three matches. The matched video of Jelani is always the same, but the counterpart of Tau stemmed from three different videos. The matches had scores of 4.31, 3.50 and 1.50. In the ground truth data they are labeled as different individuals. A closer look at the rosettes on the right hind limb reveal a chance of Jelani and Tau being the same individual. Figure 9.6 is zoomed in to the right hind limb. The circles of different color show potential visual feature matches analyzed by myself. The match must of course be evaluated by experts from the PanAfrican Programme.

Further mismatches included a match between a rock and a leopard and mismatches due to features matched in the background. In one single case a right flank is matched with a left flank of another leopard with a score of 3.6.

Figure 9.6 **Matching of Jelani and Tau**
In the ground truth data they are labeled as two different leopards. Top: Features from the IBEIS program. Bottom left: Visual feature detection by a human on zoomed in picture. Bottom right: original image zoomed in.

9.2.2 Results after trouble shooting

After cleaning the mismatches from the avoidable mismatches for which the subtitles are matched and assuming that Jelani and Tau are the same individual the updated values from chapter 9.2 are shown in Table 9.4. 116 total matches remain with 97 correct matches leading to a share of 83.62% of correct matches.

Threshold for score	% of correct matches
1.5	83.62 %
2	83.17 %
2.5	84.78 %
3	85.06 %
5	85.51 %
10	81.81 %
20	83.33 %

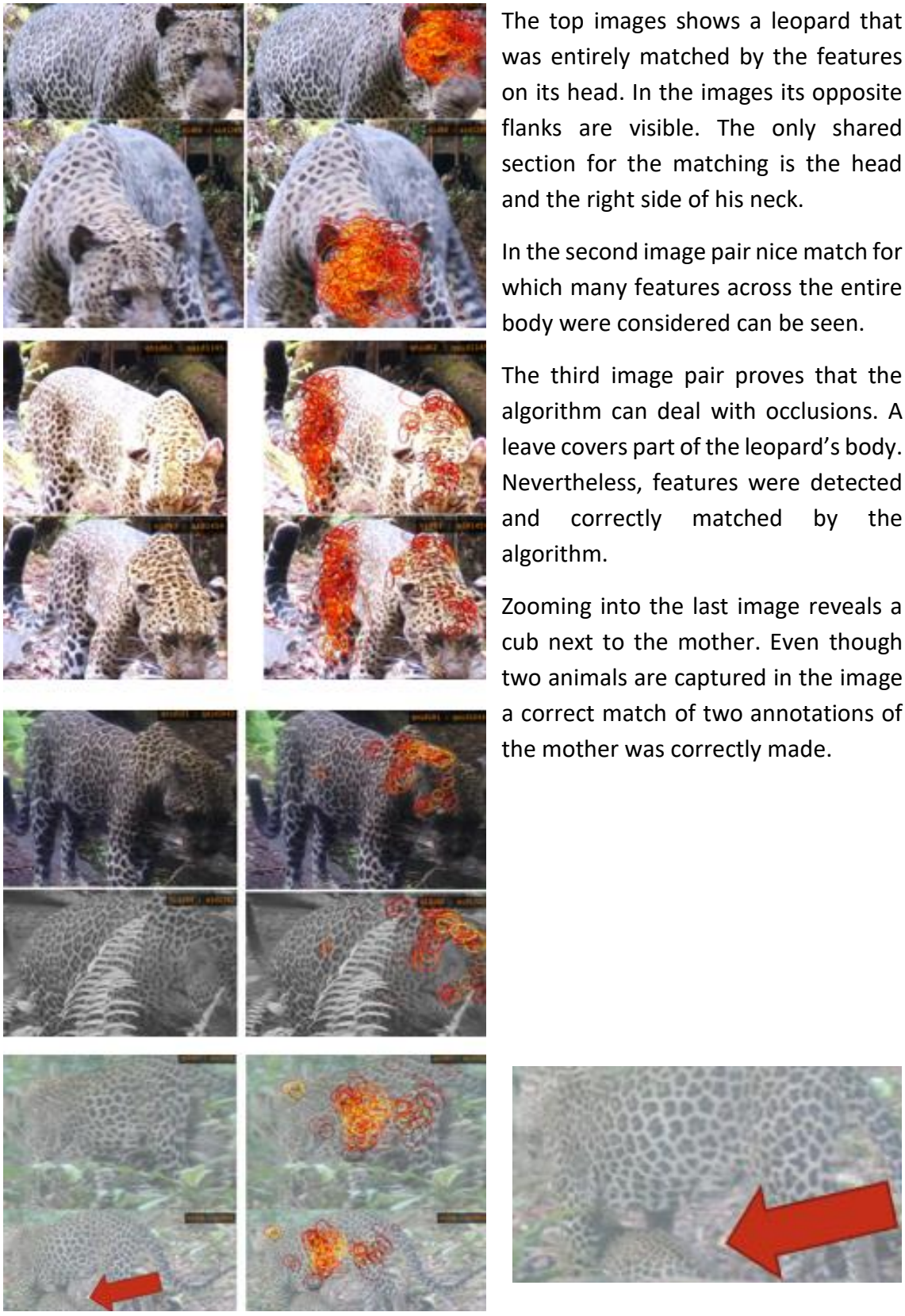
Table 9.4 **Updated table after troubleshooting**
Results after filtering of the avoidable mismatches described in chapter 9.2.1.

Table 9.4 also points out that the choice of the threshold value does not necessarily lead to higher or lower performance of the program measured on the percentage of correct matches. A lower threshold allows more potential matches, giving the chance to match more individuals, even if the underlying footage is challenging. A higher threshold for a match leads to less total value of incorrectly matched individuals, but bears the risk to miss more matches. The percentage of correct matches out of all matches remains the same. Depending on the use case it must be individually decided what is of a higher importance.

The potential matches found in the process of this thesis will be handed to the experts for verification.

9.2.3 Good and interesting matches

In the following some of the good matches are presented that stand out by special characteristics or visualize the matching well.



The top images shows a leopard that was entirely matched by the features on its head. In the images its opposite flanks are visible. The only shared section for the matching is the head and the right side of his neck.

In the second image pair nice match for which many features across the entire body were considered can be seen.

The third image pair proves that the algorithm can deal with occlusions. A leave covers part of the leopard's body. Nevertheless, features were detected and correctly matched by the algorithm.

Zooming into the last image reveals a cub next to the mother. Even though two animals are captured in the image a correct match of two annotations of the mother was correctly made.

Figure 9.7 **IBEIS matches**
Matches were visualized with the IBEI GUI

10 CONCLUSION

In this thesis the problem of animal reidentification based on camera trap data was addressed. The aim was to eliminate manual interaction for labeling or decision making on matches.

The core idea in this thesis to take advantage of video data and its consecutive frames showing the same animal enabled the automatic feeding of the database for the matching algorithm. Footage of one individual from different viewpoints was collected, if the data allowed.

With components in the pipeline that have proven its functionality on wildlife data in the past videos can be consumed without any preparation and are grouped to clusters representing an individual.

The initial concept with the side predictor to prevent the matching of opposite flanks did not perform well enough to be implemented in the pipeline and would even lower the overall performance. In the experiment on the leopard dataset it became clear that the viewpoint is not a big factor for mismatches. The biggest concern for mismatches turned out to be the background for images taken at the same location, due to the fact that camera traps are fixed to a scenery.

10.1 Future work

To improve the performance of the leopard-memory pipeline research on how to overcome the background issue must be done. Approaches might be to cut out the annotation from the images more accurately. Instead of rectangular boxes the actual shape of the animal could be extracted. In noisy sceneries like the jungle this will be challenging. Similar to the pose estimator a shape detector could be embedded. By taking advantage of the video data instead of image data additional information on the shape and movement could be extracted with optical flows. Furthermore optical flows could help resolve the necessity on the inspected animals to be solitary. The information on the movement of one individual could ease to identify different individuals within one image.

A simple way to solve the problem of opposite flanks and to get footage from multiple viewpoints would be to setup the camera traps differently in future projects. If two cameras are placed facing each other the animal will most likely be seen from both sides.

For other use cases in which the dataset contains footage of multiple species a classification algorithm can be added to the pipeline prior to the feature detector.

The GPS data that is available could be used to build further conditions for the queries based on the distance of the locations. From a biological point of view it would be interesting to map the clusters on a map of the region to have a visual overview of the leopards' habitats.

The developed pipeline will be used in the near future to help the PanAfrican Programme to identify further individuals. Potential matches were already found, but could not be included in the experiment, because no sound evaluation could have been done. They need to be verified by the experts first.

Another experiment for the future is to apply the pipeline to other species, to validate, if it suitable for cross-species applications.

11 BIBLIOGRAPHY

[1] *Brockhaus Faszination Natur - Säugetiere II*. .

[2] T. C. Hsu and K. Benirschke, “Panthera pardus (Leopard),” *An Atlas Mamm. Chromosom.*, vol. 8235, pp. 133–135, 1968, doi: 10.1007/978-1-4615-6424-9_34.

[3] J. O. Whitaker Jr and R. E. Mumford, “Handbook of the mammals of the world. Volume 1: carnivores.” 2009.

[4] A. C. Kitchener *et al.*, “A revised taxonomy of the Felidae: The final report of the Cat Classification Task Force of the IUCN Cat Specialist Group,” *Cat News*, 2017.

[5] K. Nowell and P. Jackson, “Leopard Panthera pardus (Linnaeus, 1758),” *Wild Cats Status Surv. Conserv. Action Plan. IUCN Publ. Switz.*, pp. 44–47, 1996.

[6] T. N. Bailey, “The African leopard: ecology and behavior of a solitary felid New York.” Columbia University Press.[Google Scholar], 1993.

[7] arte, WDR, ORF, and ARD, *Sambia - Königreich der Leopardin*. .

[8] Wildlife Films Botswana for National Geographic Channels, *Ein Leopard mit kleinen Schwächen*. .

[9] A. Landsmann, “Tutorial for Leopard Matching,” *10/12/2016*. https://talk.chimpandsee.org/boards/BCP000000t/discussions/DCP0001j1c.

[10] “Pan African Programme: The Cultured Chimpanzee,” 2016. http://panafrican.eva.mpg.de/english/where_we_work.php.

[11] “Chimp & See - Map & Site Locations.” https://talk.chimpandsee.org/boards/BCP0000007/discussions/DCP00009aw.

[12] Max-Plank-Institut für evolutionäre Anthropologie, “Pan African Chimpanzee Project.” https://www.eva.mpg.de/de/primat/research-groups/great-ape-evolutionary-ecology-and-conservation/panaf/.

[13] M. Arandjelovic *et al.*, “Pan African Programme The cultured chimpanzee: Guidelines for research and data collection,” pp. 1–105, 2014.

[14] L. White and A. Edwards, “Conservation research in the African,” *Conserv. Res. African*, 2000.

[15] S. C. Silver *et al.*, “The use of camera traps for estimating jaguar Panthera onca abundance and density using capture/recapture analysis,” *Oryx*, vol. 38, no. 2, pp. 148–154, 2004.

[16] “Zooniverse,” 2021. https://www.zooniverse.org/.

[17] A. Mathis, S. Schneider, J. Lauer, and M. W. Mathis, “A Primer on Motion Capture with Deep Learning: Principles, Pitfalls, and Perspectives,” *Neuron*, vol. 108, no. 1, pp. 44–65, 2020, doi: 10.1016/j.neuron.2020.09.017.

[18] E. Hergenröther, *Computer Vision Lecture - Chapter 3 - Analytische Computer Vision - Darmstadt University of Applied Science*. 2020.

[19] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *Int. J.*

Comput. Vis., vol. 60, no. 2, pp. 91–110, 2004, doi: 10.1023/B:VISI.0000029664.99615.94.

[20] T. Lindeberg, “Scale invariant feature transform,” 2012.

[21] S. Nayar, “Lecture Series on First Principles of Computer Vision,” *Columbia Engineering*. https://fpcv.cs.columbia.edu/.

[22] D. G. Lowe, “Object recognition from local scale-invariant features,” *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2, pp. 1150–1157, 1999, doi: 10.1109/iccv.1999.790410.

[23] E. Hergenröther, *Computer Vision Lecture - Chapter 4 - Convolutional Neural Networks- Darmstadt University of Applied Science*. 2020.

[24] E. Alpaydin, *Neural Networks and Deep Learning*. 2021.

[25] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[26] Y. LeCun, C. Cortes, and C. Burges, “MNIST handwritten digit database.” Florham Park, NJ, USA, 2010.

[27] “Machine Learning & Deep Learning Fundamentals,” 2017. https://deeplizard.com/learn/video/YRhxdVk_sIs.

[28] “Convolutional Neural Networks for Visual Recognition.” https://cs231n.github.io/convolutional-networks/.

[29] F. Chollet, “Keras: The python deep learning library,” *Astrophys. Source Code Libr.*, p. ascl-1806, 2018.

[30] Martin Abadi *et al.*, “{TensorFlow} Large-Scale Machine Learning on Heterogeneous Systems.” 2015, [Online]. Available: httpswww.tensorflow.org.

[31] A. Paszke *et al.*, “PyTorch: An Imperative Style, High-Performance Deep Learning Library,” in *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., 2019, pp. 8024–8035.

[32] G. Johansson, “Visual perception of biological motion and a model for its analysis,” *Percept. Psychophys.*, vol. 14, no. 2, pp. 201–211, 1973.

[33] T. D. Pereira *et al.*, “Fast animal pose estimation using deep neural networks,” *Nat. Methods*, vol. 16, no. 1, pp. 117–125, 2019.

[34] A. Mathis *et al.*, “DeepLabCut: markerless pose estimation of user-defined body parts with deep learning,” *Nat. Neurosci.*, vol. 21, no. 9, pp. 1281–1289, 2018, doi: 10.1038/s41593-018-0209-y.

[35] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele, “2d human pose estimation: New benchmark and state of the art analysis,” in *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition*, 2014, pp. 3686–3693.

[36] A. Mathis *et al.*, “Pretraining boosts out-of-domain robustness for pose estimation,” *Proc. - 2021 IEEE Winter Conf. Appl. Comput. Vision, WACV 2021*, pp. 1858–1867, 2021, doi: 10.1109/WACV48630.2021.00190.

[37] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, “OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 1, pp. 172–186, 2019.

[38] A. Toshev and C. Szegedy, “DeepPose_Human_Pose_2014_CVPR_paper.pdf,” *IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 1653–1660, 2014, [Online]. Available: https://www.cv-foundation.org/openaccess/content_cvpr_2014/html/Toshev_DeepPose_Human_Pose_2014_CVPR_paper.html.

[39] L. Pishchulin *et al.*, “DeepCut: Joint Subset Partition and Labeling for Multi Person Pose Estimation,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 4929–4937, doi: 10.1109/CVPR.2016.533.

[40] E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, and B. Schiele, “Deepercut: A deeper, stronger, and faster multi-person pose estimation model,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9910 LNCS, pp. 34–50, 2016, doi: 10.1007/978-3-319-46466-4_3.

[41] OpenCV, “Open Source Computer Vision Library.” 2015, [Online]. Available: https://opencv.org.

[42] Y. Huang, B. Sun, H. Kan, J. Zhuang, and Z. Qin, “FollowMeUp Sports: New benchmark for 2D human keypoint recognition,” in *Chinese Conference on Pattern Recognition and Computer Vision (PRCV)*, 2019, pp. 110–121.

[43] Microsoft, “AI for Earth.” https://www.microsoft.com/en-us/ai/ai-for-earth.

[44] S. Beery, D. Morris, and S. Yang, “Efficient Pipeline for Camera Trap Image Review,” 2019, [Online]. Available: http://arxiv.org/abs/1907.06772.

[45] S. Yang, “WildLabs Tech Tutors. How do I get started with MegaDetector?,” 2021, [Online]. Available: https://www.wildlabs.net/resources/tech-tutors/how-do-i-get-started-megadetector.

[46] “Snapshot Serengeti.” https://www.snapshotserengeti.org.

[47] S. Beery, D. Morris, and S. Yang, “MegaDetector - Git Repository.” [Online]. Available: https://github.com/microsoft/CameraTraps.

[48] M. A. for E. Zooniverse, the Evolving AI Lab, niversity of Minnesota Lion Center, Snapshot Safari, “LILA BC - Labeled Information Library of Alexandria: Biology and Conservation.” https://lila.science/.

[49] Microsoft AI for Earth, “MegaDetector Web API.” https://cameratrapdemo.westus2.cloudapp.azure.com/upload.

[50] “MegaDetector on Google Colab.” https://colab.research.google.com/github/microsoft/CameraTraps/blob/master/detection/megadetector_colab.ipynb.

[51] “Wildlife Protection Solutions.” https://wildlifeprotectionsolutions.org/.

[52] M. Mathis, “DeepLabCut - Mathis Laboratory.” http://www.mackenziemathislab.org/deeplabcut/.

[53] Mathis Laboratory, “DeepLabCut - Code.” [Online]. Available: https://github.com/DeepLabCut/DeepLabCut.

[54] “ImageNet.” https://www.image-net.org/.

[55] M. W. Mathis and A. Mathis, “Deep learning tools for the measurement of animal behavior in neuroscience,” *Curr. Opin. Neurobiol.*, vol. 60, pp. 1–11, 2020, doi: 10.1016/j.conb.2019.10.008.

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

[57] T. Nath, A. Mathis, A. C. Chen, A. Patel, M. Bethge, and M. W. Mathis, “Using DeepLabCut for 3D markerless pose estimation across species and behaviors,” *Nat. Protoc.*, vol. 14, no. 7, pp. 2152–2176, 2019, doi: 10.1038/s41596-019-0176-0.

[58] B. D. Lucas, T. Kanade, and others, “An iterative image registration technique with an application to stereo vision,” 1981.

[59] M. Mathis, “DeepLabCut - Modelzoo.” http://www.mackenziemathislab.org/dlc-modelzoo%0A.

[60] J. L. Alexander Mathis, Mackenzie Mathis, “DeepLabCut Workshop,” 2020. [Online]. Available: https://github.com/DeepLabCut/DeepLabCut-Workshop-Materials.

[61] A. Mathis, “DeepLabCut @ CMU Open Science Symposium 2019.” 2013, [Online]. Available: https://www.youtube.com/watch?v=ZjWPHM0sL4E.

[62] A. F. Meyer, J. O’Keefe, and J. Poort, “Two distinct types of eye-head coupling in freely moving mice,” *Curr. Biol.*, vol. 30, no. 11, pp. 2116–2130, 2020.

[63] N. Bonacchi *et al.*, “Data architecture for a large-scale neuroscience collaboration,” *BioRxiv*, p. 827873, 2020.

[64] A. Mathis and R. Warren, “On the inference speed and video-compression robustness of DeepLabCut,” *bioRxiv*, pp. 1–10, 2018, doi: 10.1101/457242.

[65] D. Joska *et al.*, “AcinoSet: A 3D Pose Estimation Dataset and Baseline Models for Cheetahs in the Wild,” pp. 13901–13908, 2021, doi: 10.1109/icra48506.2021.9561338.

[66] Mathis Laboratory, “DeepLabCut on Google Colab.” [Online]. Available: https://colab.research.google.com/github/DeepLabCut/DeepLabCut/blob/master/examples/COLAB_YOURDATA_TrainNetwork_VideoAnalysis.ipynb.

[67] Maxime Vidal and Steffen Schneider, “DeepLabCut Labeling App.” https://contrib.deeplabcut.org/#.

[68] J. P. Crall, “Identifying Individual Animals using Ranking, Verification, and Connectivity,” vol. 2017, no. June, 2017.

[69] “WildMe.” https://wildme.org.

[70] J. P. Crall, “HotSpotter Software.” 2017, [Online]. Available:

https://web.archive.org/web/20170602093944/http:/cs.rpi.edu/hotspotter/.

[71] J. P. Crall, “IBEIS - Image Based Ecological Information System - Repository.” [Online]. Available: https://github.com/Erotemic/ibeis.

[72] J. Wieczorek *et al.*, “Darwin Core: an evolving community-developed biodiversity data standard,” *PLoS One*, vol. 7, no. 1, p. e29715, 2012.

[73] D. I. Rubenstein *et al.*, “The great zebra and giraffe count: The power and rewards of citizen science,” in *Technical report*, Kenya Wildlife Service, 2015.

[74] “WildBooks.” https://wildme.org/#/platforms.

[75] “Wildlife Insights.” https://www.wildlifeinsights.org/.

[76] “ZambaCloud - Computer Vision for Wildlife Conservation.” https://www.zambacloud.com/.

[77] A. Swanson, M. Kosmala, C. Lintott, R. Simpson, A. Smith, and C. Packer, “Snapshot Serengeti, high-frequency annotated camera trap images of 40 mammalian species in an African savanna,” *Sci. data*, vol. 2, no. 1, pp. 1–14, 2015.

[78] M. S. Palmer, S. E. Huebner, M. Willi, L. Fortson, and C. Packer, “Citizen science, computing, and conservation: How can ‘Crowd AI’ change the way we tackle large-scale ecological challenges?,” *Hum. Comput.*, vol. 8, no. 2, pp. 54–75, 2021, doi: 10.15346/hc.v8i2.123.

[79] J. Parham, J. Crall, C. Stewart, T. Berger-Wolf, and D. Rubenstein, “Animal population censusing at scale with citizen science and photographic identification,” *AAAI Spring Symp. - Tech. Rep.*, vol. SS-17-01-, pp. 37–44, 2017.

[80] M. Willi *et al.*, “Identifying animal species in camera trap images using deep learning and citizen science,” *Methods Ecol. Evol.*, vol. 10, no. 1, pp. 80–91, 2019, doi: 10.1111/2041-210X.13099.

[81] “Kaggle - iWildcam 2021 Challenge.” https://www.kaggle.com/c/iwildcam2021-fgvc8.

[82] DrivenData, “Hakuna Ma-data challenge.” https://www.drivendata.org/competitions/59/camera-trap-serengeti/.

[83] “The Snow Leopard Trust.” https://snowleopard.org/.

[84] M. Hamilton *et al.*, “Flexible and Scalable Deep Learning with MMLSpark,” vol. 1, pp. 1–12, 2018, [Online]. Available: http://arxiv.org/abs/1804.04031.

[85] N. Banupriya, S. Saranya, R. Jayakumar, R. Swaminathan, S. Harikumar, and S. Palanisamy, “Animal detection using deep learning algorithm,” *J. Crit. Rev.*, vol. 7, no. 1, pp. 434–439, 2020, doi: 10.31838/jcr.07.01.85.

[86] M. S. Norouzzadeh *et al.*, “Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning,” *Proc. Natl. Acad. Sci.*, vol. 115, no. 25, pp. E5716–E5725, 2018.

[87] H. Nguyen *et al.*, “Animal recognition and identification with deep convolutional neural networks for automated wildlife monitoring,” *Proc. - 2017 Int. Conf. Data Sci. Adv. Anal. DSAA 2017*, vol. 2018-Janua, no. December, pp. 40–49, 2017, doi:

10.1109/DSAA.2017.31.

[88] R. Yoshihashi, S. You, W. Shao, M. Iida, R. Kawakami, and T. Naemura, “Classification-Reconstruction Learning for Open-Set Recognition Data61-CSIRO.”

[89] S. Beery, G. Wu, V. Rathod, R. Votel, and J. Huang, “Context r-cnn: Long term temporal context for per-camera object detection,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 13075–13085.

[90] J. Stenum, K. M. Cherry-Allen, C. O. Pyles, R. D. Reetzke, M. F. Vignos, and R. T. Roemmich, “Applications of Pose Estimation in Human Health and Performance across the Lifespan,” *Sensors*, vol. 21, no. 21, 2021, doi: 10.3390/s21217315.

[91] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, “Realtime multi-person 2d pose estimation using part affinity fields,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7291–7299.

[92] A. Mathis, T. Biasi, B. Rogers, M. Bethge, and M. Weygandt Mathis, “ImageNet performance correlates with pose estimation robustness and generalization on out-of-domain data,” *Uncertain. Deep Learn.*, no. Figure 2, 2020.

[93] M. R. I. Hossain and J. J. Little, “Exploiting temporal information for 3d human pose estimation,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 68–84.

[94] A. S. Fangbemi, Y. F. Lu, M. Y. Xu, X. W. Luo, A. Rolland, and C. Raissi, “ZooBuilder: 2D and 3D Pose Estimation for Quadrupeds Using Synthetic Data,” vol. 39, no. 8, pp. 2–3, 2020, [Online]. Available: http://arxiv.org/abs/2009.05389.

[95] S. Zuffi, A. Kanazawa, D. Jacobs, and M. J. Black, “3D menagerie: Modeling the 3D shape and pose of animals,” *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 5524–5532, 2017, doi: 10.1109/CVPR.2017.586.

[96] J. Cao, H. Tang, H. S. Fang, X. Shen, C. Lu, and Y. W. Tai, “Cross-domain adaptation for animal pose estimation,” *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2019-Octob, pp. 9497–9506, 2019, doi: 10.1109/ICCV.2019.00959.

[97] “AcinoSet.” https://www.dropbox.com/sh/z3uv6pnk7paygph/AAAiJOavquW89uPlz_Jzjtfua?dl=0.

[98] “AcinoSet Labels.” https://www.dropbox.com/sh/kp5kmatbv5cdjx2/AABfJGb7ktVK_L0lybOLQIbJa?dl=0.

[99] S. Team, “Convolutional Neural Networks (CNN): Step 3 - Flattening,” 2018. https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-3-flattening/.

[100] C. Yan *et al.*, “BV-Person: A Large-Scale Dataset for Bird-View Person Re-Identification,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10943–10952.

[101] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian, “Scalable person re-

identification: A benchmark," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1116–1124.

[102] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi, "Performance measures and a data set for multi-target, multi-camera tracking," in *European conference on computer vision*, 2016, pp. 17–35.

[103] W. Li, R. Zhao, T. Xiao, and X. Wang, "Deepreid: Deep filter pairing neural network for person re-identification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 152–159.

[104] S. Ding, L. Lin, G. Wang, and H. Chao, "Deep feature learning with relative distance comparison for person re-identification," *Pattern Recognit.*, vol. 48, no. 10, pp. 2993–3003, 2015.

[105] E. Ahmed, M. Jones, and T. K. Marks, "An improved deep learning architecture for person re-identification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3908–3916.

[106] P. Marchwica, M. Jamieson, and P. Siva, "An evaluation of deep cnn baselines for scene-independent person re-identification," in *2018 15th Conference on Computer and Robot Vision (CRV)*, 2018, pp. 297–304.

[107] H. Fan, L. Zheng, C. Yan, and Y. Yang, "Unsupervised person re-identification: Clustering and fine-tuning," *ACM Trans. Multimed. Comput. Commun. Appl.*, vol. 14, no. 4, pp. 1–18, 2018.

[108] M. Millar, "Review of Current Methods for Re-Identification in Computer Vision.," *Open Sci. J.*, vol. 4, no. 1, pp. 1–10, 2019, doi: 10.23954/osj.v4i1.2141.

[109] D. Cheng, Y. Gong, S. Zhou, J. Wang, and N. Zheng, "Person re-identification by multi-channel parts-based cnn with improved triplet loss function," in *Proceedings of the iEEE conference on computer vision and pattern recognition*, 2016, pp. 1335–1344.

[110] D. Yi, Z. Lei, S. Liao, and S. Z. Li, "Deep metric learning for person re-identification," in *2014 22nd International Conference on Pattern Recognition*, 2014, pp. 34–39.

[111] R. R. Varior, M. Haloi, and G. Wang, "Gated siamese convolutional neural network architecture for human re-identification," in *European conference on computer vision*, 2016, pp. 791–808.

[112] H. Liu, J. Feng, M. Qi, J. Jiang, and S. Yan, "End-to-end comparative attention networks for person re-identification," *IEEE Trans. Image Process.*, vol. 26, no. 7, pp. 3492–3506, 2017.

[113] S. Bak, E. Corvee, F. Bremond, and M. Thonnat, "Person re-identification using spatial covariance regions of human body parts," in *2010 7th IEEE International Conference on Advanced Video and Signal Based Surveillance*, 2010, pp. 435–440.

[114] Y. Lin *et al.*, "Improving person re-identification by attribute and identity learning," *Pattern Recognit.*, vol. 95, pp. 151–161, 2019.

[115] S. Gong, M. Cristani, S. Yan, and C. C. Loy, *Re-Identification*. .

[116] L. Zheng *et al.*, "Person Re-identification in the Wild," pp. 1367–1376.

[117] N. Zhang, M. Paluri, Y. Taigman, R. Fergus, and L. Bourdev, “Beyond frontal faces: Improving person recognition using multiple cues,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4804–4813.

[118] X. Ma *et al.*, “Person re-identification by unsupervised video matching,” *Pattern Recognit.*, vol. 65, no. March 2016, pp. 197–210, 2017, doi: 10.1016/j.patcog.2016.11.018.

[119] R. Woesler, “Real-time Recognition and Reidentification of Vehicles from Video Data with high Reidentification Rate.”

[120] D. Chen, H. Li, T. Xiao, S. Yi, and X. Wang, “Video Person Re-identification with Competitive Snippet-Similarity Aggregation and Co-attentive Snippet Embedding,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 1169–1178, 2018, doi: 10.1109/CVPR.2018.00128.

[121] G. M. Jolly and J. M. Dickson, “The problem of unequal catchability in mark–recapture estimation of small mammal populations,” *Can. J. Zool.*, vol. 61, no. 4, pp. 922–927, 1983.

[122] K. S. Bradfield, *Photographic identification of individual Archey’s frogs, Leiopelma archeyi, from natural markings*, vol. 191. Department of Conservation Wellington, New Zealand, 2004.

[123] S. Schneider, G. W. Taylor, S. Linquist, and S. C. Kremer, “Past, present and future approaches using computer vision for animal re-identification from camera trap data,” *Methods Ecol. Evol.*, vol. 10, no. 4, pp. 461–470, 2019, doi: 10.1111/2041-210X.13133.

[124] S. A. Mizroch, J. A. Beard, and M. Lynde, “Computer assisted photo-identification of humpback whales,” *Rep. Int. Whal. Comm.*, vol. 12, pp. 63–70, 1990.

[125] H. Whitehead, “Computer assisted individual identification of sperm whale flukes,” *Reports Int. Whal. Comm.*, vol. 12, pp. 71–77, 1990.

[126] G. R. Hillman *et al.*, “Computer-assisted photo-identification of individual marine vertebrates: a multi-species system,” *Aquat. Mamm.*, vol. 29, no. 1, pp. 117–123, 2003.

[127] H. J. Weideman *et al.*, “Integral curvature representation and matching algorithms for identification of dolphins and whales,” in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017, pp. 2831–2839.

[128] T. Burghardt, N. Campbell, P. J. Barham, I. C. Cuthill, and R. Sherley, “A fully automated computer vision system for the biometric identification of african penguins (Spheniscus demersus) on robben island,” 2007.

[129] E. Nepovinnykh, T. Eerola, H. Kälviäinen, and G. Radchenko, “Identification of saimaa ringed seal individuals using transfer learning,” in *International Conference on Advanced Concepts for Intelligent Vision Systems*, 2018, pp. 211–222.

[130] R. Bogucki, M. Cygan, C. B. Khan, M. Klimek, J. K. Milczek, and M. Mucha, “Applying deep learning to right whale photo identification,” *Conserv. Biol.*, vol. 33, no. 3, pp. 676–684, 2019.

[131] A. C. Ferreira *et al.*, “Deep learning-based methods for individual recognition in small birds,” *Methods Ecol. Evol.*, vol. 11, no. 9, pp. 1072–1085, 2020, doi: 10.1111/2041-210X.13436.

[132] S. J. B. Carter, I. P. Bell, J. J. Miller, and P. P. Gash, “Automated marine turtle photograph identification using artificial neural networks, with application to green turtles,” *J. Exp. Mar. Bio. Ecol.*, vol. 452, pp. 105–110, 2014.

[133] R. J. Foster and B. J. Harmsen, “A critique of density estimation from camera-trap data,” *J. Wildl. Manage.*, vol. 76, no. 2, pp. 224–236, 2012.

[134] P. D. Meek, K. Vernes, and G. Falzon, “On the reliability of expert identification of small-medium sized mammals from camera trap photos,” *Wildl. Biol. Pract.*, vol. 9, no. 2, pp. 1–19, 2013.

[135] S. Li, J. Li, H. Tang, R. Qian, and W. Lin, “ATRW: A Benchmark for Amur Tiger Re-identification in the Wild,” *MM 2020 - Proc. 28th ACM Int. Conf. Multimed.*, pp. 2590–2598, 2020, doi: 10.1145/3394171.3413569.

[136] L. Hiby, P. Lovell, N. Patil, N. S. Kumar, A. M. Gopalaswamy, and K. U. Karanth, “A tiger cannot change its stripes: using a three-dimensional model to match images of living tigers and tiger skins,” *Biol. Lett.*, vol. 5, no. 3, pp. 383–386, 2009.

[137] J. P. Crall, C. V Stewart, T. Y. Berger-wolf, and D. I. Rubenstein, “HotSpotter - Patterned Species Instance Recognition Siva R . Sundaresan,” pp. 230–237, 2012.

[138] D. T. Bolger, T. A. Morrison, B. Vance, D. Lee, and H. Farid, “A computer-assisted system for photographic mark–recapture analysis. Methods in Ecology and Evolution 3: 813–822.” 2012.

[139] O. Moskvyak, F. Maire, A. O. Armstrong, F. Dayoub, and M. Baktashmotlagh, “Robust Re-identification of Manta Rays from Natural Markings by Learning Pose Invariant Embeddings,” pp. 1–12, 2019, [Online]. Available: http://arxiv.org/abs/1902.10847.

[140] A. Loos and M. Pfitzer, “Towards automated visual identification of primates using face recognition,” in *2012 19th International Conference on Systems, Signals and Image Processing (IWSSIP)*, 2012, pp. 425–428.

[141] A. Ernst and C. Küblbeck, “Fast face detection and species classification of African great apes,” in *2011 8th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2011, pp. 279–284.

[142] M. F. Hansen *et al.*, “Towards on-farm pig face recognition using convolutional neural networks,” *Comput. Ind.*, vol. 98, pp. 145–152, 2018.

[143] C.-A. Brust *et al.*, “Towards automated visual monitoring of individual gorillas in the wild,” in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017, pp. 2820–2830.

[144] A. Freytag, E. Rodner, M. Simon, A. Loos, H. S. Kühl, and J. Denzler, “Chimpanzee faces in the wild: Log-euclidean CNNs for predicting identities and attributes of primates,” in *German Conference on Pattern Recognition*, 2016, pp. 51–63.

[145] D. Schofield *et al.*, “Chimpanzee face recognition from videos in the wild using

deep learning," *Sci. Adv.*, vol. 5, no. 9, p. eaaw0736, 2019.

[146] D. Crouse *et al.*, "LemurFaceID: A face recognition system to facilitate individual identification of lemurs," *Bmc Zool.*, vol. 2, no. 1, pp. 1–14, 2017.

[147] C. L. Witham, "Automated face recognition of rhesus macaques," *J. Neurosci. Methods*, vol. 300, pp. 157–165, 2018.

[148] B. G. Weinstein, "M otion M eerkat: integrating motion video detection and ecological monitoring," *Methods Ecol. Evol.*, vol. 6, no. 3, pp. 357–362, 2015.

[149] P. Chen *et al.*, "A study on giant panda recognition based on images of a large proportion of captive pandas," *Ecol. Evol.*, vol. 10, no. 7, pp. 3561–3573, 2020.

[150] M. Marsot *et al.*, "An adaptive pig face recognition approach using Convolutional Neural Networks," *Comput. Electron. Agric.*, vol. 173, p. 105386, 2020.

[151] K. Wang, C. Chen, and Y. He, "Research on pig face recognition model based on keras convolutional neural network," in *IOP Conference Series: Earth and Environmental Science*, 2020, vol. 474, no. 3, p. 32030.

[152] M. Clapham, E. Miller, M. Nguyen, and C. T. Darimont, "Automated facial recognition for wildlife that lack unique markings: A deep learning approach for brown bears," *Ecol. Evol.*, vol. 10, no. 23, pp. 12883–12892, 2020, doi: 10.1002/ece3.6840.

[153] A. Ardovini, L. Cinque, and E. Sangineto, "Identifying elephant photos by multi-curve matching," *Pattern Recognit.*, vol. 41, no. 6, pp. 1867–1877, 2008.

[154] V. Süßle, "Github Repository - Leopard-Memory," 2021. https://github.com/stvasues/Leopard-Memory.

[155] T. Nath, A. Mathis, A. C. Chen, A. Patel, M. Bethge, and M. W. Mathis, "DeepLabCut Cheetah Model weights." [Online]. Available: http://deeplabcut.rowland.harvard.edu/models/DLC_full_cheetah_resnet_152.tar.gz.

[156] "Tensorflow GPU - Installation requirements." https://www.tensorflow.org/install/gpu.

12 APPENDICES

12.1 Python environment

Functional setup in Python 3.7.11:

Package	Version
cudatoolkit	11.3.1
cudnn	8.2.1
tensorflow	2.3.0
tensorflow-gpu	2.3.0
keras	2.4.3
keras-applications	1.0.8
h5py	2.10.0
deeplabcut	2.2.0.3
megadetector	4.1 (downloaded from the git repository and manually added to the system path) [47]

When facing issues with the cudnn libraries the intructions under the following links could help to solve the problem:

- https://medium.com/analytics-vidhya/cuda-toolkit-on-windows-10-20244437e036
- https://www.joe0.com/2019/10/19/how-resolve-tensorflow-2-0-error-could-not-load-dynamic-library-cudart64_100-dll-dlerror-cudart64_100-dll-not-found/

12.2 DeepLabCut export

Example of a csv export file from a DeepLabCut body part prediction. For each body part a location as (x,y)-coordinates is returned and the likelihood of the estimation for that body part.

bodyparts	r_eye	r_eye	r_eye	l_eye	l_eye	l_eye	r_shoulder	r_shoulder	r_shoulder	r_front_knee	r_front_knee	r_front_knee	...	nose	nose	nose
coords	x	y	p	x	y	p	x	y	p	x	y	p	...	x	y	p
frame 1	403.86	396.07	0.08	420.18	403.86	0.04	347.92	523.99	0.34	324.13	547.90	0.74	...	412.24	403.85	0.02
frame 2	403.9	396.1	0.078	420.2	403.9	0.043	347.9	524	0.358	324.1	547.9	0.75	...	412.2	403.8	0.019
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

12.3 Decoding of temporary leopard IDs

The following table shows which temporary ID is matched to which leopard by the time this thesis was written.

Temporary ID	Leopard
LeoMaleTO06b	Murr
LeoMaleTO08	Murr
LeoMaleTO06	Murr
LeoMaleTO02b	Braveheart
LeoMaleTO02	Braveheart
LeoMaleTO02c	Braveheart
LeoMaleTO01	Hermann
LeoMaleTO03	Hermann
LeoMaleTO01b	Hermann
LeoFemTO01b	Dafne
LeoFemTO01	Dafne
LeoFemXB01	Sadie
LeoFemXB01b	Sadie
LeoMaleXB01	Aran
LeoMaleXB03	Aran
LeoMaleTO06c	Murr

12.4 Image material

Further image material can be viewed in the github repository:

https://github.com/stvasues/Leopard-Memory