***Abstract*—Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut.**

***Index terms*—Body Pose Estimation, MediaPipe, Computer Vision, Motion Capture, 3D Pose Estimation, Demo Application**

# Contents

4216

# 1 Introduction

## 1.1 Body pose estimation

# 2 A brief overview of the State Of The Art

# 3 MediaPipe Pose Landmarker

The following section describes the MediaPipe Pose Landmarker solution [1]. It first provides some context about the broader MediaPipe Framework before going into more detail on the MediaPipe Pose Landmarker solution.

MediaPipe is a collection of on-device machine learning tools and solutions by Google [2]. It consists of two main categories. There are the MediaPipe solutions, which have predefined "Tasks" that are ready to be used in your application [3]. There are tasks on vision such as the detection and categorisation of objects or the detection of hand gestures [4], [5]. One of these vision solutions is the MediaPipe Pose Landmarker. Other solutions such as text classification and audio classification are also present [6], [7]. On the other hand exists the MediaPipe Framework. It is the low-level component used to build efficient on-device machine learning pipelines, similar to the premade MediaPipe Solutions [8]. The remainder of this thesis solely addresses the MediaPipe Pose Landmarker Solution from this point forward, and will frequently be referred to as simply "MediaPipe" or "MediaPipe Pose" for the sake of conciseness.

MediaPipe Pose is available on three different platforms. One can use it in Python, on Android and on the web. However, these are only just API's to interact with the actual detection task. The application presented in this thesis is completely written in Python but all the concepts that are discussed are applicable to any platform.

### 3.1.1 Features

The main feature of MediaPipe Pose is of course, just as all body pose estimation tools, to extract the body pose from a given image or video frame. Unlike many other body pose estimation tools, MediaPipe delivers a 3D estimation instead of the more common 2D estimation, by introducing depth. However, in the measurement results, this added depth dimensionality is shown to be less than ideal.

MediaPipe has three modes of operation, called the `RunningMode`. MediaPipe can work on still images (`IMAGE`), decoded video frames (`VIDEO`) and a live video feed (`LIVE_STREAM`) [1]. Using the live video feed mode has some implications. When running MediaPipe in a real-time setting, the inference time of the model is constraint by this real-time application. When frame inference takes too long to be in the time window the frame gets dropped. Another major aspect of real-time applications is that the inference should not block the main thread and halt the program. This is why the inference in the `LIVE_STREAM` mode is performed asynchronously and results are propagated back using a callback function.

One other feature other than returning the body pose is the creation of an image segmentation mask. MediaPipe has the ability to output a segmentation mask of the detected

body pose. This mask could be used for e.g. applying some visual effects and post-processing, but it is not of much use in this implementation.

### 3.1.2 Inference models

MediaPipe Pose relies on two models for the inference of the body pose. The first one is BlazePose and is only designed to return two-dimensional data in the given frame [9]. A synthetic depth is obtained via the GHUM model fitted to the 2D points [10].

#### 3.1.2.1 BlazePose

BlazePose provides human pose tracking by employing machine learning (ML) to infer 33, 2D landmarks of a body from a single frame [9]. A standard for body pose originating from 2020 is the COCO topology [11]. The COCO 2020 Keypoint Detection Task is a challenge to develop a solution to accurately detect and locate the keypoints from the COCO dataset. The original COCO topology only consists of 17 landmarks. With little to no landmarks on the hands and feet. BlazePose extends this topology to a total of 33 landmarks by providing landmarks for the hands and feet as well. These added landmarks are crucial for the drumming application and is part of the reason MediaPipe was chosen.

Next some parts of the BlazePose design and tracking model are discussed. It is important to understand some underlying methods discussed here as it helps to interpret the measurement results. A complete and detailed overview is provided in the original paper [9].

The pipeline of MediaPipe is displayed in Figure 2. Before predicting the exact location of all keypoints the pose region-of-interest (ROI) is located within the frame using the *Pose detector*. The *Pose tracker* then subsequently infers all 33 landmarks from this ROI. When

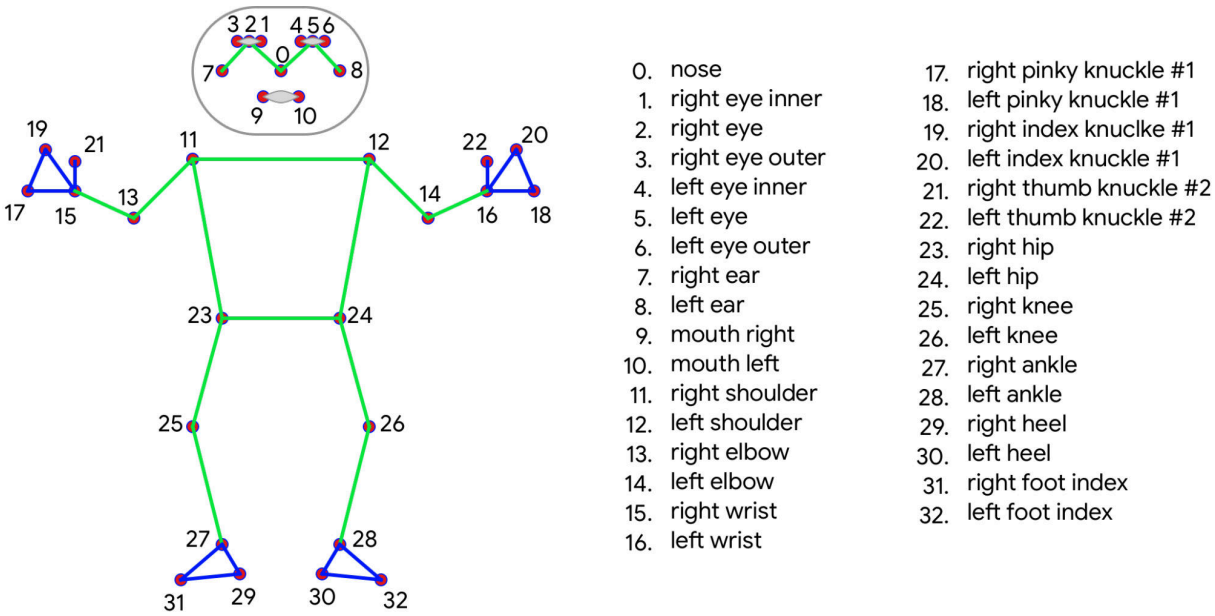


*Figure 1: BlazePose 33 landmark topology with the original COCO landmarks in green.*

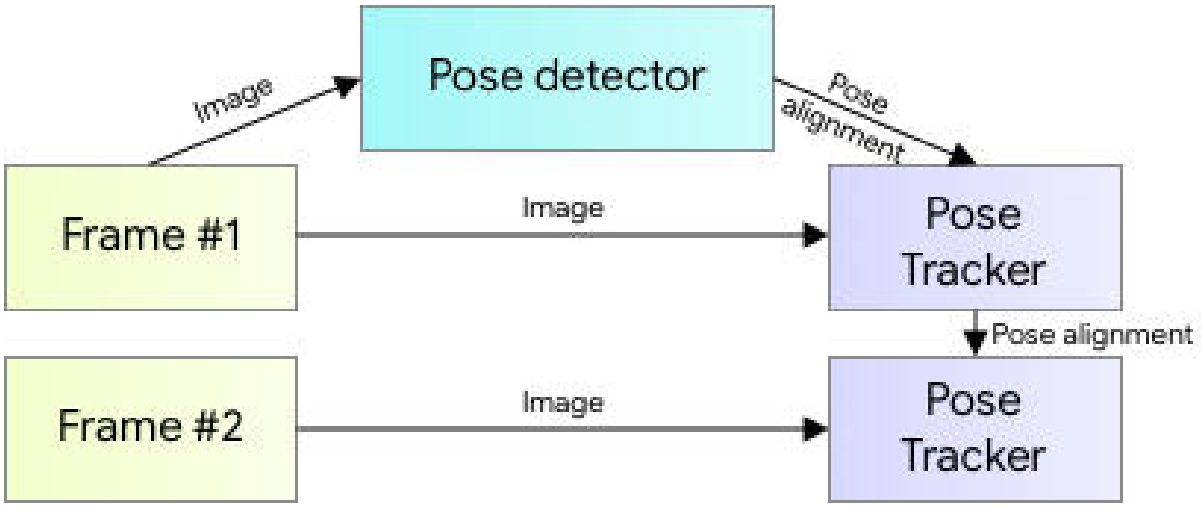


*Figure 2: MediaPipe Pose estimation pipeline*

in `VIDEO` or `LIVE_STREAM` mode, the pose detection step is only run on the first frame. For any subsequent frames, the ROI is than deriver from the previous points.

As this solution is intended to be used in a real-time setting, the inference needs to be within milliseconds. The MediaPipe team have observed that the position of the torso is most easily and efficiently found by detecting the position of the face. The face has the most high-contrast features of the entire body and thus results in a fast and lightweight inference compared to the rest of the body [12]. This of course has the logical consequence that the face should be visible within the frame. This means that the “drummer” has to face the camera head on for the best result.

After detecting the face the construction of the pose region-of-interest begins. The human body centre, scale, and rotation are described using a circle. This circle is constructed by predicting two virtual points. One point being the centre of the hips, the other lies on the circle as the extension of the hip midpoint to face bounding box vector. These two points should now describe a circle as shown in Figure 3.

Lastly, we very briefly describe the actual network architecture for the tracking of keypoints. The model takes an image as input with a fixed size of 256 by 256 pixels and 3 values for each pixel providing RGB values, Figure 4. The model uses “a regression approach that is supervised by a combined heat map/offset prediction of all keypoints” [9], [13]. The left-hand side outputs the heat maps and offset maps. Both the centre and left-hand side of the network are trained using the provided heatmap and offset loss. Afterwards the heatmaps and output maps output layers are removed and the regression encoder on the right-hand side is trained. This shows that there is no use in increasing the resolution in order to get an increase in the prediction accuracy. The measurements also confirm this hypothesis.

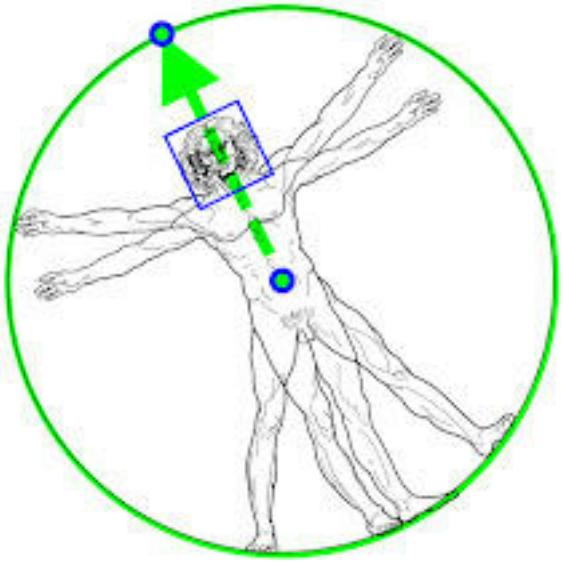

*Figure 3: MediaPipe Pose detection and alignment using virtual keypoints.*

#### 3.1.2.2 GHUM (Generative 3D Human Shape and Articulated Pose Models)

As mentioned, MediaPipe Pose offers an extra dimension unlike many other purely 2D pose estimation tools. This third dimension is of course depth. It does so by utilising an entirely different model being the GHUM model [10]. The GHUM model is able to construct a full 3D body mesh given image scans of a person. The outputs form an actual 3D mesh of 10,168 vertices for the regular model and 3,194 vertices for the lite model. MediaPipe is quite vague on the exact usage of this model in the pose solution. The only mention of GHUM is in the following sentence: *"Keypoint Z-value estimate is provided using synthetic data, obtained via the GHUM model (articulated 3D human shape model) fitted to 2D*

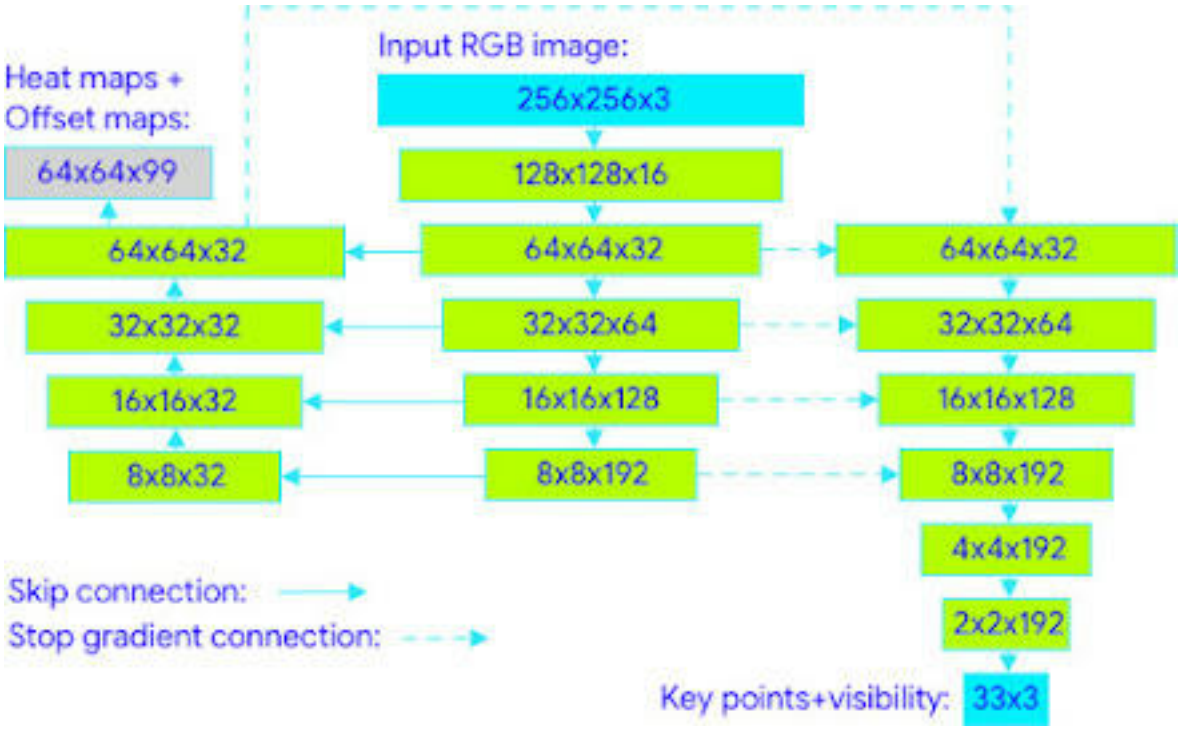


*Figure 4: MediaPipe Pose network architecture.*

*point projections."* [14]. Nonetheless, in the measurements it is shown that there is some notion of depth, but it is far from accurate. Especially when compared to the other 2D values provided by the BlazePose model.

# 4 Measuring accuracy and deviation

The following chapter discusses the accuracy and deviation of a body pose estimation tool. More specifically, the outputs of the MediaPipe Pose landmark detection solution are compared to that of a traditional optical tracking system. All base measurements are taken from a Qualisys Oqus MRI[1] system setup at the Art and Science Interaction Lab at De Krook in Ghent, provided by IDLab-Media[2].

## 4.1 Methods

To be able to get indications of the accuracy of MediaPipe Pose, a baseline, or base "truth", needed to be established. Using the aforementioned Qualisys MoCap setup at ASIL we are able to track marker locations at sub-millimeter accuracy, which are then taken as ground truth.

### 4.1.1 Measurement Setup

The setup is configured to be in line with the expected environment the drum application will be used in. Being, a single person sat on a chair facing a webcam, performing drumming motions. The person performing the movements has infrared reflective trackers positioned on the body as close as possible to where the MediaPipe Pose landmarks are located. This setup is shown in Figure 5.

The Qualisys Motion Capture setup first requires a calibration step, after which markers can be tracked with a precision of up to 0.3 mm. In later measurements, the accuracy has dropped to 0.8 mm, which can be attributed to calibrating a larger volume of space. For our use case, this level of accuracy is still quite good. As we will see, the MediaPipe accuracy will not get close to an accuracy of 0.8 mm, so the Qualisys accuracy is more than sufficient. After applying the reflective markers, motion capture can easily be performed using the Qualisys Track Manager (QTM) program[3]. After having labeled each marker and trajectory, these can be exported to a TSV (tab separated values) file [15]. One such a file looks like the snippet below, Listing 1. First, some context is provided on the measurement. After the listing of the marker names and their trajectory types, follows the actual trajectory. Every line in the file then describes the 3D position of every marker per frame. Every three floating point numbers are the x, y and z coordinates of a given marker, in the same order as they are listed.

[1] The Oqus MRI is one of Qualisys' traditional optical motion capture cameras, requiring physical trackers on the body that reflect incoming infrared light from the camera. *https://www.qualisys.com/cameras/oqus-mri/*

[2] The Art and Science Lab is a " *highly modular research infrastructure aimed at interaction research*", provided by IDLab-Media, one of the research teams within the research group IDLab from Ghent University and imec. *https://media.idlab.ugent.be/about/*

[3] *https://www.qualisys.com/software/qualisys-track-manager/*

*Figure 5: A frame from the camera recording showing the measurement setup.*

The video recordings are captured with a regular webcam with a maximum resolution of 1080p and a frame rate of 30 frames per second. The camera placement needs to be in line with the forward facing axis of the Qualisys captures to reduce any deviations introduced by being off-axis. Afterward, the videos get processed frame by frame with MediaPipe, using the *VIDEO* running mode[4]. The video mode is an offline processing mode, meaning that no frames will be dropped to satisfy a live stream constraint. This mode allows us to

```
1  NO_OF_FRAMES  6916                                                    TSV
2  NO_OF_CAMERAS 13
3  NO_OF_MARKERS 10
4  FREQUENCY 120
5  NO_OF_ANALOG  0
6  ANALOG_FREQUENCY  0
7  DESCRIPTION --
8  TIME_STAMP  2024-04-15, 10:39:38.777  406646.66137317
9  DATA_INCLUDED 3D
10 MARKER_NAMES  Wrist_L Hip_L Foot_Index_L
11 TRAJECTORY_TYPES  Measured  Measured  Measured
12 -473.521  191.980 613.382 -677.077  164.814 615.497 -138.493  187.190 53.579
13 -473.564  192.348 613.040 -677.061  164.822 615.494 -138.494  187.229 53.555
14 -473.646  192.696 612.779 -677.045  164.794 615.445 -138.465  187.193 53.464
15 -473.683  192.890 612.513 -676.988  164.782 615.436 -138.448  187.252 53.482
```

*Listing 1: A snippet of what a TSV output from a Qualisys tracked measurement looks like.*

[4] *https://developers.google.com/mediapipe/solutions/vision/pose_landmarker/python#video*

```csv
frame,time,index,x,y,z,visibility,presence,landmark_type
2,53,0,0.4807698130607605,0.22160261869430542,-0.360921174287796,0.99999964237213
2,53,1,0.4906315207481384,0.20818790793418884,-0.33244213461875916,0.999999165534
2,53,2,0.4961097836494446,0.2083108127117157,-0.33247220516204834,0.9999989271163
2,53,3,0.49966344237327576,0.20876441895961761,-0.33247825503349304,0.99999880790
2,53,4,0.4742940366268158,0.2077968716621399,-0.3331023156642914,0.99999892711639
2,53,5,0.4689987897872925,0.2078893780708313,-0.3331734538078308,0.99999892711639
2,53,6,0.4639938175678253,0.20820003747940063,-0.3331492245197296,0.9999990463256
2,53,7,0.5079091787338257,0.21490448713302612,-0.17037953436374664,0.999997854232
2,53,8,0.46058356761932373,0.21451136469841003,-0.17510229349136353,0.99999868869
2,53,9,0.49152788519859314,0.23765860497951508,-0.30100592970848083,0.99999976158
```

*Listing 2: A snippet of the CSV output taken from MediaPipe processed frames. Lines are truncated.*

measure models that we otherwise would not be able to run at a constant 30 frames per second. The result of every processed frame is then written to a CSV file. Every line in the file contains all the known information of one marker at the given frame. Resulting in the following format, displayed in Listing 2: *frame number*, *time* (in milliseconds), *index* (of the marker), *x*, *y*, *z*, *visibility*, *presence*, *landmark_type* (either 0 to indicate a `Landmark`, or a 1 to indicate a `WorldLandmark`).

### 4.1.2 Deviation from one time series to another

Before describing how the outputs from the previous section are used to come up with accuracy measurements, a brief overview of how deviation from one time series to another is derived, is needed. We want to compare the trajectories from the Qualisys capture with that of the MediaPipe captures. These trajectories consist of discrete-time points. As the frame rate of both systems differs, the discrete-time points are unaligned.

Figure 6 provides a visual representation of the problem. We have two time series to compare. Series $f$ in red and series $g$ in blue. Due to these points having a different frequency and unaligned time stamps, it is not possible to compare these just by iterating over both series at the same time. Where an iteration corresponds to jumping to the next point in time in both series.

One option would be to "walk" over the series based on the time difference between points. The walk starts by selecting the first point of each series, $A$ and $K$ in this example. Then there are three options:

1. Only jump to the next point in the first series
2. Only jump to the next point in the second series
3. Jump to the next point in both series

The option that minimizes the time difference between the selected points is chosen. This method ensures that every point in both series is taken into account. The pairs of points

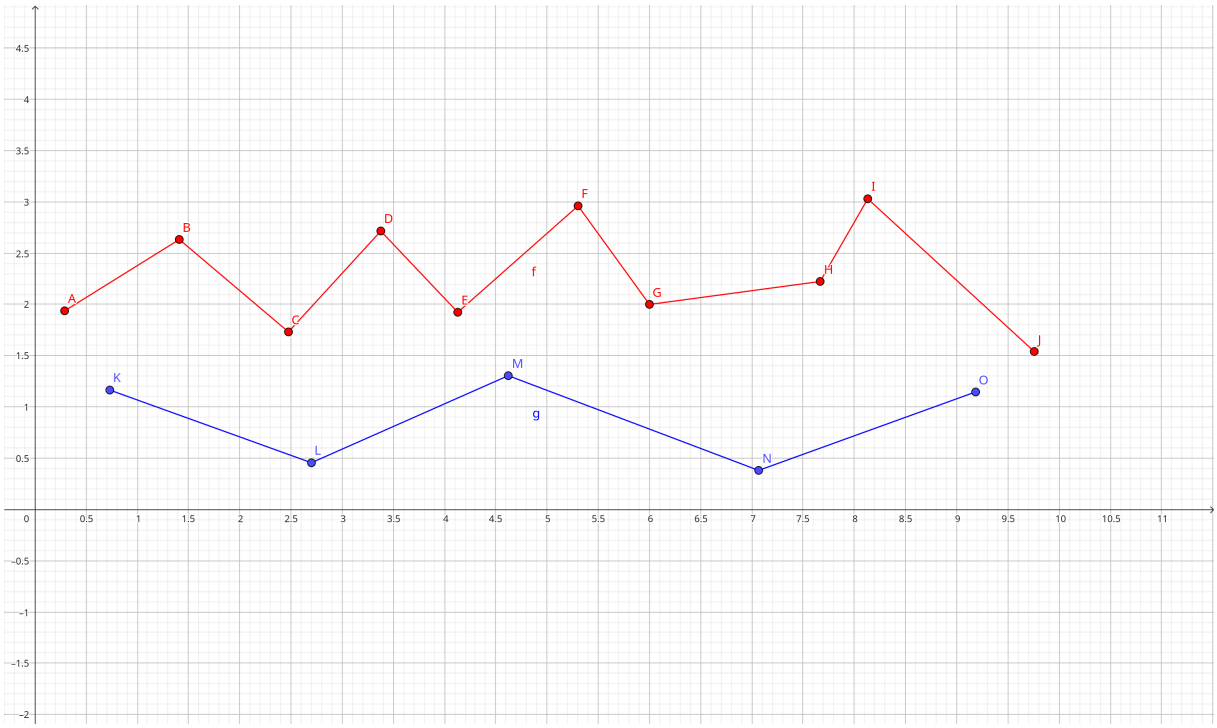


*Figure 6: Example of unaligned time series, the horizontal axis being time, the vertical axis would be some value*

that would be selected by this technique is displayed in Figure 7 by the dotted lines. The measured difference between both signals is then the sum of the length of all dotted lines (the difference between point pairs) divided by the amount of dotted lines (pairs).

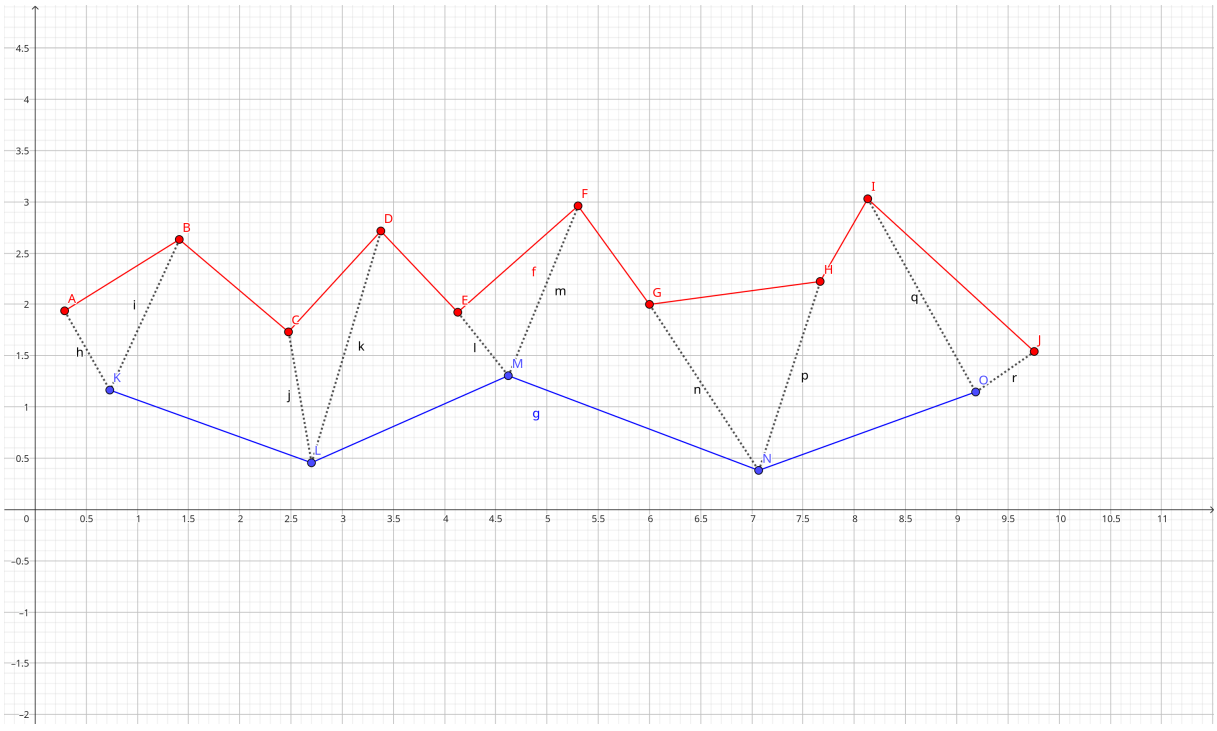


*Figure 7: Point pairs selected by walking over both series indicated by the dotted lines.*

However, it is clear to see that this method will result in a higher deviation when comparing a high frequency signal against a lower frequency signal. Which is the case in our measurements. A high frequency Qualisys measurement (100Hz or more) is compared against a lower frequency MediaPipe measurement (30Hz). Because we know these differences in frequency, we are not interested in a difference method that takes these into account. The goal of the measurements is to find the average deviation of a point generated by MediaPipe with that of the corresponding point generated by the Qualisys capture. The following new method achieves that goal.

Instead of comparing every point in both series, a dominant series is selected. This simple method iterates over the dominant series and for each point in the dominant series finds the corresponding point in the other series. The corresponding point being the point with the timestamp closest to that of the dominant point. Adapting the behavior of the "walk" to accommodate this idea of having a dominant time series, would result in the following procedure:
For each point in the dominant time series, walk over the points of the other series until the difference in time stamps no longer decreases. Then we have found a new pair. Results of this procedure are displayed in Figure 8.

With this method, we have that the irrelevant points are no longer included in the measurements. Yet this does not mean that the points that are now taken into account are perfectly aligned with one another. However, due to the high frequency of the Qualisys

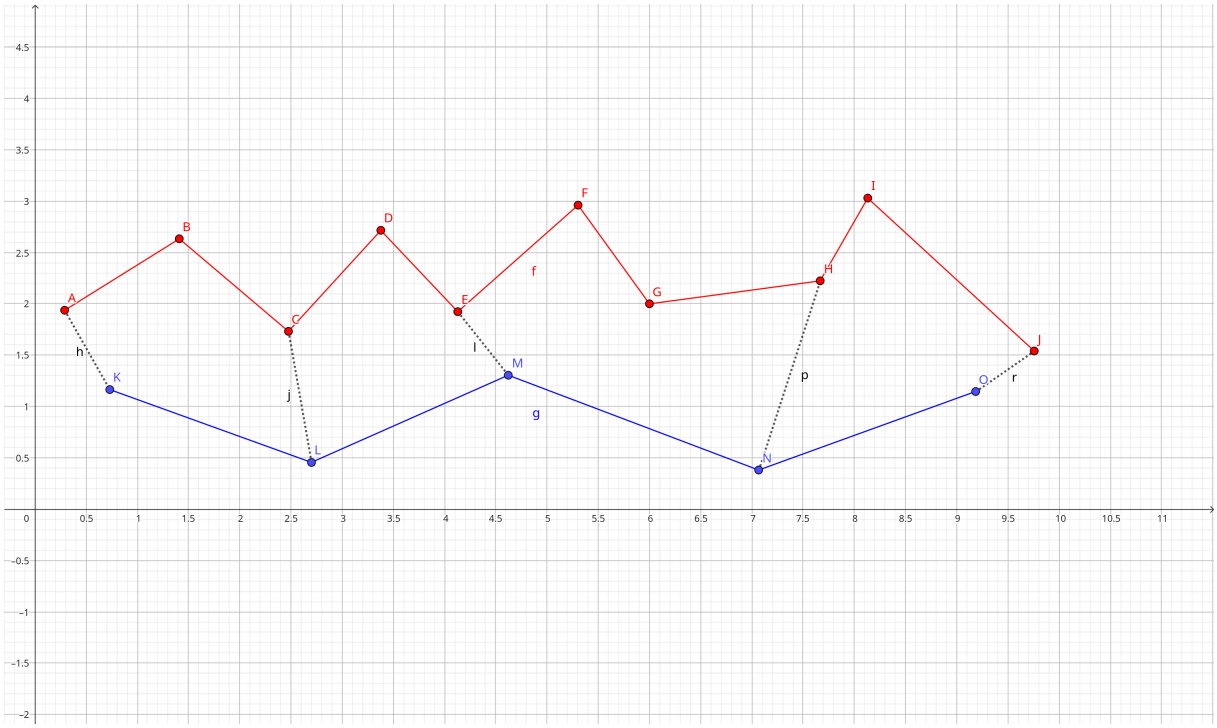


*Figure 8: Point pairs selected by iterating over the dominant series ($g$, blue) and walking over the other ($f$, red), indicated by the dotted lines.*

captures, we note that the time difference will be relatively small, and thus the difference induced by this nonalignment will be quite small. For a Qualisys capture with a frame rate of 120Hz, there is a frame every $\frac{1000\text{ ms}}{120\text{ Hz}} = 8.33...$ milliseconds. In the worst case, the MediaPipe point lies exactly between two Qualisys frames, resulting in a maximum time difference of only 4.166... milliseconds.

Two simple metrics can now be computed having found a way to get point pairs. A first one is the average offset of one signal to another. By simply taking the average difference of each point pair. A second and very similar metric is the average deviation. It denotes how much a signal deviates from another one. Instead of taking the regular difference of a point pair, the absolute difference is taken. This is an important distinction. The average offset is simply a metric that tells us how far a signal is from another. The average deviation, however, can be interpreted as the accuracy with which a signal approximates another signal. So when the Qualisys signal is seen as the base truth and the MediaPipe signal is seen as an approximation of that base truth, we have an accuracy measure for the MediaPipe signal!

After this slight detour, follow the methods used to align the MediaPipe results to the Qualisys captures.

### 4.1.3 Aligning MediaPipe to Qualisys

Using the setup described in Section 4.1.1 we have two time series that can be compared using the method from the previous section. However, before that is possible there is still one major issue that needs to be solved. The points from the MediaPipe result originate from a totally different axis and origin point. The Qualisys captures have their origin point calibrated on the floor, with the x-axis being the forward facing axis, the y-axis the horizontal axis and the z-axis the vertical axis. The MediaPipe Landmarks, on the other hand, do not actually correspond with a point in space but rather with a point in the video frame and an associated depth. For the `Landmark` markers the origin point in this MediaPipe case is the left corner of the video frame. The x-axis is the horizontal axis along the frame, the y-axis is the vertical axis along the frame, and the z-axis is the depth from the camera. MediaPipe also has a different kind of Landmarks (`Landmark`), namely the World Landmarks (`WorldLandmark`). These try to map the regular Landmarks, which are a point in the video frame, to a point in space. With the center of the hips taken as origin. The axis remain the same but are scaled so that the `WorldLandmarks` are in line with the actual size of movement in the space.

This section gives a complete overview on how the MediaPipe signal has been aligned to match the Qualisys signal and provide a proper measurement on accuracy. The entire section uses a measurement of the vertical position of the left wrist marker, taken from

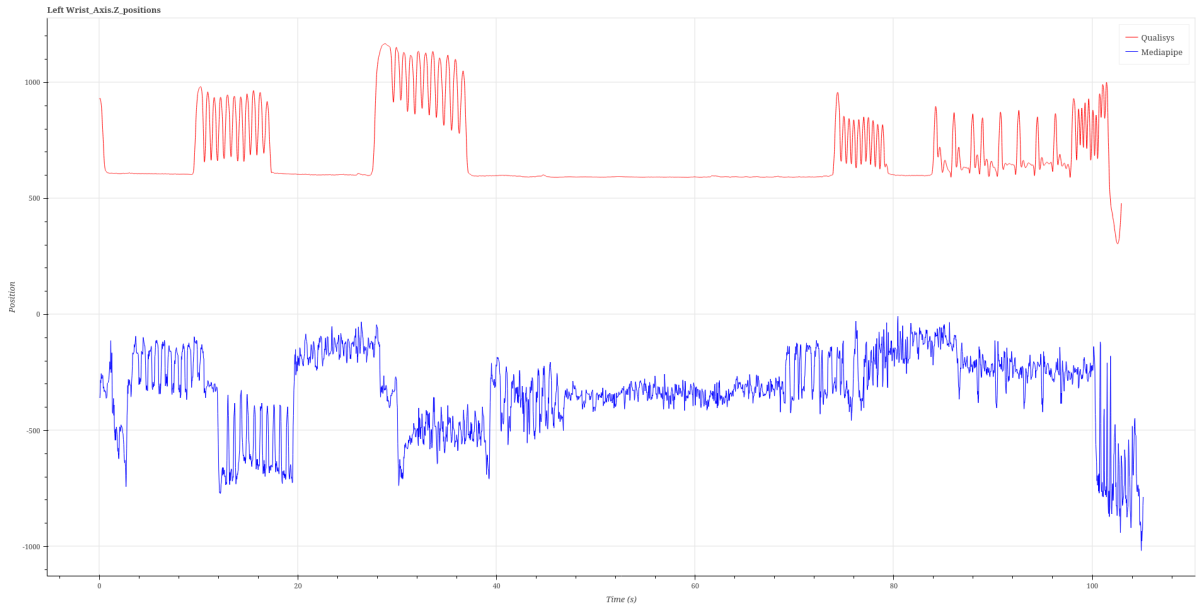


*Figure 9: Plot of the MediaPipe (Blue) and Qualisys (Red) left wrist z-axis without processing.*

MediaPipe using regular `Landmarks` and the `FULL` model. The plots in this section can be consulted in enlarged form in the appendices, Section 7.1.

The output from the recordings are time series that can be plotted. The output from the measurements is read and without any processing plotted to a line plot in Figure 9. On the horizontal axis is time in seconds. On the vertical axis is the value of the point in time in millimeter. Figure 9 shows a clear mismatch in axis. Plotted is the z-axis from both capture systems. But as mentioned, in MediaPipe the z-axis is the depth and not the vertical axis.

The problem of mismatched axes is a very simple one to solve. Before plotting the MediaPipe signal we switch the axis so they match with the actual direction of axis in the Qualisys recording. The MediaPipe axes are thus mapped as follows:

- $x \rightarrow y$
- $y \rightarrow z$
- $z \rightarrow x$

As can be seen in Figure 10, we now have plotted the proper vertical axis. One might have noted the pretty nonsensical values of the MediaPipe signal. For one, they are negative. Whereas moving up corresponds to an increase in value with the Qualisys captures, the inverse is true for the MediaPipe results. This requires us to re-invert the vertical values during analysis, resulting in negative values. Normally `Landmark` values would be in the range [0, 1], 0 being one side of the video frame, 1 the other side. But these values are inverted by multiplying by -1 resulting in a new range of [0, -1]. Because this inverting method should also work for `WorldLandmarks` which have no predefined range of values it is not possible to invert the values using the following method: $x \Rightarrow 1 - x$.

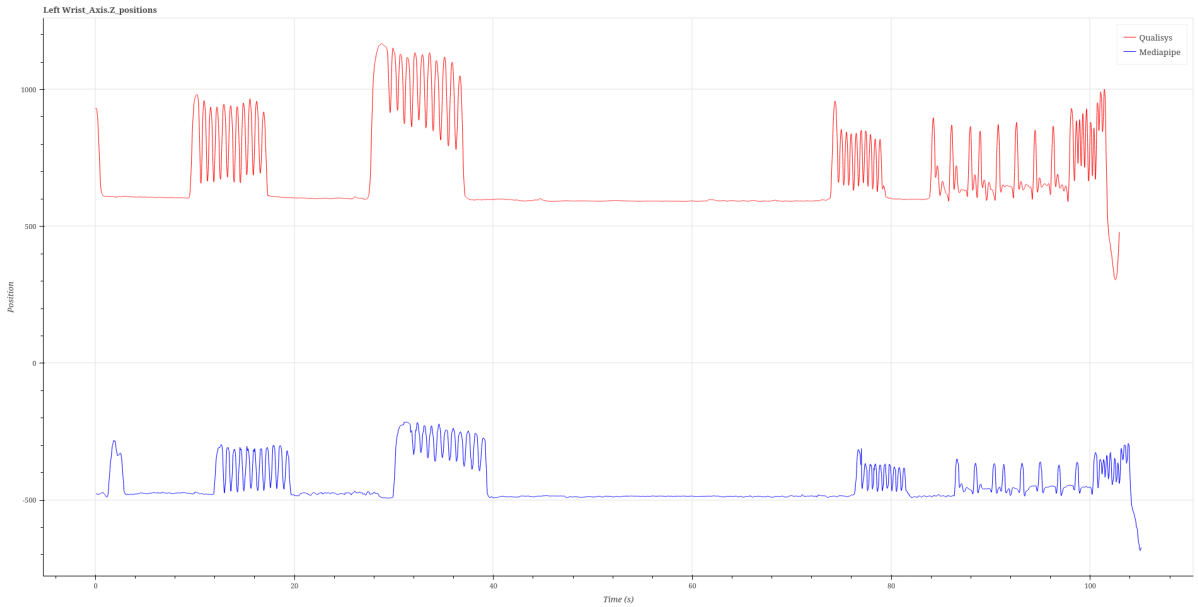


*Figure 10: Plot of MediaPipe (Blue) and Qualisys (Red) left wrist z-axis after re-arranging the MediaPipe axes.*

Secondly, all MediaPipe values have been multiplied by 1000. As the Qualisys output is in millimeter we already prepare the MediaPipe signal by interpreting the incoming signal as meter and converting it to millimeter. The MediaPipe `Landmark` signal has no real unit of course but interpreting it as meters allows for a simpler interpretation when it comes to scaling, explained later in this section.

Now that the basics are out of the way we can start aligning the signal. A first step is removing the average offset the MediaPipe signal has to the Qualisys signal. The method of walking over the dominant series (MediaPipe in this case) and gathering pairs of points from both series as discussed in the previous section, Section 4.1.2, is used for this. For every pair of points we can simply take the difference between those points. The average of these differences is then the offset of the MediaPipe signal. The result of removing this offset from the MediaPipe signal is displayed in Figure 11.

With the two signals close together another problem becomes apparent. They are offset in time. This makes sense as both measurements cannot easily be started at exactly the same time. We need to introduce a starting offset. This starting offset should minimize the deviation between both signals. This is achieved by iteratively increasing a starting offset and capturing the offset that resulted in the least deviation. In Figure 12 it is shown that this method finds the most perfect offset. Both signals are perfectly aligned in time. After this offset operation the average vertical offset is computed again and subtracted from the signal.

The final and most intricate part of the alignment is getting the scaling right. As we can see in the previous plots the scale of the signal is not at all correct. Here a scaling factor

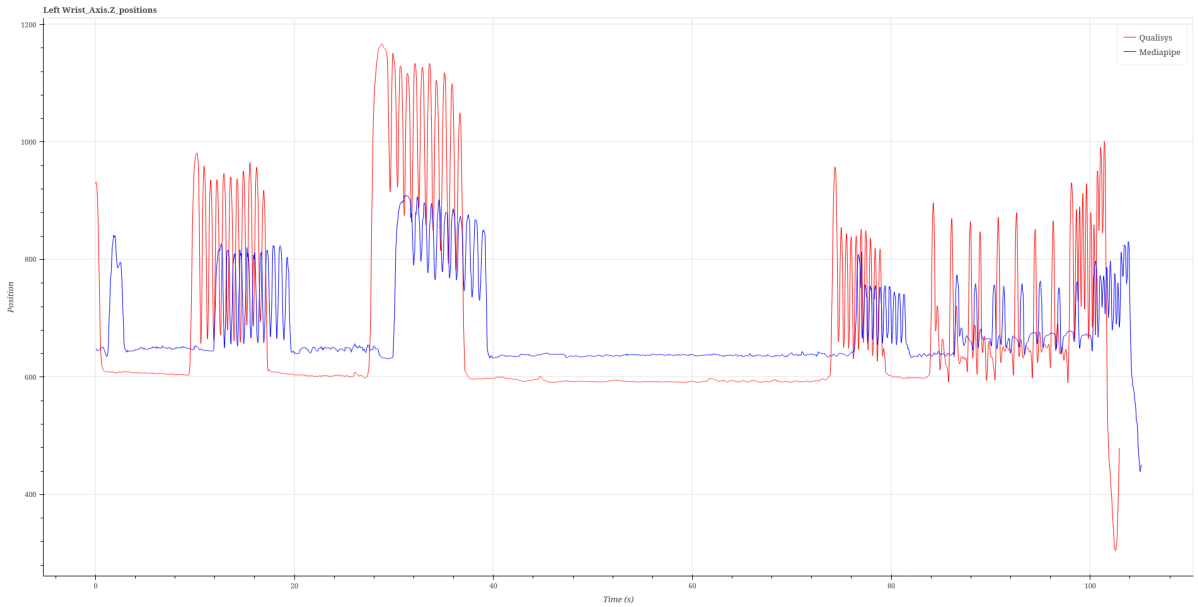


*Figure 11: Plot of the MediaPipe (Blue) and Qualisys (Red) left wrist z-axis with the average offset removed.*

needs to be found that minimizes the deviation. There is one caveat, we cannot simply scale the signal by multiplying it with a given factor. This would scale the signal away from the origin point. One can see that, in fact, we should "stretch" the signal vertically to make it align. In other words, the signal needs to be scaled around the center point of the signal that it is being aligned to.

The center point of the signal is easily calculated as the average of the signal's values. The stretching of the signal then goes as follows: For every original point of the signal, take the difference between that point and the center point of the other signal. Scale the

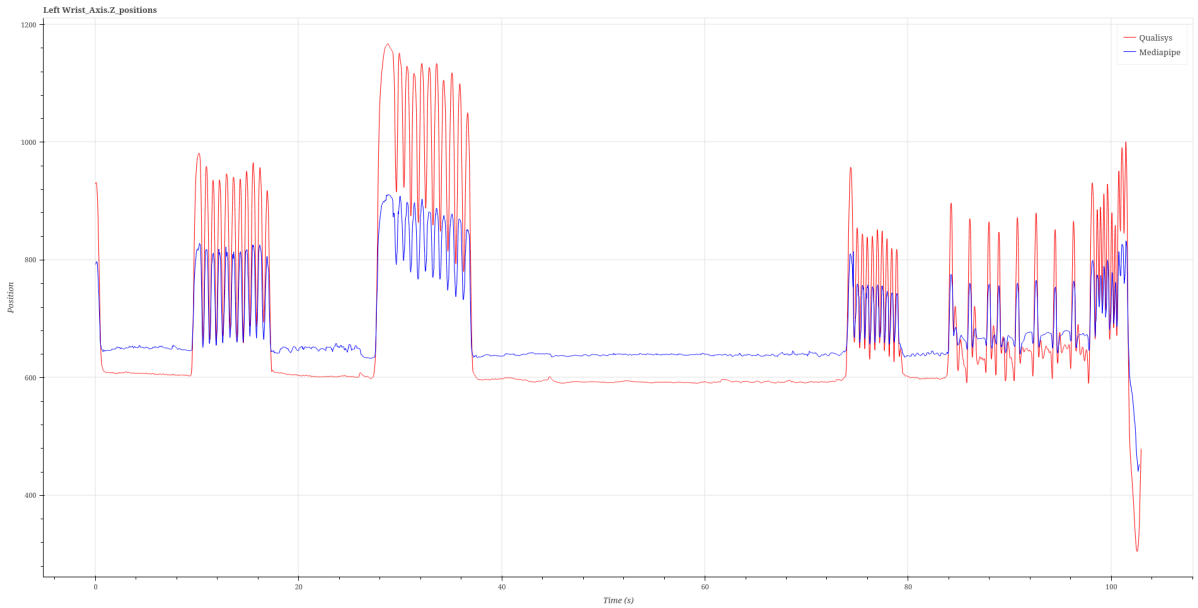


*Figure 12: Plot of the MediaPipe (Blue) and Qualisys (Red) left wrist z-axis with the time offset removed.*

difference by the scaling factor. The new stretched point is now the center point plus the scaled difference.

The previous method of finding the optimal time offset is a simple one. Since it is a discrete problem, the optimal value can easily be found by testing all possible values. The optimal scale, however, is not a discrete value. To find the optimal scaling factor we need an optimization algorithm.

The optimization problem at hand can be solved using Golden-section search [16]. It is a technique for finding an extremum (minimum or maximum) of a function inside a specified interval [17]. Which in our case the function is the deviation function that takes in the scale factor and outputs the deviation after applying the scale. The algorithm converges to one extremum by narrowing down an interval of possible values. Without going into too much detail on the algorithm and its implementation, the algorithm is initialized to search within a range of [0, 10] as possible scale factors and stops when the improvements in deviation fall below 0.01 mm. Applying the Golden-section search method on our running example returns a scale factor of around 2 and results in a nice alignment between both signals (Figure 13). The factor of 2 also makes sense. In the `Landmark` mode, the range of values lie between 0 and 1, reaching these outer values at the edges of the frame. As mentioned, the `Landmark` signal is interpreted to be in meter. As a consequence, the scaling factor is not only a scaling factor, it has become a measurement of the dimensions of what is visible in the frame. This means that the visible height in the video frame is 2 meters, at the location of the test subject, of course.

## 4.2 Results

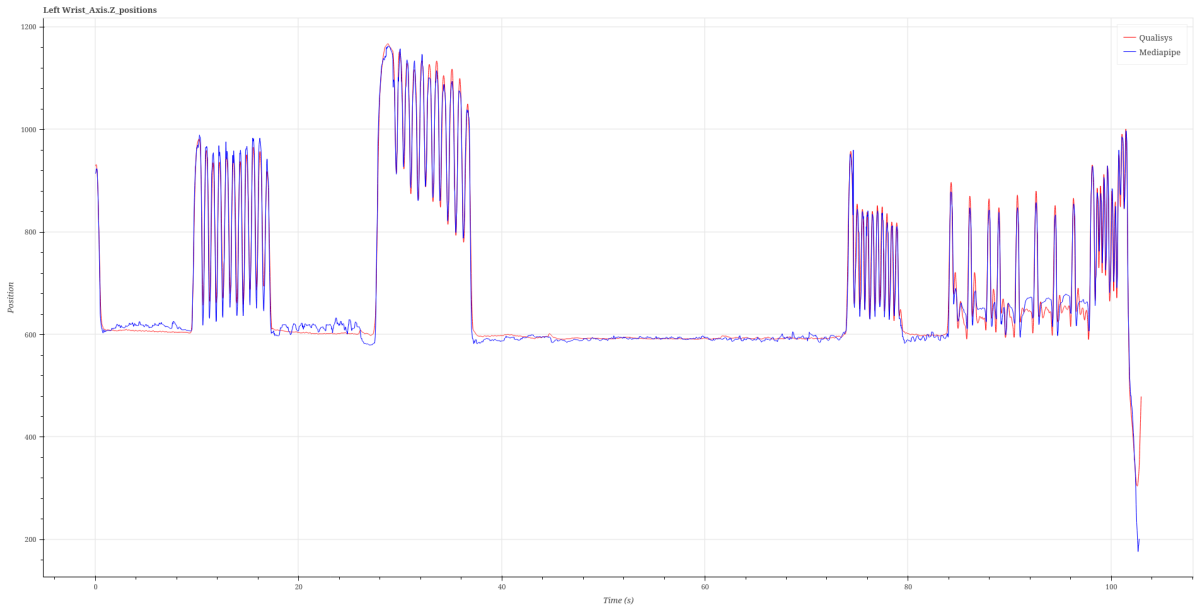


*Figure 13: Plot of the MediaPipe (Blue) and Qualisys (Red) left wrist z-axis with the MediaPipe signal scaled to match.*

# 5 Improving accuracy and reducing jitter

## 5.1 Methods

## 5.2 Results

# 6 Drum Application

# 7 Appendices

# List of Figures



## 7.1 Figures

Figures that are either not included in the text or added in an enlarged form.

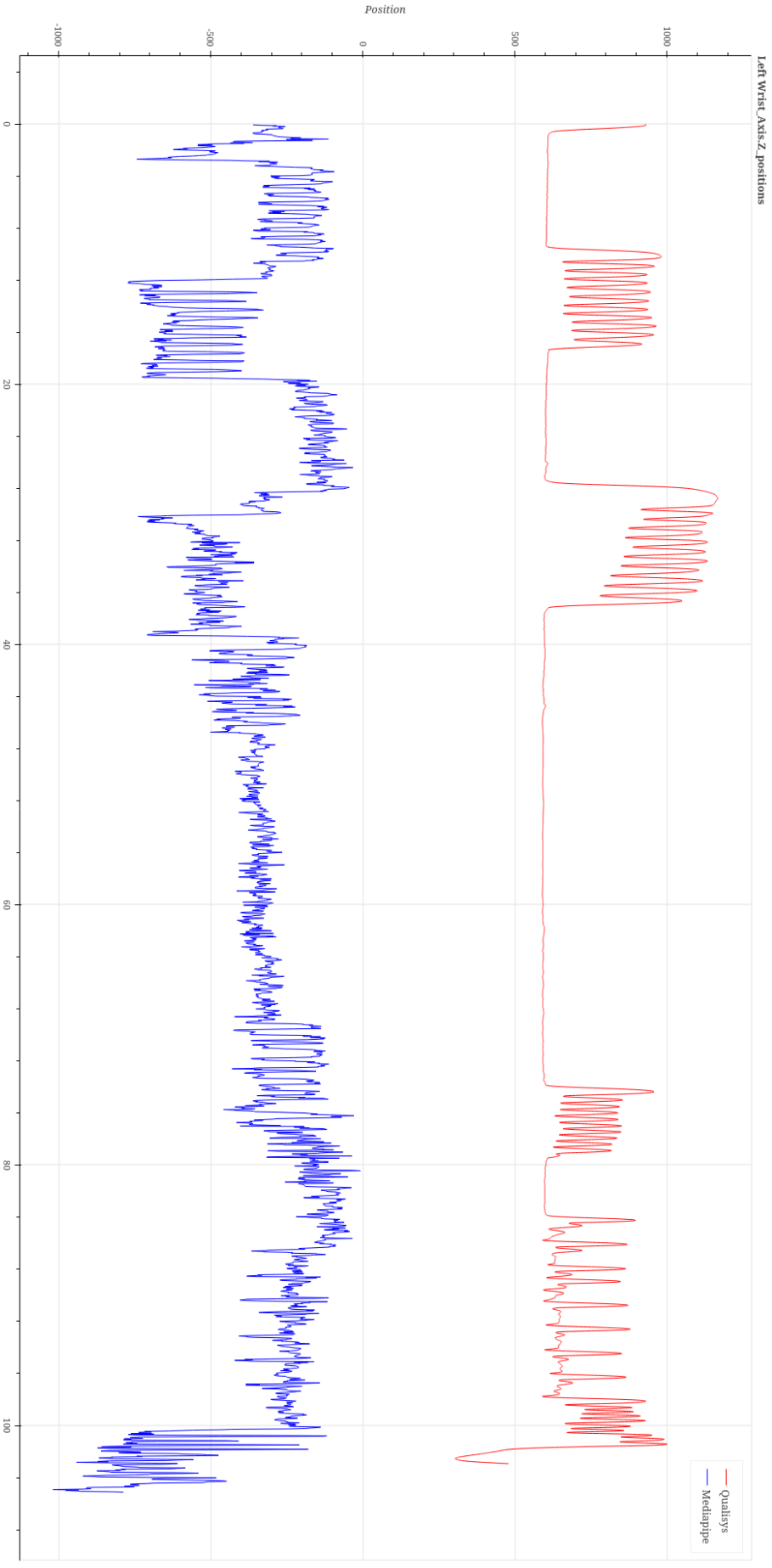


*Figure 7.1: Plot of the MediaPipe (Blue) and Qualisys (Red) left wrist z-axis without processing.*

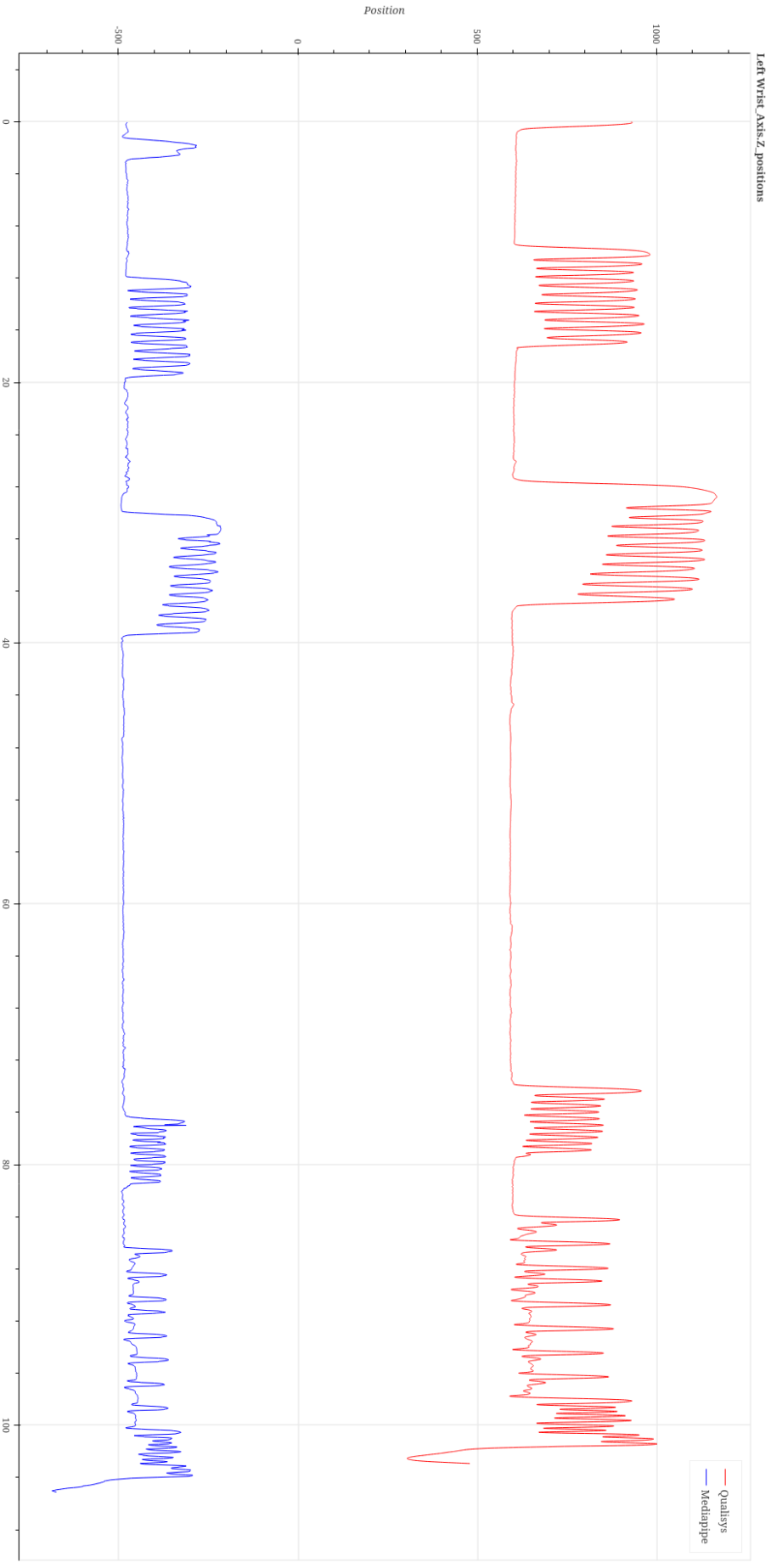


*Figure 7.2: Plot of MediaPipe (Blue) and Qualisys (Red) left wrist z-axis after re-arranging the MediaPipe axes.*

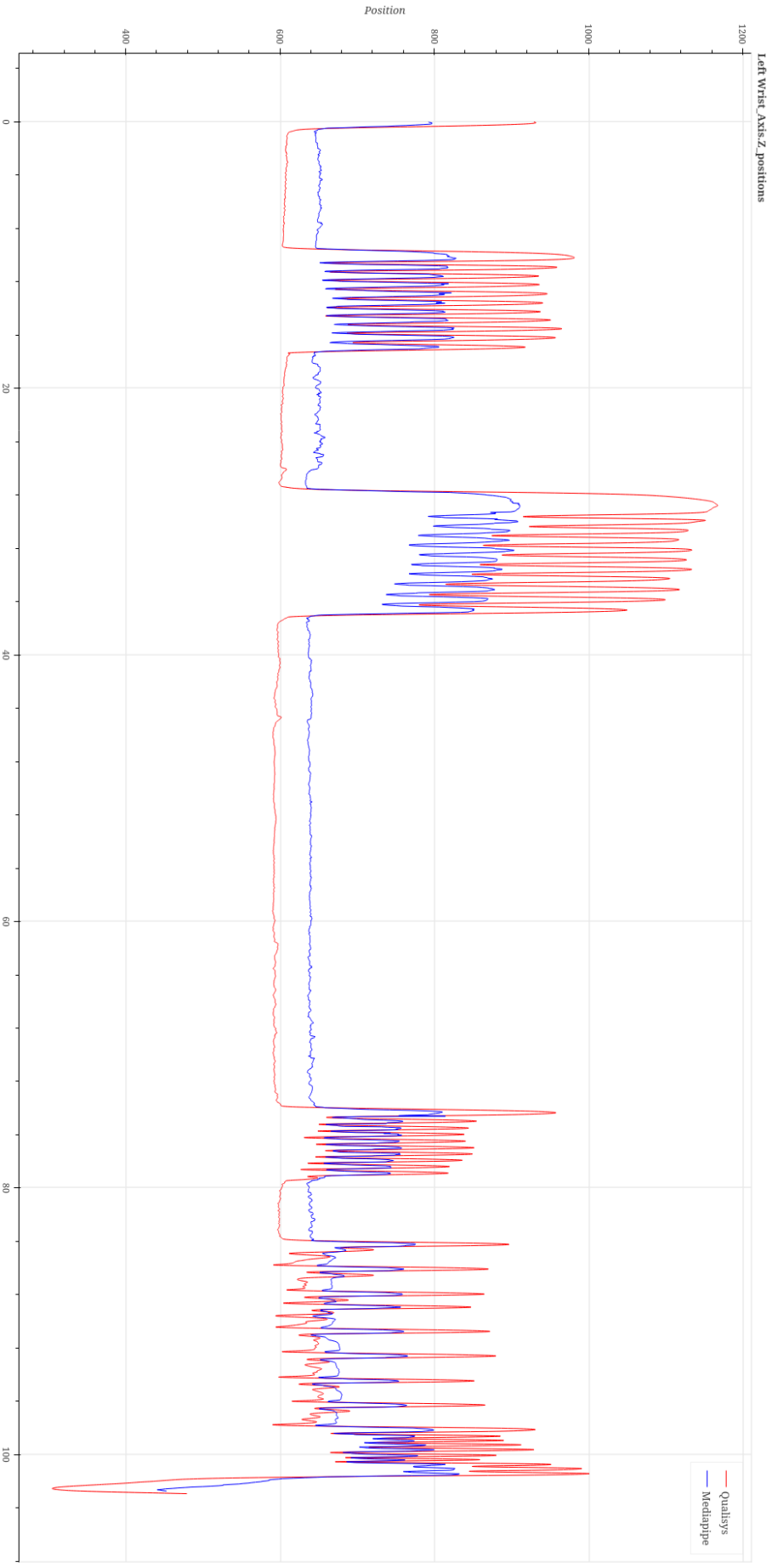


*Figure 7.3: Plot of the MediaPipe (Blue) and Qualisys (Red) left wrist z-axis with the time offset removed.*

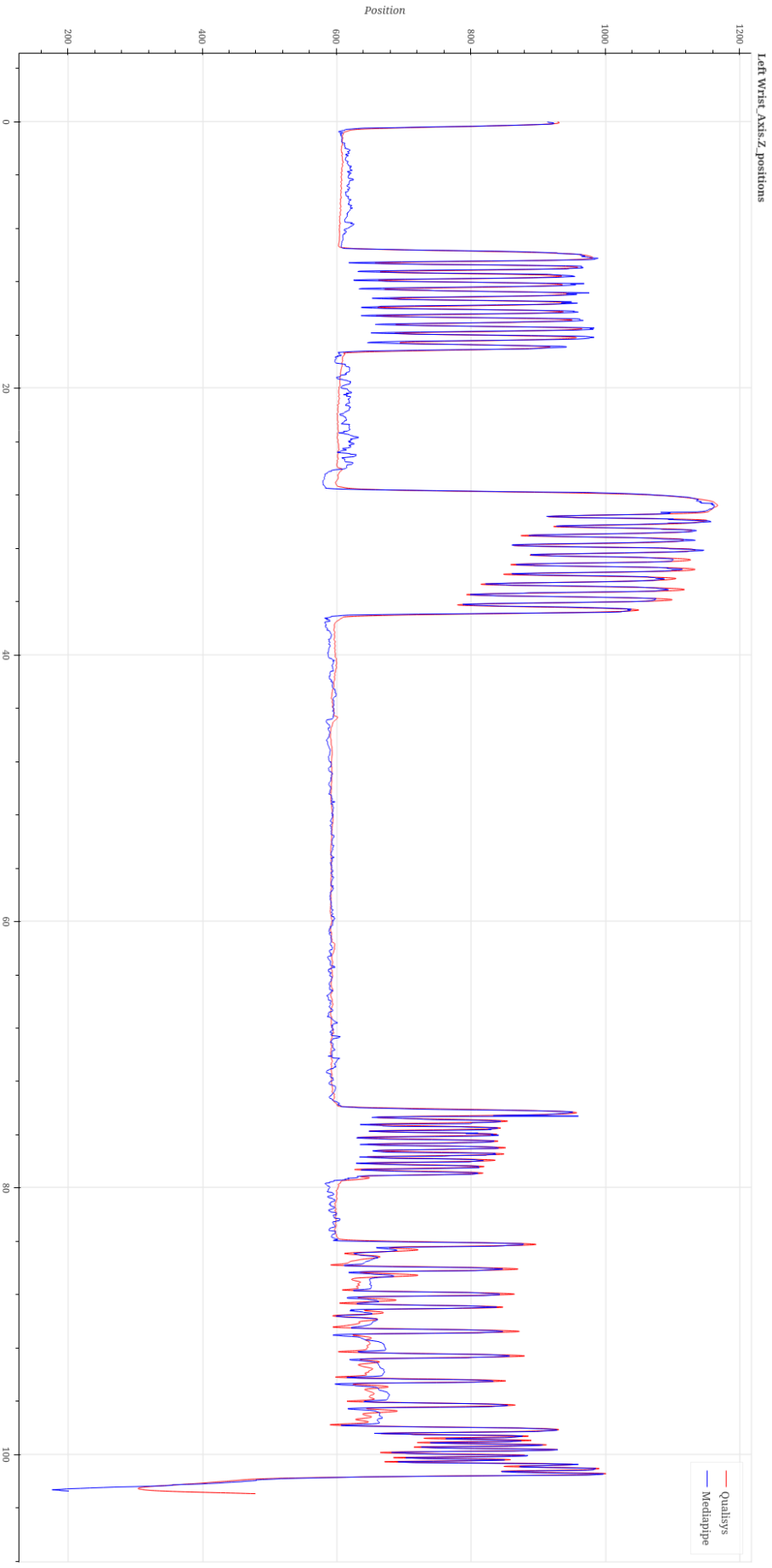


*Figure 7.4: Plot of the MediaPipe (Blue) and Qualisys (Red) left wrist z-axis with the MediaPipe signal scaled to match.*

## References


[1] Google, “MediaPipe Audio Classification.” Accessed: Apr. 30, 2024. [Online]. Available: *https://developers.google.com/mediapipe/solutions/vision/pose_landmarker*

[2] Google, “MediaPipe.” Accessed: Apr. 30, 2024. [Online]. Available: *https://developers.google.com/mediapipe*

[3] Google, “MediaPipe Solutions.” Accessed: Apr. 30, 2024. [Online]. Available: *https://developers.google.com/mediapipe/solutions/guide*

[4] Google, “MediaPipe Audio Classification.” Accessed: Apr. 30, 2024. [Online]. Available: *https://developers.google.com/mediapipe/solutions/vision/gesture_recognizer*

[5] Google, “MediaPipe Audio Classification.” Accessed: Apr. 30, 2024. [Online]. Available: *https://developers.google.com/mediapipe/solutions/vision/object_detector*

[6] Google, “MediaPipe Text Classification.” Accessed: Apr. 30, 2024. [Online]. Available: *https://developers.google.com/mediapipe/solutions/text/text_classifier*

[7] Google, “MediaPipe Audio Classification.” Accessed: Apr. 30, 2024. [Online]. Available: *https://developers.google.com/mediapipe/solutions/audio/audio_classifier*

[8] Google, “MediaPipe Framework.” Accessed: Apr. 30, 2024. [Online]. Available: *https://developers.google.com/mediapipe/framework*

[9] V. Bazarevsky, I. Grishchenko, K. Raveendran, T. Zhu, F. Zhang, and M. Grundmann, “BlazePose: On-device Real-time Body Pose tracking.”

[10] H. Xu, E. G. Bazavan, A. Zanfir, W. T. Freeman, R. Sukthankar, and C. Sminchisescu, “GHUM & GHUML: Generative 3D Human Shape and Articulated Pose Models.”

[11] COCO Consortium, “COCO 2020 Keypoint Detection Task.”

[12] Google, “On-device, Real-time Body Pose Tracking with MediaPipe BlazePose.” Accessed: May 04, 2024. [Online]. Available: *https://research.google/blog/on-device-real-time-body-pose-tracking-with-mediapipe-blazepose/*

[13] A. Bulat and G. Tzimiropoulos, “Human pose estimation via Convolutional Part Heatmap Regression.”

[14] Google, “Model Card: MediaPipe BlazePose GHUM 3D.”

[15] Paul Lindner, “Definition of tab-separated-values (tsv).” Jun. 1993. Accessed: Apr. 27, 2024. [Online]. Available: *https://www.iana.org/assignments/media-types/text/tab-separated-values*

[16] J. Kiefer, *Sequential minimax search for a maximum*. 1953.

[17] Wikipedia, “Golden-section search.” Accessed: Apr. 28, 2024. [Online]. Available: *https://en.wikipedia.org/wiki/Golden-section_search*