DHBW
Duale Hochschule
Baden-Württemberg
Mannheim

# Development of a drone-based evaluation tool for motion analysis in athletics long jump

**Studienarbeit**

im Studiengang

Informationstechnik

*an der Dualen Hochschule Baden-Württemberg Mannheim*

von


| | |
|---|---|
| Name, Vorname: | Faust, Jakob |
| Abgabedatum: | 16.04.2024 |


| | |
|---|---|
| Bearbeitungszeitraum: | 17.10.2023 - 16.04.2024 |
| Matrikelnummer, Kurs: | 5507125, TINF21IT1 |
| Betreuer: | Jürgen Schultheis |

Unterschrift Betreuer: _____

Mannheim, den

# Contents

# List of Figures

# Listings

# List of acronyms

**AI** Artificial Intelligence

**CPU** Central Processing Unit

**FPV** First Person View

**GPU** Graphical Processing Unit

**GUI** Graphical User Interface

# 1 Introduction

Long jump is an athletic discipline that is renowned for its technical complexity and the precise movement patterns it demands from athletes. Even apparently small technical inaccuracies can significantly impact an athlete's performance. Moreover, as shown in [1] the forces during the take-off phase can reach up to 10 times the athlete's body weight, increasing the risk of serious injuries due to technical inaccuracies. Therefore, it is crucial to understand and continuously improve these movement patterns in training. However, taken the high approach velocity[1] into account, this can quickly become a difficult task. Especially the take-off phase can be very short and therefore hard to analyze.

Professional athletes often employ expensive high speed camera systems in combination with body pose markers to capture and analyze every single step they make.
Yet, this approach comes with some limitations. Due to their stationary installation, such camera systems are restricted to a fixed location. Moreover, they often combine multiple cameras like Murray et al. [2] used for sprint analysis in order to be able to capture the whole movement from the beginning of the approach until the landing. This leads to complex post-processing software requirements. Additionally, fixed markers need to be attached to an athletes body to be able to track their body position.

While these methods provide exact and reliable results, they are usually not accessible for hobby- and semi-professional athletes.
In recent years however the advances in Artificial Intelligence (AI) and especially within the area of deep neural network paved the way for analyzing methods that require less complex setups. As of 2023 deep neural networks trained for body pose detection are even used in medical applications like gait analysis [3]. Because of the already extremely high and continuously improving accuracy, its application within the area of motion analysis in long jump is treated in the scope of this work.
A semi-autonomous drone based evaluation tool is newly developed. It is supposed to offer a portable alternative to address the lack of existing opportunities in analyzing long jump performances in training. For this purpose, the drone should autonomously fly next

[1] around 10 m/s in male semi-professional long jump

to the athlete throughout the whole jump, capturing their motion and therefore allow for a complete jump analysis. The drone itself is based on First Person View (FPV) drone hardware. It is build from scratch using an onboard single-board computer as flight control unit responsible for capturing the video. Additionally, a ground station software is developed to allow for a convenient jump analysis regarding the overall body pose as well as a fixed set of important parameters, i.e. knee angles, arm angles, hip position. The project's source code is available under `https://github.com/JF631/FLYJUMP`.

# 2 Methodology

The following chapter provides an overview over the relevant development components that are used within this project. Therefore, the used software packages are introduced before a short outline of the utilized drone hardware is given.

## 2.1 Software fundamentals

As the main part of this project's software will run on a portable remote computer allowing for not only to control the drone but also for performing the long jump analysis on video inputs, every software component is chosen to demand as little hardware requirements as possible. Especially no Graphical Processing Unit (GPU) is required to run the software. All image processing is performed using the Central Processing Unit (CPU) only. Furthermore, the software is designed to run platform independent.

### 2.1.1 Programming Language and why Python

Because of its interpreted nature and many cross-platform libraries, Python is one of the most used programming languages in the scientific area. Furthermore, it offers a high level of abstraction allowing for rapid prototyping approaches which is a key factor for this project. Besides fast implementation, Python nevertheless supports complex programming concepts like object-orientation.
Additionally, as stated in subsection 2.1.2 many machine learning and AI projects for detecting body poses have already been successfully implemented using Python.
Third party libraries and frameworks like *OpenCV* for image processing and *PyQt* for Graphical User Interface (GUI) development are widely used and therefore well documented. This leads to the decision to use Python as programming language within this work.

### 2.1.2 Mediapipe for detecting body poses

One of the software's main tasks is to perform a human body pose detection on video inputs. Because this part runs on the remote computer only, it can also handle pre-recorded

videos that should be evaluated.
The evaluation itself is performed using the mediapipe framework. This framework uses a pre-trained convolutional neural network that is able to detect 33 key points in human body poses [4]. The network could theoretically even be fine-tuned to improve its accuracy on specific input types. Even if this so called *transfer-training* method requires significantly less training data than training a neural network from scratch, it is not applied within this project as first test runs already showed reliable results.
Furthermore, the mediapipe framework does not require any hardware acceleration and is renowned for its precise output. Hii et al. for example showed in [3], [5] that the framework can be applied in medical gait analysis applications to replace marker based approaches. Moreover, Mediapipe offers three different detection models that differ in terms of speed and accuracy. The fastest detection model offers the lowest accuracy and vice versa.
Additionally, Mediapipe is optimized for multiple input types including videos and live streams, which is ideal for this project.
Figure 2.1 shows an overview of the 33 detectable key points.

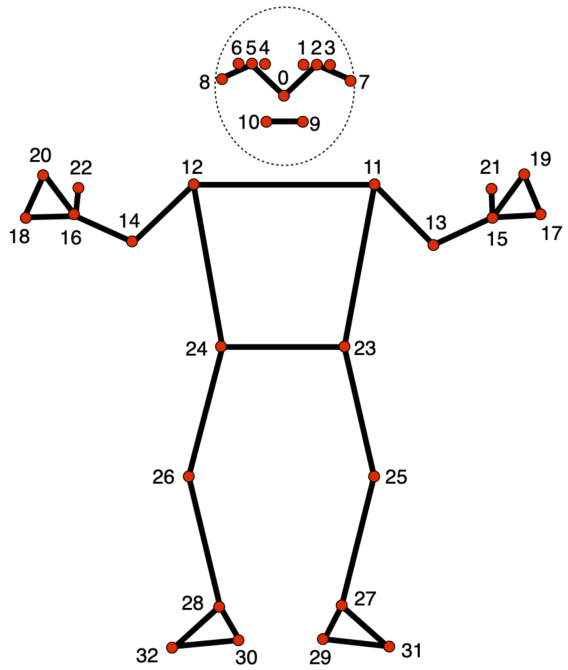


Figure 2.1: Fixed set of detectable body key points offered by the mediapipe framework [6]

Within this work, the key points in the head area (range [0 - 10]) are not of great interest apart from visualization purposes.
The knee, hip and arm key points however will be used for angle calculations and ground contact detection. Thus, a good performance in detecting the according key points within these areas is crucial for the software's overall reliability.

### 2.1.3 Why Mediapipe?

In recent years many approaches towards accurate body pose detection were developed and implemented. Many of those offer decent accuracies, but often lack reasonable performance, especially when no GPU is available for hardware acceleration. Following two common alternatives to Mediapipe, namely OpenPose and AlphaPose, are shortly presented and differentiated from the chosen Mediapipe framework.

One of the most widely used human pose detection libraries is the open source library *OpenPose*. As shown in [7] it offers a Multi-Person pose estimation that is especially useful when dealing with groups of people. However, as this project is meant to be used for long jump evaluation, only one person needs to be tracked at a time. Even though OpenPose of course can handle one person pose estimation, mediapipe outperforms OpenPose in this area. Back in 2016 Kocabas et al. achieved around 23 FPS using GPU accelerated OpenPose pose detection [8] and Osokin later proposed an improved neural network design, allowing for up to 28 FPS without hardware acceleration [9]. As of 2023 these benchmarks are still reasonable. Mediapipe however achieves speeds of up to 63% higher.

While OpenPose uses a *Bottom-Up* approach due its multi-person application, Mediapipe uses the less computational complex *Top-Down* approach.

Bottom-Up implementations first detect all body key points present in an input image and then move on to grouping the recognized points in clusters. Points in the same cluster are then assigned to one person.

Top-Down approaches however first roughly detect the overall body position within the input image and then define a region of interest[1] around the subject. The following processing therefore only needs to take this defined region of interest into account, leading to significantly less computational complexity.

AlphaPose is another open source library often used for body pose estimation. Just like OpenPose it uses a Bottom-Up approach to reliable offer multi-person body pose detection. Additionally, AlphaPose, like Mediapipe, offers multiple detection models that differ regarding accuracy and speed. Again however, Mediapipe outperforms AlphaPose because of its Top-Down approach and because AlphaPose is designed to work with GPU acceleration, rather than running on CPU only.

Another advantage of Mediapipe is its output. While OpenPose and AlphaPose offer 2D coordinates for each detected key point, Mediapipe additionally offers a depth estimation resulting in a spatial 3D coordinate for each detected key point. Thereby, more comprehensive analysis can be performed.

[1] sometimes also referred to as *Bounding Box*

### 2.1.4 OpenCV

### 2.1.5 HDF5 file format

To avoid analyzing the same jump, therefore the same video file, multiple times, the jump parameters should be saved after the analysis process alongside with the annotated video file, that visualizes the detected body pose (see Figure 2.1). Because parameters such as knee angles, arm angles, ground contact, etc. need to be stored frame-wise, a structured file format is suitable.
One common open source structured file format is HDF5. It is an acronym for Hierarchical Data Format (Version 5) and is very helpful to store large amounts of data as well as heterogeneous data. As the name already suggests, data is stored in a hierarchical, tree-like way.
A HDF5 tree mainly consists of three pre-defined components that allow to organize data in a file-system like fashion. The tree root, groups and datasets. *Groups* are folder-like structures that can either contain more groups or Datasets. *Datasets* hold the actual data that need to be stored.
Furthermore, each level (tree root, groups and datasets) can hold additional information via metadata. The metadata could for example contain information about metrics, timestamps or any other describing information. Therefore, each HDF5 file itself becomes a self-describing file that does not require any more than the included information to be interpreted correctly.

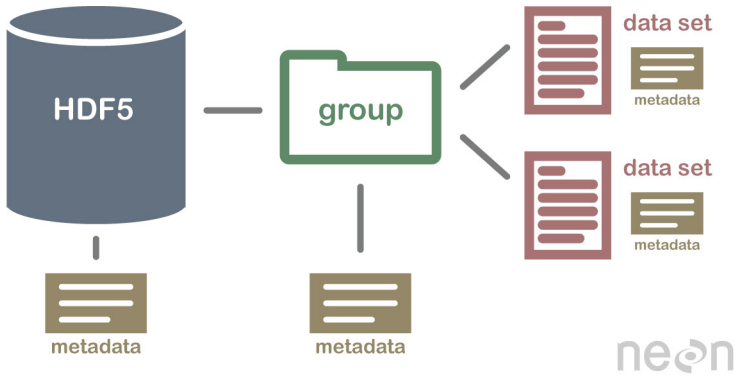


Figure 2.2: Principle HDF5 file structure [2]

Within this work, one HDF5 file will be created per jump analysis. The frames are stored in a group each and the actual calculated parameters are saved group-wise as datasets.

## 2.2 GUI development

The software that is developed within this work should be useable on-field to allow for a fast analysis process. Besides the video analysis, the drone control should be embedded in the same software to always guarantee control over the drone.
Thus, a simple GUI is developed to offer a convenient drone control and video analysis process.
The GUI is developed using Qt and its Python binding PyQt. Both are presented in this section.

### 2.2.1 Development framework Qt

Qt is a development framework based on the programming language C++. It includes a GUI toolkit and therefore enables platform-independent application development. All common platforms including the Linux, Windows and MacOS are supported by Qt. Additionally, both mobile operating systems, Android and iOS, are supported.
This project however will focus on the development of an application that is able to run under Windows, Linux and MacOS.
The Qt framework is mainly chosen because of its rich and comprehensive documentation[3] and the availability of the well-supported Python binding *PyQt*.
As Qt is based on C++ all Qt source files are translated to C++ code. This step is realized by the `Meta Obejet Compiler (MOC)` which is integrated as pre-processor. Thus, all Qt files are translated to so-called *Meta Obejet Code*, which can be seen as C++ source code with some enhancements. The most important enhancement is the signal and slot functionality which allows for an easy communication between different software and design elements (e.g. show a message dialog when a button is clicked).
Another important enhancement for this work is the convenient multi-threading management necessary for offering a responsive GUI even when cpu-bound calculations such as image processing is performed.

### The GUI module PyQt

Like explained earlier, Qt is based on the programming language C++. As justified in subsection 2.1.1 however, Python is chosen as programming language for this project. To combine the Python's machine learning advantages with Qt's platform-independent GUI development options, the module *PyQt5* [4] is used.
This module allows building Qt applications using Python instead of C++. Every other

[3] `https://doc.qt.io/`
[4] `https://pypi.org/project/PyQt5/`

described advantage Qt offers remains valid with the use of PyQt. Thus, the whole software within this project including the GUI can be developed using Python.

# Literatur


[1] A. Seyfarth et al. “Dynamics of the long jump”. In: *Journal of Biomechanics* 32.12 (1999), pp. 1259–1267. ISSN: 0021-9290. DOI: https://doi.org/10.1016/S0021-9290(99)00137-2. URL: https://www.sciencedirect.com/science/article/pii/S0021929099001372.

[2] Murray Evans et al. “Automatic high fidelity foot contact location and timing for elite sprinting”. In: *Machine Vision and Applications* 32.05 (2021). DOI: 10.1007/s00138-021-01236-z.

[3] Chang Soon Tony Hii et al. “Marker Free Gait Analysis using Pose Estimation Model”. In: *2022 IEEE 20th Student Conference on Research and Development (SCOReD)*. 2022, pp. 109–113. DOI: 10.1109/SCOReD57082.2022.9974096.

[4] Valentin Bazarevsky et al. “BlazePose: On-device Real-time Body Pose tracking”. In: *CoRR* abs/2006.10204 (2020). arXiv: 2006.10204. URL: https://arxiv.org/abs/2006.10204.

[5] Amit Gupta et al. “Knee Flexion/Extension Angle Measurement for Gait Analysis Using Machine Learning Solution “MediaPipe Pose” and Its Comparison with Kinovea®”. In: *IOP Conference Series: Materials Science and Engineering* 1279.1 (March 2023), p. 012004. DOI: 10.1088/1757-899X/1279/1/012004. URL: https://dx.doi.org/10.1088/1757-899X/1279/1/012004.

[6] Google Developers. *Pose landmark detection guide*. 2023. URL: https://developers.google.com/mediapipe/solutions/vision/pose_landmarker (visited on November 5, 2023).

[7] Zhe Cao et al. “Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields”. In: *CoRR* abs/1611.08050 (2016). arXiv: 1611.08050. URL: http://arxiv.org/abs/1611.08050.

[8] Muhammed Kocabas et al. “MultiPoseNet: Fast Multi-Person Pose Estimation using Pose Residual Network”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. September 2018.

[9] Daniil Osokin. “Real-time 2D Multi-Person Pose Estimation on CPU: Lightweight OpenPose”. In: *CoRR* abs/1811.12004 (2018). arXiv: 1811.12004. URL: http://arxiv.org/abs/1811.12004.