UNIVERSITY OF TWENTE.

Master Thesis Interaction Technology

Exploring Somersault Feedback Design using OpenPose: A User Centered Approach

by

Niels van Dalen

Supervisors:	dr. D.W.B. Postma
	dr. ir. D. Reidsma and dr. A. Karahanoglu
Date:	09-08-2024
Credits:	40 ECTS

Abstract

Providing effective feedback is crucial for the improvement of athletic performance, particularly for complex movements like somersaults. Various technological tools are available, though complexity and costs renders them often inaccessible to amateurs. Rapid movements and complex poses from a challenging situation for feedback since they are difficult to observe for both humans and technological systems. This research explores the utility of a easy to use sports feedback system for a complex movement: the somersault. The system uses OpenPose, which is a marker less motion capture system that detects body parts from footage. A contextual exploration was performed both through literature and interviews with gymnasts. After, the performance of OpenPose on a somersault was evaluated qualitatively and quantitatively. Together with gymnastics trainers, a diverging and converging ideation phase led to the development of six low fidelity prototypes for feedback. The designs were reflected on with gymnasts and refined into four high fidelity prototypes, which again were evaluated. OpenPose performs poorly on somersaults, this was accounted for by using a data enhancement strategy as well as a user centered strategy which aimed to bypass the limitations of data by design choices. Altogether, this research provide three main contributions. Firstly, a quantitative analysis of OpenPose's performance on somersaults is given. Additionally, design guidelines for sports technology are presented, including strategies of coping with incomplete or inaccurate data. Finally, four Hi-Fi prototypes were implemented and evaluated, forming a basis for novel somersault learning approaches. In conclusion, OpenPose is a promising system for developing feedback systems for sports.

Contents

1

Introduction

In the process of learning sports, self reflection on performance can be an effective method for improvement. This is not subject to level of skill in sports, the professional athlete as well as the beginner will benefit from reflecting on his/her performance. Various technological tools aim to improve quality of reflection by providing feedback to a user. This includes the usage of body sensors, performance data, custom dashboards, video analyses etc... These technologies come in a wide range of complexity and costs. Professional athletes can afford elaborate and expensive technologies, whereas the average amateur athlete is often limited to recording and playing back videos. Amateurs can benefit from different feedback technologies as well. Thus, a system that is easy to use, has low costs and requires no expertise to operate would be beneficial in the learning process of amateur athletes.

Motion capture (MoCap) is the process of digitally tracking and recording the movements of objects or living beings in space [1]. MoCap can be used in a feedback system in sports, since it has potential to add to the quality of real time human observations. OpenPose is an AI based MoCap system, that performs human pose estimation using imagery as input. Due to its implementation as software, OpenPose is straightforward in usage. Apart from a camera and (optional) spatial reference it doesn't require any calibration nor dedicated hardware. On top of that, lack of dedicated sensors or other wearable components enable freedom of movement to users. OpenPose's performance remains accurate on low quality imagery, retaining decent performance on poorly lighted, noise infected images at 60x60 pixels [2]. OpenPose is the winner of the 2016 COCO keypoints challenge [3], a competition involving the localization of human keypoints in challenging and uncontrolled conditions. More information on OpenPose and keypoints can be found in section 3.1.1. OpenPose's properties enable its usage beyond controlled and calibrated environments. This suggest that OpenPose might be a good MoCap choice in challenging settings, as is the case for many sports.

However, being an AI based system, OpenPose's performance depends on how well the system can generalize to input data (new, unknown imagery). Images that closely resemble training data will be evaluated accurately, whereas images that deviate a lot from training data are at a higher risk of poor evaluation. OpenPose is trained on the common objects in context (COCO) dataset [4], which contains annotated images of people. This dataset contains mostly conventional human poses such as standing, sitting or lying down. Unusual poses, which

are common in various sports (being upside down, tucking in etc.) are underrepresented in training data. Though there are some strategies to minimize inaccuracies (such as data augmentation and extension of the training database [5]), dealing with poor performance (inaccurate data) of OpenPose on unusual human poses is inevitable. This is a challenge that will be addressed throughout this work.

This research explores how OpenPose can be utilized in a feedback system for sports involving unusual poses. It is investigated how the combination of video and (incomplete) OpenPose data can provide rich and tangible (personalized) feedback for sports, ultimately aiming to improve the learning process by exploring the design space of OpenPose and video data. For this, the forward tucked in somersault (from here on referred to as just 'somersault') is taken as a case. Among others, the somersault has been chosen since it is challenging to observe. For humans it is difficult to provide elaborate feedback based on just the observation of a very rapid ballistic movement, and OpenPose struggles with unconventional poses which are prevalent throughout a somersault. Furthermore, the somersault is a fundamental, thus relevant, movement within gymnastics. An elaboration on the somersault and its context is provided in chapter 2.

This research presents three main contributions. Firstly, it provides in depth information on the performance of OpenPose on unusual poses and subsequently presents two strategies of coping with its limitations. Secondly, four design guidelines are proposed for the utilization of OpenPose in designing personalized feedback systems for sports. Thirdly, a feedback system consisting of four different prototypes was developed which can be used for somersault reflection. This forms a basis of a novel learning approach for somersaulting and can also be used as baseline for similar research concerning different sports movements. This research is guided by the following research question:

How can OpenPose data complement footage in providing feedback for a somersault?

2

About The Somersault

This chapter contains contextual information on the somersault. This includes a motivation for the somersault as case, an examination of what constitutes a proper somersault (see 2.2) and current learning strategies for a somersault (see 2.3).

In this chapter various insights are extracted from interviews with gymnastics trainers and athletes (n=9). These are referred to as (P=n) in which n is the number of interviewees out of mentioned nine in total who, without prompting, independently presented the comment. Additionally, field observations and gymnastics resources (recommended by trainers) were consulted. Insights from the interviews are used throughout other chapters as well, the raw interview notes can be found in A.1.3. In advance, all interviewees were notified about the interview via a consent form (see A.1.1). This, among others, indicated the scope of research, their voluntary participation and how their data would be (anonymously) gathered and stored. The interviews were semi-structured along a script containing various questions. The script and written notes can be found in A.1.2. The setting of the interviews was both in situ and ex situ depending on the interviewee's preference and convenience. During the interviews the researcher wrote down notes digitally. Before the conclusion of the interview these notes were summarized by the researcher and subsequently discussed, corrected and approved by the interviewee.

2.1. Motivation for the somersault as case study

As a case for this study, the forward tucked in somersault within a gymnastics discipline is considered. This is an aerial movement involving a full forward rotation of the body along the vertical axis. The somersault is a swift movement that averages around 600ms to complete [6]. The somersault is a key movement during gymnastics since it can be done at multiple apparatuses (gymnastics elements, such as the floor or uneven bars) and forms the basis for many advanced (aerial) routines. The explosive nature of the somersault makes it challenging to properly observe its form and technique. Athletes rotate too fast to visually distinguish their surroundings and determine their orientation well. On top of that, apart from changing inertia, there is no way to make kinesthetic corrections during flight. Furthermore, the somersault can be considered as a holistic movement: one has to commit to the full movement during performance. Additionally, there is a lack of representative (i.e., exercises with high

'transfer' [7]) intermediate learning steps and fear of injury in somersault learning. Moreover, gymnastics classes can be big, rendering (extensive) individual attention not possible. In conclusion, the somersault is a prime example of an unusual human pose that would greatly benefit from innovation on feedback to athletes.

2.2. Characterising a proper somersault

The tucked in forward somersault is the case sports movement for this study. This section provides an overview of what this somersault consists of and which factors are important in order for a somersault to be considered as 'good'. The somersault can be divided into five phases:

1. Run up
2. Jump into trampoline
3. Trampoline take off
4. Flight phase
5. Landing

Figure 2.1 shows these somersault phases. During the run up (phase 1) it is important to gain sufficient speed (P=4) - or kinetic energy - so that sufficient height can be attained. The jump into the trampoline (phase 2) should be powerfully initiated from one foot and landed arms down with both feet next to each other in the center of the trampoline (P=6). In the subsequent forward take off (phase 3) arms should be swung upwards in order to gain as much height as possible (P=7). During flight (phase 4) the athlete controls his/her rotation by tucking in (rotational velocity increases) or stretching out (rotational velocity decreases) making sure that an upright orientation can be attained on landing (P=9). First contact with the landing spot (phase 5) may be achieved whilst tilted slightly backwards, where inertia and forward momentum push the athlete to an upright position while simultaneously finding balance (P=5). A well executed landing doesn't require balance corrections by feet [8] [9].

Additionally, a proper somersault entails keeping the feet and legs closely aligned with toes pointed downwards. In official regulation by the Fédération Internationale de Gymnastique (International Gymnastics Federation, FIG) a somersault is considered 'tucked' if the angles of upper body to thighs (A) as well as thighs to lower legs (B) are equal to or smaller than 135 degrees, see figure 2.2. However, achieving minimal separation between the upper body and thighs, thereby minimizing angle A (see figure 2.2), is considered the best form [9] (P=7). Additionally, arms should be held in close proximity to the body and extended when possible (P=5), with hands positioned below the knees during tucking [8]. Though not grabbing at all will be awarded with more points [9] (P=2). Notably, the specific apparatus or gymnastic event in which a somersault is performed significantly influences the desired technique. For instance, trampoline somersaults require landing in place, whereas mini trampoline or tumbling routines require horizontal translation during the somersault. From the run to the landing the somersault should be performed along a straight line (ie. no lateral deviation). Finally, all interviewed trainers (P=9) stressed that safety measures (for example landing

with arms forward in order to catch over rotation) are considered important KPI's during the learning, despite such measures not contributing to higher points in official competition.

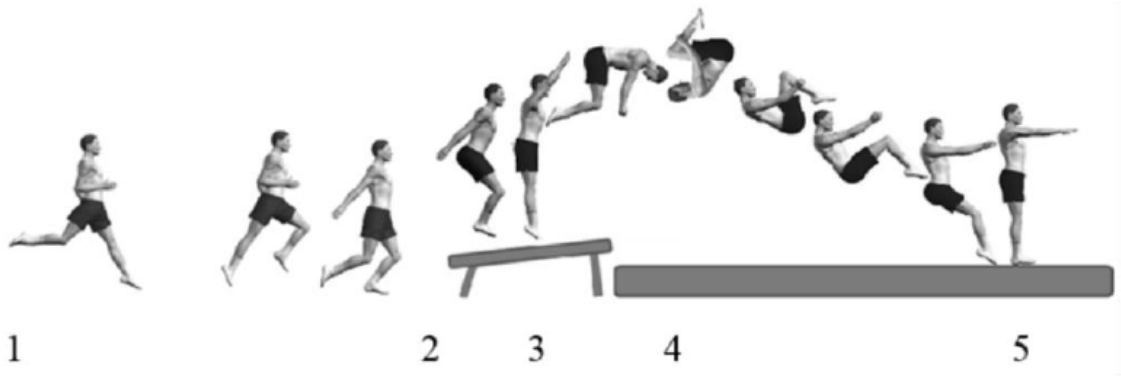


Figure 2.1: A tucked in forwards somersault from a trampoline onto a mat distinguishing five different phases, taken from [8]

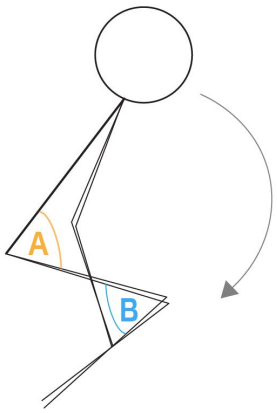


Figure 2.2: Schematic posture of a tucked somersault with angles A,B $\leq 135°$

2.3. Common Somersault Learning Approaches

In learning a somersault, various techniques and exercises need to be mastered before being able to do, or even attempting, a somersault. Therefore new learners (often children, P=7) are presented with various exercises that allow them to train important (kinesthetic) skills required for a somersault. These exercises provide preparation for a somersault with limited injury risks. Below is a list with some common prerequisite exercises in approximate skill (chronological) order:

- Head rolls (with many variations such as: on a downward slope or onto elevated mat, whilst holding objects between feet, knees or chin and chest, with or without hands, P=8)
- Jumping practice (squat jumping, jump and tucking in without rotation, P=4)

- Dive rolls (ground roll from a standing position by jumping forward, P=2)
- 'Tip' somersault (performing a somersault whilst allowing brief hand contact to a surface, P=2)
- Aided somersault (using a salto chair or physical support by trainer, P=7)

Additionally, trainers can opt to provide (personalized) instructions in order to improve certain form elements. For example 'grab your ankles' (P=5) or 'touch your chest with your chin' (P=4) are practices that naturally make the body tuck in, desirable for rotational velocity. More importantly: such intermediate instructions are simple and tangible instruction for children that, by doing so ultimately improves their overall tucking form. All trainers stressed the importance of an individual approach and assessment of exercise type and difficulty (P=4). Additionally, there are a lot of other exercises that can be practiced as prerequisites for a somersault. Some examples can be found at the Dutch 'beter turnen' website [10].

In terms of technology usage, videotaping an exercise and reflecting on the footage is common practice (P=7). This helps athletes become more aware of their true orientation as opposed to their idea, feeling or understanding of it (which is a challenge, if not impossible according to [6]). Such footage forms the basis of various (digital) aids such as the apps 'Videodelay' and 'VisualEyes'; allowing easy video recording, sorting and annotating using a smartphone. An impression of VisualEyes is given in figure 2.3.

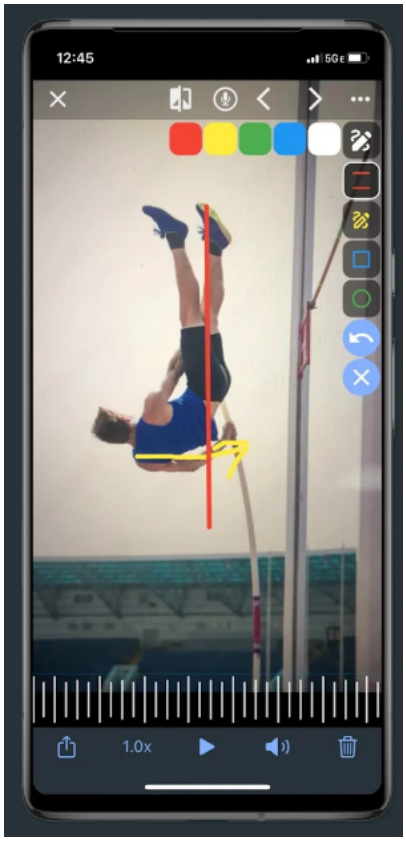


Figure 2.3: Snapshot from the VisualEyes app; an app designed as digital aid in training sports and providing feedback

While such apps offer a good start for personalized feedback, their utility is directly constrained by (quality of) footage. They enable the user to retrieve data from a 3rd person view but remain superficial, lacking the capability to delve into more nuanced patterns and dimensions of data for feedback. Additionally, feedback value of such apps is limited by the reflective competence of athletes themselves.

3

Related Work

This chapter describes related work concerning OpenPose's state of the art, a comparison of OpenPose versus common MoCap techniques, strategies on incomplete data evaluation and data driven sports training approaches.

3.1. About OpenPose

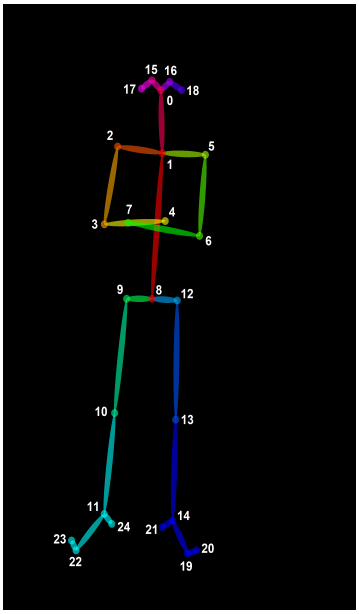
3.1.1. Description of OpenPose

OpenPose is a realtime (multi)person pose estimation system that is able to detect human postures. It takes footage as input and is able to detect human postures using a coordinate system consisting of keypoints (KP's). It uses a convolutional neural network (CNN) in order to find these KP's as well as part affinity fields (PAF) in order to correctly identify multiple persons. Additionally, it proclaims to be the first realtime system for body, foot, hand and facial KP detection [11]. OpenPose can run on images, videos and realtime on a video stream, though experiences suggests that the latter requires expensive dedicated hardware for reasonable framerates [3]. In order to use OpenPose, only a camera and a computer is required. OpenPose's first version was released in 2017 and has since then become widely recognized and used for human analysis research.

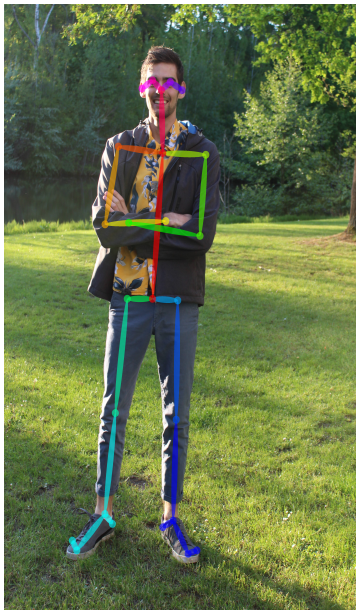
OpenPose's output consists of highlighted (i.e., overlayed keypoints overlayed on footage) frame by frame videos, images or raw data. Openpose writes keypoint data as (x,y,c) coordinates in which x and y represent the position in a frame (or image) and the corresponding c is a value of confidence in the estimation of that keypoint. Although OpenPose supports up to 137 KP's (25 for body, 2x21 for hands and 70 for face), this research will be focused on gross motor skills and will only utilize outputs containing 25 body keypoints. Figure 3.1 displays an overview of the KP's and an example output of OpenPose. The version of OpenPose used for this research is OpenPose version 1.6.0 (GPU release).

```
{0, "Nose"},
{1, "Neck"},
{2, "RShoulder"},
{3, "RElbow"},
{4, "RWrist"},
{5, "LShoulder"},
{6, "LElbow"},
{7, "LWrist"},
{8, "MidHip"},
{9, "RHip"},
{10, "RKnee"},
{11, "RAnkle"},
{12, "LHip"},
{13, "LKnee"},
{14, "LAnkle"},
{15, "REye"},
{16, "LEye"},
{17, "REar"},
{18, "LEar"},
{19, "LBigToe"},
{20, "LSmallToe"},
{21, "LHeel"},
{22, "RBigToe"},
{23, "RSmallToe"},
{24, "RHeel"},
{25, "Background"}
```

(a) Keypoints



(b) Keypoints render



(c) Keypoints rendered onto image

Figure 3.1: OpenPose example output

3.1.2. OpenPose's State of the Art

OpenPose can be considered a state of the art software based MoCap system. Nonetheless, various current cases of improper detection are known. Besides the risk of false detection on rare poses, the developers distinguish a number of other cases that pose problems to the performance of OpenPose, originating mostly from obstructed body parts, visual interference of multiple people and objects, or animal resemblances to humans. A more in-depth explanation of why such errors occur and some potential solutions are given in [11]. Some common failure cases are shown in figure 3.2. For the case scenario of this research, multi-person as well as resemblance issues can be easily avoided. On the other hand, occlusion and rare poses constitute a potential large risk of failure. Thus, issues c,d,e,f (as specified in 3.2) are unlikely whereas rare poses (a) and missing or false parts detection (b) are considered major risks, which in a sports context was also found by [5]. The somersault is a rare human pose and missing or false detection can occur because tucking induces occlusion.

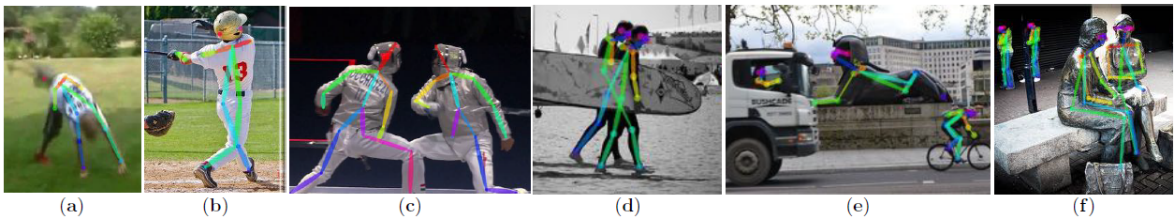


Figure 3.2: Common failure cases as determined by the OpenPose developers: (a) rare pose or appearance, (b) missing or false parts detection, (c) overlapping parts, i.e., part detections shared by two persons, (d) wrong connection associating parts from two persons, (e,f): false positives on statues or animals. Taken from [11]

The performance of OpenPose, being an AI based system, is limited by the accuracy and consistency of labels of training data, as well as the representativeness of training data [12][13]. Though there are possibilities to verify OpenPose's results in hindsight by comparison or manual corrections, this is undesirable since this is not always possible and in any case a very laborsome task. Without reliable reference, keypoints of obstructed body parts can never be verified, thus never accurately evaluated. Although the relevance hereof depends on context, the true accuracy of a system like OpenPose is unknown in some cases. Additionally, it was found that for high accuracy applications, OpenPose was not adequate compared to bi-planar radiography, considered a gold standard [14].

Rare poses tend to fail because the COCO dataset that OpenPose was trained on, contains few rare poses. Thus making OpenPose overfit its training data. In an attempt to improve performance, [5] resorted to two strategies. Firstly, refining the dataset with multi-angle sports pose footage in order to improve training data diversity. Secondly, implementing an algorithm that orientates imagery such that body poses are upright (i.e. head at the top and feet at bottom). This approach of rotating input footage is also proposed by [15], [12] and proposed as a solution by the OpenPose developers [11]. With this approach performance on sports imagery (i.e., the confidence level of keypoints) improved by approx. 26% [5], and processing intensity increased by a factor of eight. In [15] and [12], no quantitative performance figures were given, though both studies reported a significant improvement using the rotating input imagery methodology. No figure for processing load was given by [15], though it is suggested by [12] likely that the processing load increases depending on the size of the rotational angle interval taken, which was between a factor 4 and 36 in [15].

As mentioned, occlusion of body parts can lead to failure of detection by OpenPose. Missing body parts were considered as noise by [16], and then fed into an autoencoder in order to predict hidden parts. This was found to decrease missing keypoint values by approximately 25%. However, performance depended largely on the body part of interest. The amount of false ankle detections dropped by 52%, while performance on the neck keypoint remained the same.

3.2. Comparing OpenPose to other Motion Capture techniques

This section provides a qualitative overview of common MoCap techniques and a subsequent comparison thereof against OpenPose. For an elaboration on the workings of OpenPose, see section 3.1.1.

Capturing motion can be achieved with various technologies. Some common approaches involve the usage of depth sensitive cameras [17], inertial measurement units (IMU's) [18], electromagnetic field sensors [19] or optical markers [20]. Hybrid MoCap systems combine two or more techniques, this can increase accuracy as well as resolve problems such as occlusion [21]. Hybrid MoCap systems are out of the scope of this research.

Perhaps the most popular depth sensitive camera is the Microsoft Kinect. Besides a 'traditional' (optical) camera, depth sensitive cameras use an IR light and sensor in order to determine the Time-of-Flight (ToF), which corresponds to how close or far an object is. In the case of the Microsoft Kinect, this poses two problems: diffusely reflective materials become undetected [22] and the system is susceptible to IR noise, especially outdoors [22][23][24]. An advantage of depth cameras is direct output of 3D information. OpenPose would require multiple viewpoints and triangulation thereof before resulting data would comprise 3-dimensional information. Similar to OpenPose, depth sensitive cameras require little to no calibration, this is merely required for sophisticated high accuracy computer vision applications [25]. However, software accompanying depth sensitive cameras might be prone to misdetection of unusual poses as well. An anecdotal exploration of the Kinect sensor for somersaults is described in section 4.1.1.

Another approach to MoCap involves inertial measurement units (IMU's). IMU's are sensors that detect inertia, orientation and gravitational force [26]. For MoCap, IMU's are designed to be wearable and placed at various parts of the body, sometimes using a suit that embeds them. IMU data is then transmitted wirelessly. A big advantage of IMU's is their insensitivity to any visual influences such as unfavorable lighting conditions or visual obstruction, well-known problems for depth cameras and OpenPose. IMU's are generally considered to be very accurate [18], and can get quite expensive [5]. In [27], a comparative study of OpenPose against IMU's was done on gait. Here, IMU data was used as baseline comparison. It was found that in ideal conditions (used cameras had full visibility of all joints) OpenPose's joint angles deviated $1.6° \pm 0.3°$ from IMU values, yet a more realistic scenario (which does include occlusion) produced absolute errors of $14.0° \pm 1.8°$ where the magnitude of deviation depends on the camera positioning. Though the relevance depends on context, a 14 degree angle deviation is quite substantial. Especially since the research merely concerns gait: a very common appearance, thus a high expected performance of OpenPose. On the other hand, wearable sensors such as IMU's inherently restrict freedom of movement, especially when embedded in a suit. IMU's do require calibration in order to function well. Taking this into account, IMU based MoCap for groups will be laborsome. On top of that, the problem of soft tissue artefacts dictates a varying accuracy [28] dependant on sensor positioning and type of movement, requiring recalibration if possible, though in any case decreasing quality of data [29].

A gold standard within the field of MoCap is optical marker tracking. Markers are placed at known points on an object or body, after which the resulting point cloud is labelled to

compose a 3D model of the subject. Marker tracking is known for high spatial and temporal accuracy [30] given that markers and camera are precisely positioned during calibration and the capturing scenario resembles that of calibration [31]. On the other hand, marker based MoCap systems are sensitive to soft tissue artefacts (similar to IMU's). They are also time intensive to set up and do require trained operators [13]. In another comparative study on gait analysis, OpenPose was compared against optoelectronic marker-based three-dimensional motion capture (3DMC). OpenPose's joint angles for hip, knee and ankle were under all conditions (i.e., various types of gaits, using a single side-view camera as input source for OpenPose) within a 5% margin of 3DMC [32]. Under the assumption that the 3DMC methodology is a reasonable baseline comparison, this indicates that OpenPose, given proper conditions, can provide rather accurate lower body angles. According to the data of this study, the overall deviation (over all body parts under all tested conditions) was 4.5° ± 1.7°, with a minimum deviation of 2.3° and a maximum deviation of 9.4°.

All in all, capturing the human body and its movements is a problem to which many (partial) solutions exist. In controlled environments state of the art marker based systems achieve high accuracy, at the expensive of extensive calibration and required operating competence. Occlusion of markers or body parts remains a challenge for marker based systems, depth cameras and OpenPose. Though IMU's are not susceptible to occlusion, these wearable sensors might limit freedom of movement. Depth sensitive cameras provide a cheap and non-intrusive strategy for MoCap, though they risk misdetection due to IR interference or diffusely reflecting objects. Compared to other MoCap techniques, OpenPose is a system that requires little hardware and is very flexible in usage, though it is more susceptible to incomplete data which is also more difficult to verify. OpenPose's accuracy is less than that of state of the art techniques such as IMU's or optical marker tracking. Though the exact application determines whether this is relevant.

3.3. Processing and Interpreting Incomplete Data

Regardless of MoCap methodology, practical work in capturing motion inherently involves interpreting raw data and more importantly, accounting for missing data points [33]. This section is a study of literature concerned with approaches of dealing with incomplete human movement data.

Complementing an incomplete dataset can be done by interpolating missing values. In a study by [34] three interpolation methods were performed on an artificially diminished human movement dataset and compared against the full dataset. Several 'missing data' intervals (i.e., duration of movement of which data was omitted) were created ranging from 200-850ms. A Local coordinate system (LCS) interpolation method was most successful yielding a 9.3% deviation error overall and also proving to perform best at longer intervals. An average somersault takes around 600ms to complete [6] and it is possible that OpenPose will fail to detect most keypoints during the tuck and rotation, a substantial part of that duration.

Similar to the duration of a missing data interval, sample rate of data influences the performance of interpolation techniques [35]. Backpropagation (BP) interpolation was found to increasingly outperform the common midpoint (CMP) methodology the higher the sample rate of data. Next to interpolation methods, inverse kinematics (IK) can also be used to find

missing data points. Downside of this is approach is that some initial data is required (for example: subsequent frames of a wrist location) in order to determine missing body parts and joint angles. Nonetheless, using IK for automated gap filling in data is very promising: [36] found its accuracy is not compromised compared to manual gap filling and this can be achieved in a negligible amount of (processing) time.

Apart from estimating missing values in data, one can also accept the data as is and opt for a different data analysis approach. This was done by [33] in a study on human activity recognition where up to 25% data was missing. Rather than running algorithms to predict missing values, a Hidden Markov Model was used in order to identify activities with the available data. In such an approach data values (whichever available) are used to indirectly infer corresponding observations. In this case the data consisted of keypoint values and the corresponding observation a type of human activity. For example 'eating' corresponded with a lot of vertical hand movement, whereas 'running' corresponds with a large vertical range of keypoints (i.e., feet are far away from head suggesting person is standing) and a lot of movement by the legs. The system achieved activity recognition at 94.7% - 100% precision depending on activity type. Thus, identifying for example different phases of a somersault with incomplete OpenPose data is likely possible without missing data value estimation techniques. The downside is that this approach is rather holistic and doesn't support extraction of more sophisticated continuous information features.

3.4. Data-driven Sports Training

Human observations are prone to subjectivity [37] and only recall 30% of key factors that determine a sports performance [38]. Using data to complement observations is an effective way to gain better insight into the learning and/or performance of sports [37] [39] [40] [41]. Additionally, data over time (i.e., weeks, months or years) allows for capitalisation on less apparent changes and more elaborate options of interpreting progress. Required is that data should have good literacy. Data literacy is defined as "the knowledge on access, interpretation, communication and preservation of data in order to gain new understanding of the world" [42]. Data literacy is determined by acquisition methods, but also the capabilities of people involved with the data and the goal of using data.

An example of qualitative data usage is presented in [43]. A headset and motion capture system was designed to aid an athlete's learning by providing a new perspective on their own posture and movement via the display of a real time avatar. The athlete can select the viewpoint and other settings such as gridlines as well as record and play back the avatar's motion. Researchers found that for both an American football and bowling the new perspective was found to be helpful in improving posture and self-reflection, during practice of their movements.

Quantitative assessment of an activity is a challenge given each exercise has a unique motion characteristic. In order to generalize interpretation of sports movement (such as a somersault), consistency can be considered a property of proficiency, with higher consistency implying more proficiency. Using this approach [44] concluded this can be an effective way of using data to evaluate performance. This method is robust against missing data points yet on itself cannot provide nuanced feedback.

4

Design Process

This chapter describes the full iterative design process of creating prototypes that aim to provide meaningful feedback for a somersault utilizing OpenPose data in combination with video. For this, the limitations of OpenPose pose two main challenges. Firstly, how can prototypes be developed to provide effective feedback whilst not being fully dependant on complete data coverage. And secondly, how can OpenPose's data be retrospectively enhanced in order to improve its overall quality. Note how the formulation of these challenges induce a user centered approach and a technical approach are taken.

This process of solving these questions involved an analysis of OpenPose's performance on somersaults, a diverging and converging ideation phase, a technical implementation of prototypes and its user evaluations.

4.1. Evaluation of OpenPose Keypoints during a Somersault

Chapter 3 presents insight on the performance of OpenPose under various conditions and different poses. Since this study specifically considers a somersault, it is relevant to investigate the performance of OpenPose on somersaulting in order to get a complete understanding of the data that can or can not be provided for this context.

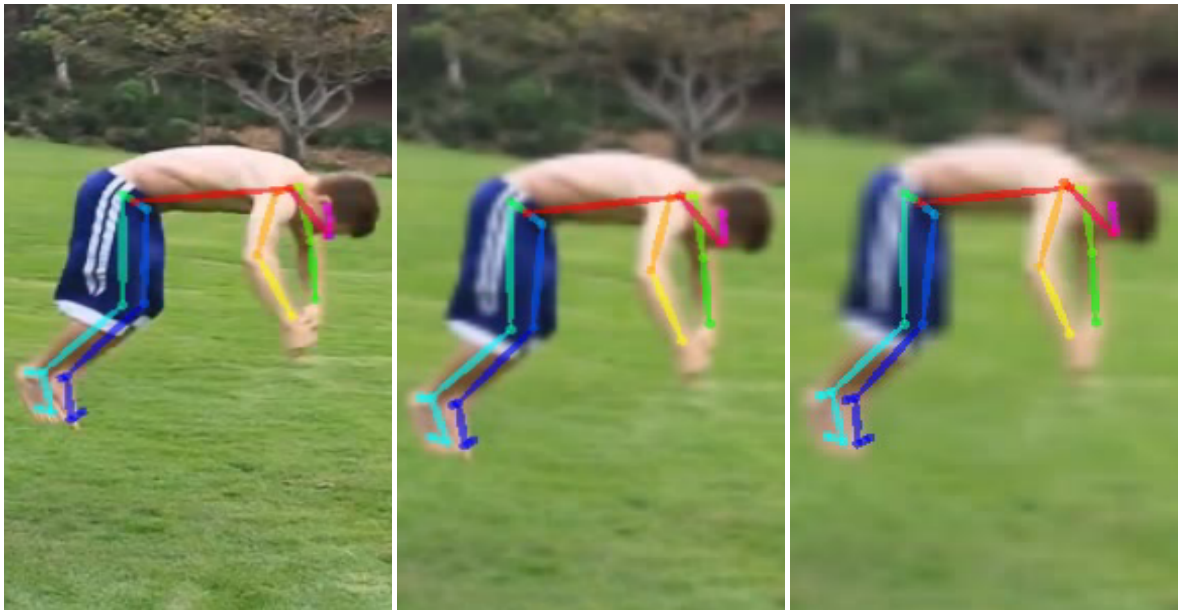
4.1.1. Initial Exploration

This section describes the outcome of qualitative preliminary testing with OpenPose and the Microsoft Kinect sensor.

Initially, a small dataset was created consisting of randomly chosen gymnastics clips (1:07 total video length, taken from the SVW 'Sports Videos in the Wild' database [45]) and some self shot footage of somersaults (0:41 total length). Testing was one in order to explore different conditions that might be relevant to OpenPose's performance. The results are summarized in table 4.1 below. Note that these are not scientifically grounded conclusions, but rather some anecdotal observations meant to aid in determining a proper recording setup for further testing. An example of the type of 'evidence' that led to these ideas is given in figure 4.1.

Table 4.1: Initial Remarks on OpenPose Performance on Gymnastics Footage

Variable	Remarks on Performance Change
Video encoding (mp4, avi)	No perceived performance change
Lighting	Little change, note: hard shadows are sometimes detected as a person
Resolution (1080p, 720p, 360p, 144p, 72p)	Only relevant at very low resolution, no noticeable differences down to 144p
Framerate (30fps, 20fps, 10fps, 5fps)	No perceived performance change
Motion blur	No perceived performance change
Amount of people in frame	Very relevant, OpenPose can 'jump' around people during rare poses, multiple people to be avoided
Amount of objects in frame	Relevant, 'busy' backgrounds with elongated shapes were prone to false detection
Age/Length/Gender	No perceived performance change



(a) 1080p **(b)** 144p **(c)** 72p

Figure 4.1: Example comparison of different Resolutions as input for OpenPose

4.1.2. Quantitative Analysis

This section provides a quantitative analysis of the performance of OpenPose on a somersault based on the estimated confidence levels of keypoints and the amount of empty frames (i.e., no person detected).

For this evaluation a number of somersaults (n = 115) was performed and captured by action cameras (GoPro) at 60fps and a resolution of 1080p from two different angles: 0 degrees (front view) and 45 degrees (diagonal view) respectively. Cameras were placed at 120cm height, an estimate for the vertical center of the body during somersault. The somersaults were performed on a trampoline. The somersault footage was manually separated per somersault and run through OpenPose, that returned annotated frames and keypoint data as JSON files. A python script (see Appendix D, 'quantitative_analysis.py') was used to extract and process keypoint data from the JSON files in order to make graphs. The graphs from the diagonal viewpoint illustrate this section, the front view's graphs support these findings and can be found in Appendix B.1. For the keypoints, a distinction is made between considering all keypoints ('raw') and merely the most relevant keypoints for a somersault ('weighted'), as proposed by gymnastics trainer interviewees. The keypoints that are considered relevant are given in table 4.2. A keypoint is considered 'relevant' when mentioned twice or more indepently by interviewees.

Table 4.2: Relevant keypoints for somersault assessment as indicated by gymnastics interviewees (n=9)

Keypoint Index	Body part
0	Nose
1	Neck
2	RShoulder
3	RElbow
4	RWrist
6	LElbow
7	LWrist
8	MidHip
9	RHip
10	RKnee
11	RAnkle
12	LHip
13	LKnee
14	LAnkle

Figures 4.2, 4.3 and 4.4 confirm the expected (poor) performance of OpenPose on unusual body poses. In terms of confidence, keypoints are estimated better on frames showing typical human poses (phase 3 and 5, see 2.2) and are evaluated poorly during tucking and rotating during flight (phase 4). Note that undetected keypoints have a confidence value of 0. Additionally, phase 4 induces a large number of frame drops (i.e., no keypoints/person detected at all), peaking at well over 30 % (diagonal) and even 50% (front) depending on

viewpoint. The confidence levels of the front and diagonal both substantially drop during phase 4, yet their cumulative overall confidence values differ (see table 4.3).

(a) frame 0 **(b)** frame 21 **(c)** frame 42 **(d)** frame 62

Figure 4.2: OpenPose render of a somersault (No. 88) from the dataset (diagonal view)

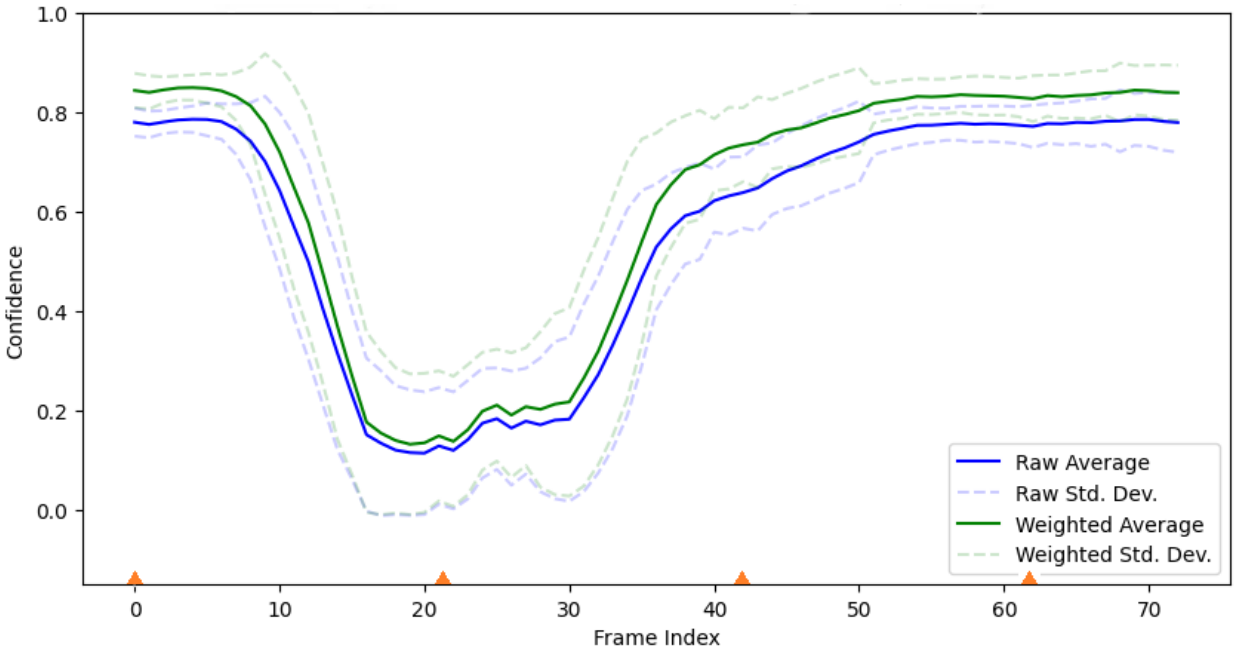


Figure 4.3: OpenPose overall keypoint confidence of somersaults (n = 115) - diagonal view. The orange markers represent the location of the rendered frames in figure 4.2.

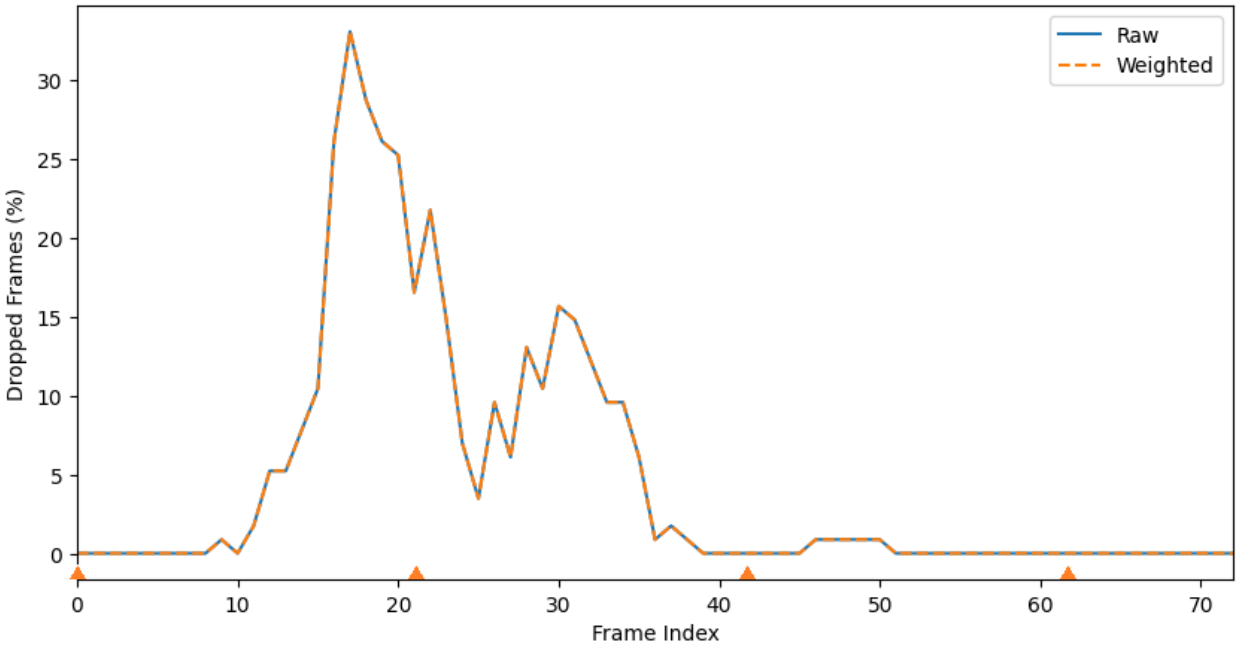


Figure 4.4: Cumulative amount of frames dropped (i.e. no keypoints detected) of somersaults (n = 115) - diagonal view. The orange markers represent the rendered frames in figure 4.2.

Table 4.3: Average OpenPose keypoint confidence level of a somersault (n=115)

	Raw	Weighted
Front View	0.493	0.537
Diagonal View	0.563	0.621

Viewing angle influences the performance of OpenPose. In [32] it was found to be a substantial factor. In this study the data induces a maximum of 15.6% difference in confidence level. This is likely due to less obstruction of keypoints for a diagonal view. Weighted (relevant) keypoints are assessed better than raw (all) keypoints, thus it can be assumed that filtering on merely the weighted keypoints will decrease processing while improving the quality of data.

4.1.3. Conclusion of OpenPose's Performance on a Somersault

OpenPose's performance overall is consistent regardless of technical properties of footage such as resolution, frame rate and encoding. The environment seems to be more significant: hard shadows, complex backgrounds and multiple people in frame can all induce false detections. OpenPose's greatest limitation is evaluating the tucked rotation. Various body parts are obstructed and the atypical human form poses a challenge in properly distinguishing a person and his pose. This leads to a big drop of keypoint confidence levels as well as dropping up to approx. 55% (front view) of frames during tucking. The results of this quantitative analysis are in line with other sports evaluations of OpenPose described in [5][12][13][15]. It can be concluded that OpenPose has significant limitations when attempting to accurately measure a somersault.

4.2. Relevance and Technical Feasibility of Biomechanical Somersault KPI's

Section 2.2 presents an overview of important factors that characterise a (proper) somersault and section 4.1 adds to this by highlighting how not all keypoints are equally insightful from a gymnastics point of view. In this section, somersault key performance indicators (KPI's) are being prioritized by a) their relevance (or weight) for performance of a somersault and b) the expected technical feasibility of measuring them, based on the OpenPose performance analysis as also described in section 4.1. The KPI's are based on aforementioned interviews (n=9), scientific literature on somersaulting ([46] and [47]) and the International Gymnastics Association's judgement rubrics ([9]). The KPI's are elaborated in the list below along with quotes and paraphrases where applicable. The number of interviewees imparting a notion is again given as P=n.

Phase 1: Run up

1. Enough translational speed during run up ('enough built-up energy', P=4, 'explosive power' [47])

Phase 2: Jump into Trampoline

1. Adequate translational velocity of jump ('big leap', P=4)
2. Jump is one legged (P=4)
3. Central jump takeoff position in trampoline (P=6)

Phase 3: Trampoline Takeoff

1. Adequate vertical takeoff ([46], 'gain sufficient height' P=7)
2. Adequate angular momentum ([47])
3. Legs and feet are kept together ([47], P=2)
4. Stretching out fully at takeoff ('stretched out takeoff', P=6)
5. Proper takeoff angle ('this determines phase 5[landing]', P=2)
6. Downward swing of arms for momentum (P=7)

Phase 4: Flight Phase

1. Adequate rotational velocity ('good control and timing thereof' P=9)
2. Enough height ('higher is better', P=7)

Phase 5: Landing

1. Slightly bent knees ([9], P=4)
2. Hands forward (balance/safety, P=2)
3. Core posture (reflects success of phase 4, P=2)
4. Correction of movements after landing (P=9, [9])

A combination of KPI's and their expected technical feasibility is provided below in figure 4.5.

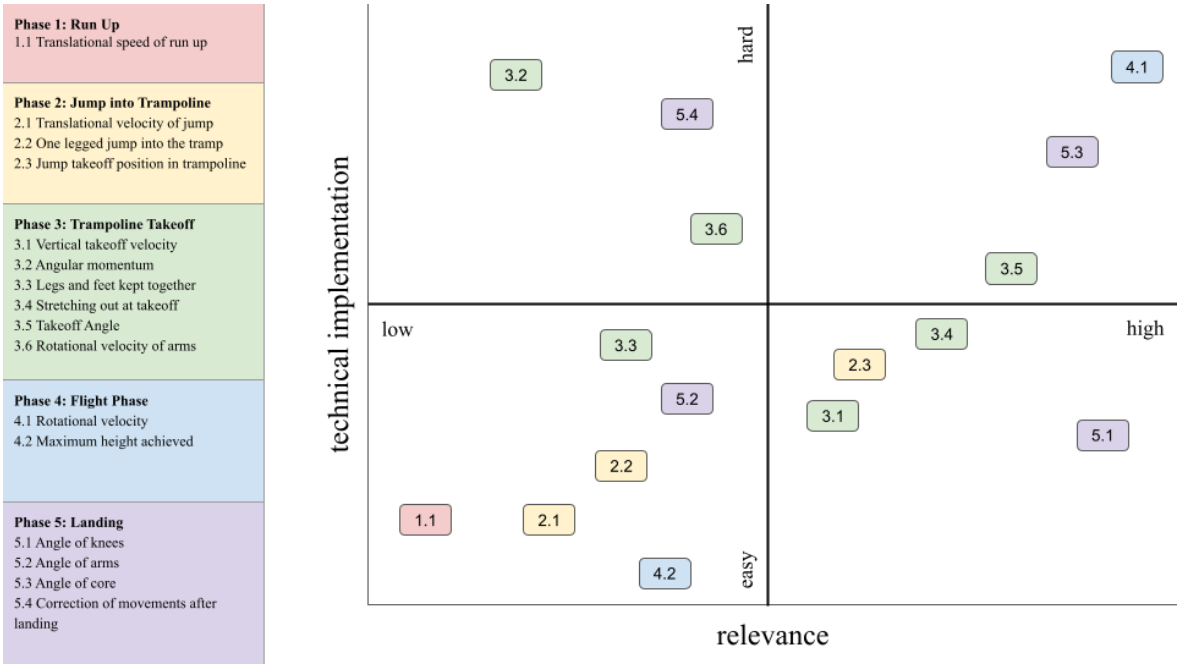


Figure 4.5: Scatterplot of KPI relevance versus technical feasibility

This scatterplot is used as input for further brainstorming (i.e., which KPI's are most effective designing for) and as a tool in selecting the best of said ideas (based on feasibility).

4.3. Idea generation and Design Themes

After gaining contextual understanding of somersaulting and evaluating the respective performance of OpenPose, a total of 78 design ideas was generated. Idea generation was guided by the question 'What is the unique value of OpenPose in combination with video data?'. Brainstorming was done in three consecutive sessions, one by the researcher individually and two accompanied by a gymnastics trainer. Brainstorming involved short brainwriting sessions followed by a brief discussion in order to clarify/extend ideas. As aid three personas were used. The personas consisted of a competitive gymnast, a gymnastics trainer and a child at gymnastics class. They can be found in Appendix C. Ideas were grouped based on conceptual similarity (affinity mapping) and six design themes were distinguished, see figure 4.6.

1. **Gamification:** involves game-like elements in the design, such as competition, rewards etc.
2. **Interaction:** users explicitly interact with a system, they can give input and select options).
3. **Numerical/Direct Feedback:** presentation of quantitative (physical) aspects of the somersault, such as velocity and limb angles.

4. **Guidance:** instructions and/or aids are given to the user, such as visual guidelines and examples.
5. **Data Visualisation:** ways of effectively visualizing data to the user, exploring different concepts and forms of visual feedback.
6. **Compare Yourself:** presenting a comparison in order to learn or reflect, intrapersonal and interpersonal.

Figure 4.6: Design Themes

4.4. Lo-Fi Prototyping

For each of the six design themes distinguished in 4.3, the most promising idea was chosen based on a number of criteria:

1. It generalizes well (does not concern a specific user group)
2. It is not constrained by prerequisite knowledge (the value of feedback is not dependent on the skill or prerequisite knowledge of the user)
3. It is a neutral tool (i.e., the system itself does not objectively evaluate performance, that is out scope for this research)
4. It is feasible (guided by figure 4.5)

Subsequently, the best ideas were made into a low fidelity (Lo-Fi) prototype. Lo-Fi prototyping is an effective approach at gathering user opinions before investing in elaborate development, which risks bypassing user's true needs [48] [49]. The goal of these prototypes is to credibly convey a design idea in order to receive feedback from users. The Lo-Fi prototypes are not functional (i.e., not automated and/or connected to real data) and serve as a (visual) impression of concept that invites reflection from users.

Prototypes were evaluated with adult gymnasts of varying ages (19-53 years), level of skill and occupation within gymnastics (i.e., gymnasts, trainers and judges) differed between participants. This is considered valuable as it makes for a representative test group. In total 12 gymnasts participated in testing the Lo-Fi prototypes, this was done in four rounds with groups of three persons each. A group setting was chosen since this stimulates discussion and collaborative reflection [50] [51]. The prototypes show (different) somersaults of the same person and written notes were made in order to capture user's input. A summary of the evaluation method is provided in the list below, the full evaluation protocol is provided in Appendix A under A.1.4.

- **Research Question:** Which feedback concepts are valuable to users when reflecting on the performance of a somersault?
- **Sampling:** Adults gymnasts (n=12)
- **Evaluation Methodology:** Focus groups, guided by general and prototype-specific questions
- **Data Acquisition:** Written notes (verified by users)
- **Timespan:** 26 minutes
- **Location:** Room within the gymnastics hall

4.4.1. Evaluation of Prototypes

Compare Yourself
This video prototype consisted of a video with a temporally synced 'ghost' overlay of another somersault. Users perceived it as an information overload. It was hard to distinguish between the 'current' jump from its comparison and the realtime playback speed was way too fast. A slowed down playback or merely the frames at certain key moments of the somersaults (see 2.2) were suggested. A comparison should not be random but against a verified 'good' or 'bad' somersault in order to get valuable input for reflection, with comparison against a better somersault being noted as more potent for learning purposes. Finally, as a general tool users saw value in this concept, yet for improving more specific aspects of the somersault they found the guidance prototype (see 4.4.1) more potent since this helps to narrow focus on a single body part, thus increasing the concreteness of feedback.

Figure 4.7: Compare yourself: video shows a 'ghost' image of another somersault

Data Visualisation
This prototype presents a comparative graph showcasing a(ny) physical quantity in sync with a video playback, perceived as easy to interpret for users. For its three comparison modes (previous, average and ideal graphs) people saw most value in an ideal graph since 'comparing only to yourself does not show you how to improve your technique'. A challenge for this idea is the concept of an 'ideal' trajectory or somersault, participants agreed upon the notion that somersaults can be well executed in various different ways (such as: different timings of rotation and the horizontal distance traversed during the somersault).

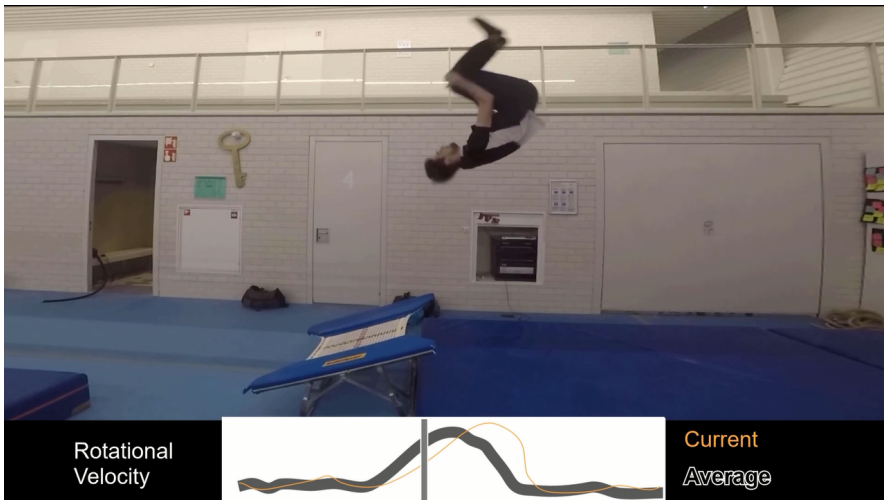


Figure 4.8: Data visualisation: shows comparative graphs corresponding to a physical quantity in time

Gamification
This three part video prototype shows scores for aspects of a somersault (height, rotational velocity and stick, the latter being the degree of landing without requiring balance corrections). Showing rotational velocity was considered redundant and not insightful ("this seemed illogical to me, I will be making one full rotation in any case"). The scoring system was found to have potential only in group settings. Users saw little value in personal score tracking and questioned how meaningful a property as jump height would be in a group where athletes have different body types and techniques. Though they did see added value in putting a parameter like height into perspective (as a rating for kids or when accompanied by a aid such as a grid that makes it easier to visually interpret height). Overall, users were unsure of the potential of a quantifying system as a whole: "to me, gymnastics is mainly about elegance".

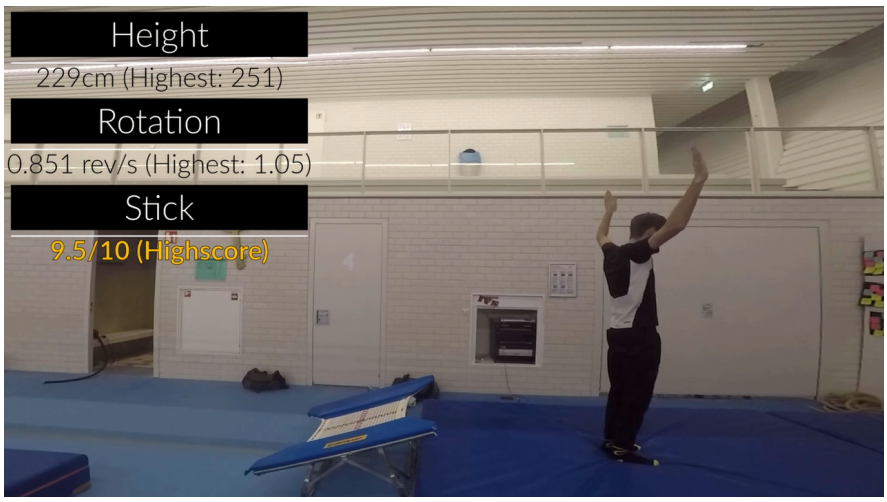


Figure 4.9: Gamification: award scores to aspects of your somersault

Guidance
As a tool for guidance, a prototype was developed to present the idea of a continuous line that reflects a body part's trajectory (specifically, the arbitrarily chosen left foot). This line could be shown individually or compared to another somersault. Individually, the line was perceived as rather (too) abstract. Additionally, "the perfect line" of a somersault was claimed by some participants to not exist, but rather is dependant on many physical and executive factors of a somersault. In line with findings from the *compare yourself* prototype, line comparisons ought not to be random but against a verified level of somersault in order to be instructive. However, whereas a full body comparison can be overwhelming still, highlighting a specific body part was evaluated as useful; participants were able to point out many points of improvement for the researcher's somersault based merely on this prototype. Finally, this concept is potent for visualizing main axes of rotation, i.e., which body part is most static during rotation and can therefore be considered as an imaginary axis an athlete is rotating 'around'. This can be a performance indicator of a somersault.

Figure 4.10: Guidance: traces and draws a body part's location

Interaction
This prototype enabled users to temporally navigate a somersault as well as show body angles of their somersault at specific moments in time. This prototype focused on the usability of an interface that allows users to control their feedback. Most notable is that users did not mind the discretization of the somersault, viewing the key phases was enough for them to reflect on the somersault. Participants were positive about the ability to review angles. However, the angles are limited by 2d coordinates and thus might not always represent the full context or information of interest. In a critical discussion initiated by questioning the concrete value of angle figures, participants reached a conclusion that comparison against oneself or a 'good' or 'ideal' somersault would be most effective. Again, advocating for an evaluated set of somersaults.

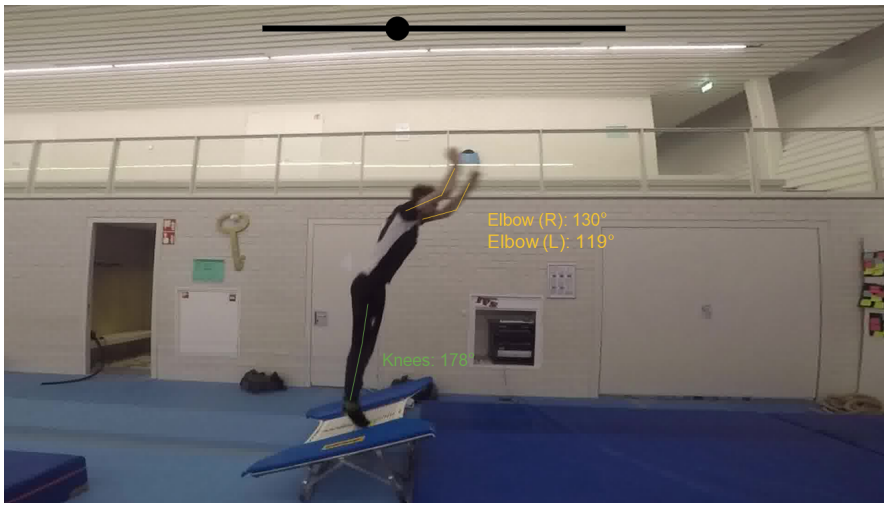


Figure 4.11: Interaction: enables user to temporally navigate their somersault and show quantities

Numerical/Direct Feedback
This prototype provides data that can be extracted from OpenPose data directly: velocity vectors and body angles. Participants did not see much value in the presentation of quantitative figures because (without context) they are are too abstract thus hard to link to reflective thoughts on a somersault. Grid lines were proposed in order to clarify and contextualize said angles. Presented velocities were found to be surplus as participants seemed to extract more information from the video directly ('seeing my horizontal doesn't make sense to me since I can spot that from where I land'). In conclusion, this prototype was perceived as too abstract would require more context to its metrics.

Figure 4.12: Numerical/Direct Feedback: display of physical quantities at certain points in time

4.4.2. Conclusions of Lo-Fi Prototype Evaluation

Based on the evaluation of these Lo-Fi prototypes several pointers guide the design space of OpenPose data in conjunction with video.

1. **Temporal simplification:** representation of time should be simplified or controllable to liking (for example, use only the frames at key moments during the somersault)

   For the 'compare yourself' prototype users couldn't process the feedback in real time playback speed, and for the guidance prototype it was perceived as useful to view a static line, a simplified representation of the somersault over its whole duration. Additionally, simplification does not imply loss of information to users: in the interaction prototype six frames at key moments of the somersault were perceived as sufficient for reflection.

2. **Benchmarked comparison:** any form of comparison is more valuable when done against a (human) rated set of somersaults

   For the 'compare yourself' prototype, it took users quite some effort to make sense of the comparative element, proposing it would be more useful to have a more explicit skill level as comparison (note: this was implemented as a star rating but users made no mention of this).

3. **Beware of data without context:** giving data directly as a means of feedback should be avoided, this is hard to interpret

   Similar to comparing against a random somersault, providing data is only useful when it can be contextualized and interpreted. During the gamification prototype users were critical about the metrics and how insightful they were. For example: rotational velocity was not perceived as a useful input for reflection.

4. **Visual references add interpretability to data:** the problem of interpretation (mentioned above) can be bridged by adding visual aids.

   An example being horizontal and vertical grid lines in order display angle values with a visual reference. For the interaction and numerical/direct feedback prototypes users found it difficult to interpret what the angles meant, and at times with respect the angle value was meant to be. Additionally, the jump 'height' parameter from the gamification prototype didn't mean much numerically, though users indicated that a more visual approach or to showing this would make reflection easier and more concrete to them.

5. **Balance holistic and specific feedback:** a fully holistic approach is an information overload, whereas very specific feedback cannot be translated into useful steps for improvement

   The 'compare yourself' prototype is an example of too much information to interpret: showing full body, two somersaults and real time playback speed all at once. On the other hand, very specific parameters such as jump height (gamification) or velocities of the body (numerical/direct feedback) were too difficult for users to translate into specific points of improvement.

4.5. Hi-Fi Prototyping

Prototypes of higher fidelity are more potent in eliciting user needs [52]. Using the insights from the Lo-Fi evaluations, four high fidelity (Hi-Fi) prototypes were made. The goal of these prototypes is to learn about the value of OpenPose data for the provisioning of feedback for a somersault, taking an approach such that the prototype's underlying (more general) concepts and implications for OpenPose usage can be distinguished and reflected on. This section describes the process of implementing, testing, analyzing and reflecting on the user evaluation of four Hi-Fi prototypes.

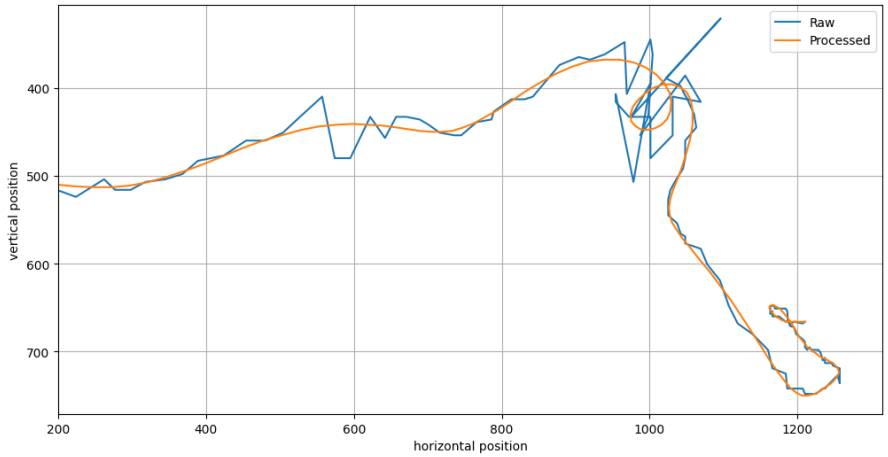
The prototypes use OpenPose's (JSON) data in combination with video and present four different feedback designs, programmed using the Python programming language. The source code can be found in Appendix D. The recordings and OpenPose data of eleven side view somersaults, performed by the same athlete, are used as input for the prototypes. Each prototype uses different somersaults, thus preventing learning effects in the user's reflection on the somersault's performance. The somersaults each have a rating (determined in advance of user evaluation) based on the (averaged) independent judgement of three gymnastics trainers. These ratings are on a scale of 1-3 in which '1' indicates a poor somersault, whereas '3' indicates a good somersault. The ratings are used by prototypes that incorporate comparisons between somersaults. The prototypes all were built on top of a video window that shows a static somersault frame, and frames can be navigated by using a slider at the top

of the interface or a keyboard. The 'Body Angles' and 'Ghost Comparison' prototypes were implemented without data enhancement, whereas the 'Body Part Tracking' and 'Center of Mass' prototypes did utilize enhanced data. This was done in order to explore the challenge of poor OpenPose performance using both a user centered and technical approach.

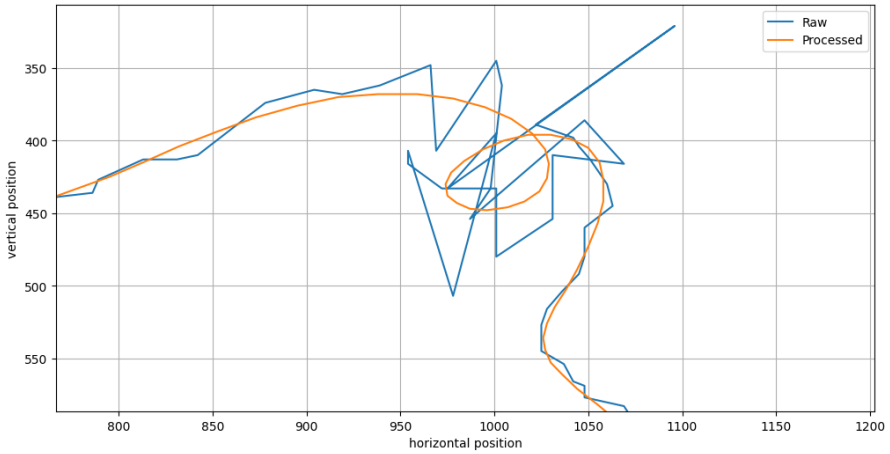
In order to account for the poor performance of OpenPose during the tuck phase (elaborated on in section 4.1), two approaches to this challenge were taken. The first approach was to leave data as is (allowing poor data coverage) and consider solutions from a user centered design perspective. That is, coming up with designs that do not require good data in order to be effective. Secondly, and further elaborated here, is the approach of processing the data. Various steps were taken in order to improve the quality of data before usage in the prototypes:

- Detection of missing (i.e., not containing keypoint values) data points
- Interpolation of missing data points using high degree polynomial curve fitting
- Low pass filtering, this accounts for false detections by OpenPose that show up as high frequency noise (i.e., false detections caused large 'jumps' in coordinate values)

An example comparison of the raw and processed data is given below in figure 4.13.



(a) Full trajectory of right elbow



(b) Trajectory of the right elbow during aerial rotation

Figure 4.13: Comparison of raw and processed OpenPose data for the trajectory of a somersault, showing the right elbow as an example

4.5.1. Description of Prototypes

In this subsection each of the four prototypes is described, along with a snapshot of its functionality.

Body Angles
This prototype shows angles of a body angle for each frame together with a grid overlay. Three keypoints can be selected and the angle between these will be calculated. During the evaluation the knee angle was (arbitrarily) chosen. Its underlying goal is to investigate how useful OpenPose data is for presenting a quantitative metric at a single point in time. Thereby adhering to the insights from the Lo-Fi evaluation: temporal simplification (a single moment in time is considered) and giving a visual reference (a grid) to the angle such that the angle is not a standalone value but can be related to perspective.

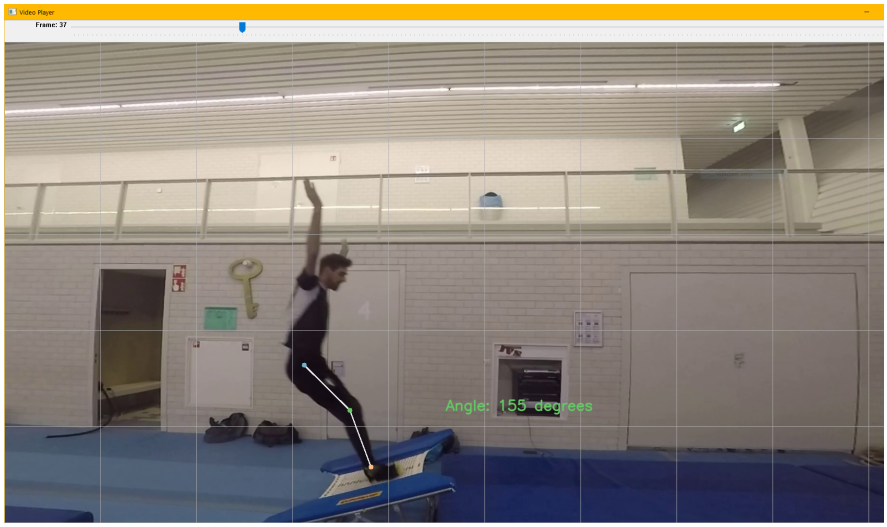


Figure 4.14: Body Angles prototype: shows the angle of of a body part for each frame

Ghost Comparison

The ghost comparison prototype allows a user to compare their current somersault against a 'ghost' (semitransparent overlay) of another somersault, it is an iteration on the Lo-Fi prototype 'compare yourself' (described in 4.4.1). The somersaults are temporally synced on the moment of trampoline takeoff. Temporal synchronization is desirable since it allows for a direct comparison between the two somersaults at the same moment in time. The interface displays individual frames of the somersault, and allows the user to choose any frame in order to compare the somersaults. In this prototype, the ghost is a somersault with the highest rating (three), thus facilitating comparison against an exemplary somersault, which was suggested to be more useful (see 4.4.2) as feedback. This prototype invites feedback from users on data opportunities for feedback provisioning.

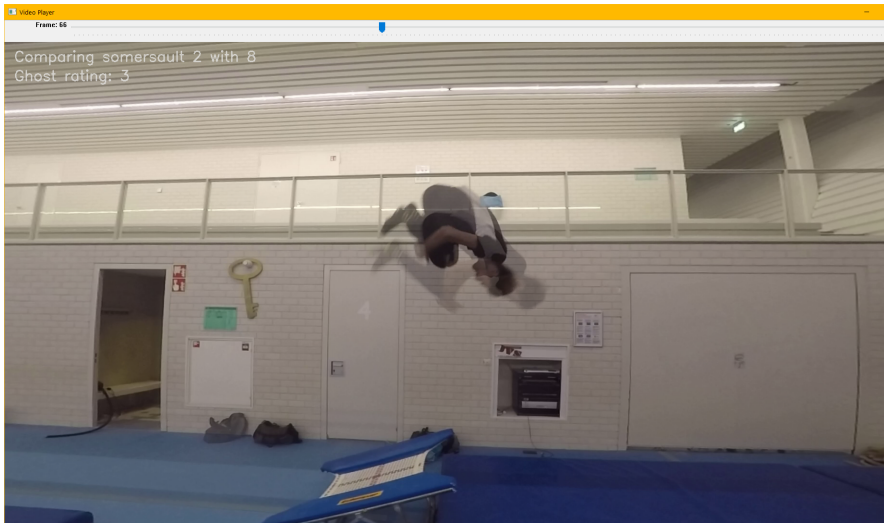


Figure 4.15: Ghost comparison prototype: shows a 'ghost' overlay of another rated somersault on next to the current somersault

Body Part Tracking

An (arbitrary, adjusted to preference during evaluation) body part is tracked and its trajectory during the full somersault is shown as an overlayed line on the video frame. The entire line is visible regardless of the current selected frame. It is compared with the trajectory of the same body part of another somersault with the highest performance rating. The body part tracking prototype is a combination of a bench marked comparison that strives to give a clear visual reference relating to the performance of the somersault. Whereas the ghost comparison prototype is considered holistic, the body part tracking is specific, focusing on a single body part represented by a simple visual.

Figure 4.16: Body part tracking prototype: shows the trajectory of a body part (in this case: right shoulder) of the current somersault and a rated comparison

Center of Mass
The center of mass (CoM) prototype calculates the athlete's center of mass using the segmentation method [53]. This method considers the body to consist of segments, each having an average mass percentage, position and inclination. Based on this information, the center of mass of the athlete is calculated. The CoM's of the current frame and 75 previous frames are shown by a dot. The CoM prototype strives to balance between very specific (body part tracking) and holistic (ghost comparison) feedback by incorporating a metric (the CoM) that is a representation of the whole body's form.

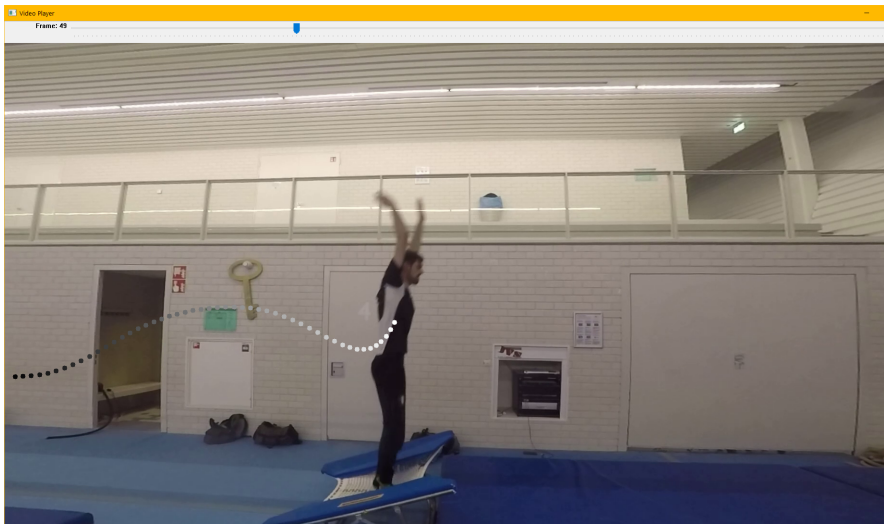


Figure 4.17: Center of mass prototype: calculates and displays the center of mass of the athlete for the current and previous frames

4.5.2. Evaluation Methodology

The prototypes were evaluated within subject with adult gymnasts (n=4) of varying ages, level of skill and occupation within gymnastics (gymnasts, trainers and judges). No additional selection criteria were given, stimulating diversity among participants in order to get a wide range of feedback on the prototypes. Participants were shown the prototypes one by one (in randomized order) and were asked to complete some tasks whilst thinking out loud (cognitive walkthrough) and answer specific questions related to the prototypes' value and utility. In a 25 minute long session all audio was recorded and transcribed clean verbatim. Additionally, non verbal cues were observed and noted by the researcher. The evaluation protocol can be found in Appendix A.1.5. The transcript of the evaluations was coded based on thematic differences as suggested by [51], [54] and [55]. Subsequently, the transcripts and observations were discussed with an emphasis on the concept of using data from OpenPose as a tool for feedback on a somersault. An overview of the evaluation is given below.

- **Research Question:** How can OpenPose data in combination with video be used for feedback on a somersault?
- **Sampling and participants:** Adults gymnasts (n=4)
- **Timespan:** 25 minutes

- **Location:** Closed room at gymnastics hall
- **Evaluation Methodology:** Within subject evaluation, thinking out loud in a cognitive walkthrough with specific questions
- **Data Acquisition:** Audio recording and written observations
- **Data analysis:** Thematic coding of a clean verbatim transcript, triangulated with written observations as input for discussion performed by researcher

4.5.3. Observations

Body Angles
For the body angle prototype it was noted that participants spent most of their time to reflect on the feedback at the trampoline takeoff and landing of the somersault. More generally, participants were particularly interested in any moment of transition between the somersaults phases as described in 2.2. During the same phase (i.e., between the first and last frame of a phase), frames were considered for a shorter period of time. An exception being phase 4 during which the aerial rotation takes place.

Ghost Comparison
Participants scrolled through the frames a lot for this prototype, often going back and forth over the duration of the whole somersault. At similar frames participants spent significantly more time in order to observe differences. Often participants moved physically closer to the screen for inspection and pinch their eyes while trying to affirm their observations with the researcher.

Body Part Tracking
For this prototype, participants didn't interact much with the frame selection controls of the prototype. Most time was spent at the initial frame. The scrolling was minimal, and contrary to the 'Body Angles' and 'Ghost Comparison' prototypes, the amount of reflection depended little on the state of the current frame.

Center of Mass
It was observed that participants had difficulty interacting and reflecting with this prototype; it took them longer to answer reflective questions whilst being less vocal overall (as can be observed also from the transcript).

4.5.4. Transcript

This section presents an overview of comments by users during the evaluation of the Hi-Fi prototypes.

Body Angles
Users stated they focused on two specific phases (being takeoff and landing, which was also found through observations in 4.5.3) because the angle of the knee was considered most useful at these moments. A straight, slightly tilted forward, posture is an indicator of a good takeoff and slightly bent knees underneath an upright core are an indicator of a stable landing. Additionally, the knee angle during rotation was valued as an indicator of tuck form.

On the other hand, participants expressed preference for other angles than that of the knee. Examples being the angle(s) of the entire body during takeoff and the angle of the core's inclination upon landing contact. Body angles were stated to be useful for comparisons and objective adjudication purposes, where the most the relevant angle does depend on the phase of the somersault.

Ghost Comparison
Users were divided on the visual clarity and value of this prototype, some stated they found it easy to distinguish where the somersaults differed, others found it challenging to tell apart the somersaults and not confuse the ghost with motion blur during phases of similarity. During evaluation users were surprised by the similarity between the somersaults up until the phase of extending for landing. It was also stated that initially, even when somersaults differ significantly (and are thus more easily perceived as visually distinct), it can be hard to pinpoint technical differences between somersaults. Users needed more time to process the feedback yet were afterwards able to tell in detail about the performance of somersaults. Lastly, it was suggested that it would be helpful to show 'green' and 'red' margins indicating the correct and incorrect form.

Body Part Tracking
Users were unanimously positive about this prototype. Without questioning, users were able to tell which one was the best somersault, whilst giving details about the run up, achieved jump height, and angle of inclination towards landing. Its simplicity was appreciated by users, stating 'You (the feedback) can add a lot of measurements but in my experience it is pretty challenging already to take specific feedback into account for improvement of a performance.' Similar to the body angles prototype, the body part of interest does matter, though this was less of a deciding factor for this prototype. Additionally, it was mentioned that this feedback would be very useful for training for specific points of improvements. For example, aiming for a certain jump height (compare the body part line to a virtual reference) or improving the stick of the landing (considering the amount of moving around compared to a still reference point upon landing). Users could not mention points of improvement for this prototype when asked.

Center of Mass
The evaluation questions were found to be a lot harder to answer and it took more time for users to reflect on this feedback. Additionally, all users to some extent expressed they did not know how to translate the CoM information into tangible feedback and also pointed out that the center of mass is different between male and female, believing this makes feedback from it more difficult to generalize. Adding a comparative element '...does not account for the fact that I still don't think the center of mass is very useful'. Users understood what the center of mass is, but didn't value it during this evaluation due to its complex interpretability.

4.5.5. Discussion of Hi-Fi prototype evaluation

Body Angles
Considering the usage of OpenPose for somersault feedback, the body angles prototype provides two insights. Firstly, when OpenPose's data is not accurate over the entire time span of a movement (as is true for the somersault), it can still provide valuable feedback, given that

key moments of the movement can accurately be represented. Specifically for showing the knee angle, landing and takeoff were the most insightful moments to users, which both are phases at which OpenPose performs well. This implies that (extensive) data processing is not necessarily a requirement for OpenPose as feedback tool. Secondly, metrics represented by data from OpenPose (here: angles) can be merged with the same metric of the environment in order to enrich feedback. In this case it is known that the core angle of inclination towards landing (i.e., angle of body relative to horizontal ground level) was found to be a useful metric for judging somersault performance. This indicates that the combination of a core angle with respect to the angle of a landing surface (environment) provides more insight than the core angle on itself.

Ghost Comparison
In order to yield valuable feedback, the ghost comparison prototype can make use of OpenPose data in a number of ways. For comparisons against oneself, the (processed) OpenPose data can be compared between somersaults and provide a measure of similarity. Distinguishing differences is something that users struggled with during evaluation. A set of rated somersaults can enable a better way of comparing one's posture against that of a high rated somersault. Over time, it then becomes possible to make good (high data correlation) and bad (low data correlation) phases of a somersault explicit in feedback. Additionally, overlaying, showing and color marking OpenPose coordinates of different jumps improves clarity.

Body Part Tracking
The lines of the body part tracking prototype were received well for a number of reasons. First of all, its feedback was perceived as simplistic and direct. It did so by simplifying the whole somersault into a single trajectory, making it accessible for interpretation to users. Furthermore, showing the entire trajectory regardless of frame allows the user to have a complete temporal overview of their somersault. This allows for relational reflections (for example, observing a more height means expecting a straighter landing). Provided that the data is properly accounted for at phases of poor performance, OpenPose does allow this simplification. In visual terms, avoiding modest clarity, and in terms of reflection by pushing forward a specific parameter (body part) to focus on for the user. The OpenPose enabled simplifications of the somersault have the added benefit of making comparison more insightful by avoiding information overload.

Center of Mass
Using OpenPose to calculate new metrics for feedback is a possibility in the realm of giving feedback. However, the fact that such processes are possible does not imply they add value. A parameter like the center of mass is not explicitly apparent when looking at footage. Based on this prototype evaluation, it is hard to incorporate knowledge of a center of mass in combination with video as feedback. There is a mismatch in the form and abstraction of feedback.

4.5.6. Conclusions of Hi-Fi prototypes

In this section four high fidelity prototypes have been evaluated, each striving to enrich video feedback of a somersault using data from OpenPose. Concluding, a number of design

opportunities can be distinguished:

1. **OpenPose can provide feedback aid even when data coverage is incomplete/inaccurate**

   Firstly, interpolation and filtering during time frames of poor performance are an effective method for improvement of data accuracy for a somersault (see figure 4.13). Additionally, during the somersault each phase (or frame) is indicative of a larger time frame of performance. Thus, even when reliable data is missing during certain time frames, OpenPose data can be used as indication for overall performance. For the somersault, analyzing the moment of takeoff implies a lot about how fast the rotation will be set off, how much height will be achieved, etc.

2. **OpenPose data is more valuable for feedback when taking the environment into account**

   Richer feedback can be provided if information of body parts is considered relative to the environment instead of relative to each other, for example comparing body angles with respect to a vertical reference or the surface of landing.

3. **OpenPose enables the provisioning of a holistic temporal overview of a movement**

   As the body part tracking prototype demonstrates, OpenPose can enable feedback that captures an entire somersault in one frame. Provided that the feedback is kept simple (with in this case a line representing a body part's trajectory), this gave users a lot of valuable insights about the somersault as a whole . OpenPose can enable reflection that goes beyond a single moment in time or single movement. Rather, it can span an entire action or even a multitude of actions over time at once.

4. **OpenPose enables simplification of a complex movement**

   It was found that the trajectory of a somersault can be simplified into a line with no apparent information loss (through the 'Body Part Tracking' prototype). Users were able to reflect better since presented information helped to focus on a specific point of improvement. OpenPose enables highlighting the essence of a movement and presented merely that performance metric to the user for reflection, making cognitive load low and feedback tangible.

5

Discussion & Conclusion

This thesis explores the value of OpenPose data combined with video in the provisioning of feedback for a somersault. Part of this was tackling the challenge of dealing with incomplete OpenPose data. The research was led by the research question:

How can OpenPose data complement footage in providing feedback for a somersault?

Initially, the functionality of OpenPose, other motion capture techniques, strategies for dealing with incomplete (movement) data and data-driven sports training approaches were explored in a literature research. Subsequently, the somersault and common approaches to learning the somersault were characterized. Then, the performance of OpenPose on somersaults was analyzed qualitatively as well as quantitatively. After a diverging and converging design phase, six feedback design ideas were converted into low fidelity (Lo-Fi) prototypes and evaluated with users. These user evaluations were used as input for the development of four high fidelity (Hi-Fi) prototypes that were again evaluated with users. Based on user observations and a transcript from the Hi-Fi prototype evaluation, a number of design opportunities for using OpenPose in a sports context were found. In the remainder of this chapter the contributions and implications of this work are discussed, as well as suggestions for future research and an overall conclusion.

5.1. Contributions

The first contribution of this work is a detailed overview of the performance of OpenPose for complicated sports movements, based on the case of a somersault. Qualitative as well as quantitative insights are provided for the entire time span of the somersault. OpenPose has very limited performance on tucked and upside down poses, which is in line with existing literature. The unique contribution of this work lies in a concrete quantisation of this limitation.

Another contribution is providing a novel perspective on usage of data for feedback; current research focuses on improving the technical performance of applications that use data from OpenPose [5] [12] [13] [15] [16] whereas in this study a user centered approach is adopted. The Body Angles prototype (see 4.5.1) shows that the evaluation of a somersault does not require accurate and complete data coverage thereof. Users can predict or trace back their

form during the aerial phase based on posture at takeoff and landing. The human ability of temporally interpolating the spatial qualities of a motor movement was found to be reliable by [56]. Altogether, this study suggests that a well designed feedback representation of key moments in time (in this case: takeoff and landing) is sufficient for users to reflect on their performance. It is expected that this insight generalizes to other complex sports movements (i.e., OpenPose performs poorly on these movements) as movements can be characterized by key moments in time [57] [58]. Thus, complete data coverage of a movement of interest is not a requirement for the usage of OpenPose in sports feedback design.

On the other hand, it is preferable to have as much data coverage as possible in order to enlarge the technical opportunities for a feedback design. A method is proposed for processing poor or missing OpenPose data. A tucked rotating pose was found to be very challenging for the OpenPose system to accurately evaluate (see figure 4.2, 4.3, 4.4). Based on the thorough performance analysis in this study and the findings of [5] [12] [13] [59] , other rare human poses are expected to be evaluated poorly as well. However, interpolating missing data using a high degree polynomial function and subsequently low pass filtering the data resulted in an effective approximation of the body's keypoints during a somersault. This methodology enabled users to evaluate performance also during phases of poor initial data coverage.

Additionally, the design process used during this study can be applied by other researchers as well. In this research, end users were actively involved in the exploration, brainstorming and evaluation phase of the design process. The active involvement of end users throughout the design process ensures that the resulting designs and prototypes are better adapted to meet user's needs and preferences [60] [61]. Engaging users throughout the design process allows researchers to gain valuable first-hand insights about challenges and opportunities of an envisioned design [51].

Finally, the Hi-Fi prototype Body Part Tracking brought forth (see 4.5.1) two main contributions. Firstly, this feedback allows gymnasts to comprehend the relevant spatio-temporal properties of their somersault in a single frame. Thus, representing relevant properties for reflection in a novel compact dimension. This is supported by feedback from users, who considered the prototype insightful. In a broader sense, all sports movements or actions can be represented as the movement of a body (part) or object over time [62]. Thus the successful design for the somersault can inspire researchers and designers in other sports as well. Besides the compact spatio-temporal representation of the somersault, the Body Part Tracking prototype was found to be successful for its simplification. This was done by focusing on a single body part and by keeping a minimalistic interface. Cognitive overload hinders the ability to learn from reflection [57] [63] [64]. Altogether, a compact representation of data can aid effectiveness of feedback and this can be achieved without cognitively overloading the user. Two requirements for this approach arise. A representative simplified parameter of the movement (here: trajectory of an essential body part) should be distinguished, and the design of feedback should exclusively utilize direct input from this parameter.

5.2. Implications & Future Work

In order to account for gaps in the data of OpenPose, an effective processing methodology was used. Though promising results are provided here, it is not certain how well said methodology

would hold up for more stochastic trajectories. Despite a lot of erroneous data points, the overall trajectory of a(ny) bodypart in a somersault is reasonably predictable. Physics teaches that the netto displacement of a frictionless free falling body should follow a parabolic trajectory [65]. This could explain why polynomial curve fitting is an effective method of data interpolation. However, it is unknown whether this curve fitting is equally effective in sports for which the keypoints of interest vary more stochastically in time. Additionally, different motion capture techniques yield different data. It is unknown whether this processing methodology would hold for alternative MoCap systems. This study implies that for future research, a preliminary data exploration would benefit choosing an effective data processing methodology.

Furthermore, it would be interesting to see how well the design opportunities proposed in section 4.5.6 transfer to different sports movements. The design opportunities are intentionally formulated on a conceptual level, with the goal of making findings of this research inspiring to sports technology design outside of the somersault and gymnastics scope. Literature suggests that other sports disciplines would benefit from these design opportunities. However, this assertion remains theoretical, and further research is needed to determine its practical value. One way to research this could be done by studying a different case of a complex movement (for example, a pole vault). After a cotextual exploration (i.e., definition of the movement, common learning strategies etc.), the design opportunities as proposed in this study, can be used as guidance for feedback design. The generalizability of these principles can then be evaluated based on the effectiveness of the new feedback designs.

Another recommendation for future works is exploring the the motor learning process using proposed feedback strategies. This study is an initial exploration of feedback design possibilities with OpenPose. Current research suggests that feedback directed directly at mistakes aids motor learning [66] [67]. However, the feedback designs in this study are neutral in nature. In other words, they do not judge the performance themselves. Rather, they are a tool that provides input for a user to evaluate their performance with. Future studies should be done on the effectiveness of this type of feedback for the actual motor learning process. As mentioned by one of the participants during evaluation: (see A.1.5):

"...noticing something [about your somersault performance] does not necessarily imply that you also have the capability to change this."

Just because a certain type of feedback seems useful to users does not imply that it transfers into effective improvement of motor learning.

5.3. Limitations

As mentioned in the implications & future work section, the feedback designs in this study do not judge performance. This limits the value of feedback to the user's own knowledge on somersault performance. Users without any prior knowledge or experience in performing somersaults may need assistance in order to interpret feedback.

Another limitation of this research is that all feedback design had a similar visual nature. Essentially all designs consist of an overlay on top of a video frame. This did cover several design opportunities, yet similarities between designs might have limited the full range of

possibilities. Especially the Body Part Tracking (4.5.1) and Center of Mass (4.5.1) prototypes were quite similar and this was a missed opportunity for exploring the design space using diverse prototypes.

The user evaluation is subject to a few limitaitons as well. The selection of participants for evaluation was subject to only two criteria (adult and gymnast). This was done for two reasons: increasing the range of perspectives during evaluation and to increase sample size. However, this means that the results of this study may not be as valuable for different target groups. For example, children would likely prefer gamified setups and top tier athletes might actually prefer more detailed information. Furthermore, the evaluations with users were done during gymnastics training time. However, evaluation input might have been more authentic if the users were reflecting during actual somersault practice. During this study, a prerecorded set of somersaults was used instead. Implementing a live measuring setup with OpenPose would take considerably more time and effort than the current approach. Due to time constraints this was not realized, consequently limiting the evaluation to a setting and time in which an athlete might not have been fully immersed in the movement of a somersault. On the other hand, this is a realistic setting for gymnastics coaches and adjudicators, who play a part in giving feedback.

5.4. Conclusion

In conclusion, this study demonstrates the potential of using OpenPose data with video footage to provide feedback for somersault performance. It provides a concrete overview of the performance and limitations of OpenPose on the somersault movement. Through an iterative design process, four high fidelity prototypes were developed and evaluated with gymnasts. This revealed valuable insights into the effectiveness of such feedback. Among others, this research finds that complete data coverage is not essential for meaningful feedback, as a few key moments in a movement can be enough for users to reflect. Additionally, an effective implementation of data interpolation is proposed. Furthermore, the study underscores the importance of simplifying feedback to avoid cognitive overload, thereby enhancing its practical utility. While the methodology and findings are promising, further research is needed to explore the generalizability of these insights to other sports and to investigate the direct impact of these feedback strategies on motor learning. The study also identifies limitations of the feedback designs and the evaluation context, subsequently suggesting directions for improvements and future research.

References

[1] M. Menolotto, D.-S. Komaris, S. Tedesco, B. O'Flynn, and M. Walsh, "Motion capture technology in industrial applications: A systematic review," *Sensors*, vol. 20, no. 19, p. 5687, 2020.

[2] R. Z. Ye, A. Subramanian, D. Diedrich, H. Lindroth, B. Pickering, and V. Herasevich, "Effects of image quality on the accuracy human pose estimation and detection of eye lid opening/closing using openpose and dlib," *Journal of Imaging*, vol. 8, no. 12, p. 330, 2022.

[3] D. Osokin, "Real-time 2d multi-person pose estimation on cpu: Lightweight openpose," *arXiv preprint arXiv:1811.12004*, 2018.

[4] T.-Y. Lin, M. Maire, S. Belongie, *et al.*, "Microsoft coco: Common objects in context," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., Cham: Springer International Publishing, 2014, pp. 740–755, ISBN: 978-3-319-10602-1.

[5] T. Kitamura, H. Teshima, D. Thomas, and H. Kawasaki, "Refining openpose with a new sports dataset for robust 2d pose estimation," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022, pp. 672–681.

[6] B. G. Bardy and M. Laurent, "How is body orientation controlled during somersaulting?" *Journal of Experimental Psychology: Human Perception and Performance*, vol. 24, no. 3, p. 963, 1998.

[7] D. B. Postma, R. W. van Delden, J. H. Koekoek, *et al.*, "A design space of sports interaction technology," *Foundations and Trends® in Human–Computer Interaction*, vol. 15, no. 2-3, pp. 132–316, 2022.

[8] A. Atiković, A. N. Mujanović, J. Mehinović, E. Mujanović, and J. Bilalić, "Effects of a mini-trampolone exercise during 15 weeks for increasing the vertical jump performance.," *Sport Scientific & Practical Aspects*, vol. 15, no. 1, 2018.

[9] M. D. P. Mr. Horst Kunze Mr. Miguel Vicente, "2022 – 2024 code of points for trampoline gymnastics," Fédération Internationale de Gymnastique, 2021.

[10] *Beter turnen - turnoefeningen voor turntrainers en turnsters*. [Online]. Available: `https://beterturnen.nl/koprol-checklist/?utm_source=ActiveCampaign&utm_medium=email&utm_content=Hier%2Bis%2Bje%2Baangevraagde%2Bchecklist%21%F0%9F%98%83&utm_campaign=Hier%2Bis%2Bje%2Baangevraagde%2Bchecklist&vgo_ee=32YtGgs9HYGOTRBwZ%2Bt1%2FaOf27FJyeAacbmt%2BpemOSmLAa138Hk%3D%3Adxonh8BAvPRBaFNTmhyFSR0HH66qB%2F%2B3`.

[11] G. Hidalgo, Z. Cao, T. Simon, *et al.* "Openpose: Real-time multi-person keypoint detection library for body, face, hands, and foot estimation." (2017), [Online]. Available: `https://github.com/CMU-Perceptual-Computing-Lab/openpose`.

[12] K. Toyoda, M. Kono, and J. Rekimoto, "Post-data augmentation to improve deep pose estimation of extreme and wild motions," in *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, IEEE, 2019, pp. 1570–1574.

[13] R. M. Kanko, E. K. Laende, E. M. Davis, W. S. Selbie, and K. J. Deluzio, "Concurrent assessment of gait kinematics using marker-based and markerless motion capture," *Journal of biomechanics*, vol. 127, p. 110 665, 2021.

[14] L. Wade, L. Needham, P. McGuigan, and J. Bilzon, "Applications and limitations of current markerless motion capture methods for clinical gait biomechanics," *PeerJ*, vol. 10, e12995, 2022.

[15] X. B. Peng, A. Kanazawa, J. Malik, P. Abbeel, and S. Levine, "Sfv: Reinforcement learning of physical skills from videos," *ACM Transactions On Graphics (TOG)*, vol. 37, no. 6, pp. 1–14, 2018.

[16] N. Carissimi, P. Rota, C. Beyan, and V. Murino, "Filling the gaps: Predicting missing joints of human poses using denoising autoencoders," in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018, pp. 0–0.

[17] Z. Zhang, "Microsoft kinect sensor and its effect," *IEEE MultiMedia*, vol. 19, no. 2, pp. 4–10, 2012. DOI: 10.1109/MMUL.2012.24.

[18] D. Roetenberg, H. Luinge, and P. Slycke, "Xsens mvn: Full 6dof human motion tracking using miniature inertial sensors," *Xsens Motion Technol. BV Tech. Rep.*, vol. 3, Jan. 2009.

[19] R. W. Bohannon, S. Harrison, and J. Kinsella-Shaw, "Reliability and validity of pendulum test measures of spasticity obtained with the polhemus tracking system from patients with chronic stroke," *Journal of NeuroEngineering and Rehabilitation*, vol. 6, no. 1, 2009. DOI: 10.1186/1743-0003-6-30.

[20] N. Ghorbani and M. J. Black, "Soma: Solving optical marker-based mocap automatically," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2021, pp. 11 117–11 126.

[21] M. Bentley, *Wireless and visual hybrid motion capture system*, US Patent 9,320,957 B2, 26 2016. [Online]. Available: https://patents.google.com/patent/US9320957B2/en.

[22] M. R. Andersen, T. Jensen, P. Lisouski, *et al.*, "Kinect depth sensor evaluation for computer vision applications," *Aarhus University*, pp. 1–37, 2012.

[23] Z. Zhang, "Microsoft kinect sensor and its effect," *IEEE multimedia*, vol. 19, no. 2, pp. 4–10, 2012.

[24] B. Langmann, K. Hartmann, and O. Loffeld, "Depth camera technology comparison and performance evaluation.," in *ICPRAM (2)*, 2012, pp. 438–444.

[25] R. Avetisyan, "Depth camera calibration," *University of Rostock, Yerevan, Armenia, unpublished*,

[26] N. Ahmad, R. A. R. Ghazilla, N. M. Khairi, and V. Kasi, "Reviews on various inertial measurement unit (imu) sensor applications," *International Journal of Signal Processing Systems*, vol. 1, no. 2, pp. 256–262, 2013.

[27] E. D'Antonio, J. Taborri, I. Mileti, S. Rossi, and F. Patané, "Validation of a 3d markerless system for gait analysis based on openpose and two rgb webcams," *IEEE Sensors Journal*, vol. 21, no. 15, pp. 17 064–17 075, 2021. DOI: 10.1109/JSEN.2021.3081188.

[28] M. Lapinski, C. Brum Medeiros, D. Moxley Scarborough, *et al.*, "A wide-range, wireless wearable inertial motion sensing system for capturing fast athletic biomechanics in overhead pitching," *Sensors*, vol. 19, no. 17, p. 3637, 2019.

[29] Q. Hu, L. Liu, F. Mei, and C. Yang, "Joint constraints based dynamic calibration of imu position on lower limbs in imu-mocap," *Sensors*, vol. 21, no. 21, p. 7161, 2021.

[30] N. Ghorbani and M. J. Black, "Soma: Solving optical marker-based mocap automatically," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 11 117–11 126.

[31] L. P. Maletsky, J. Sun, and N. A. Morton, "Accuracy of an optical active-marker system to track the relative motion of rigid bodies," *Journal of biomechanics*, vol. 40, no. 3, pp. 682–685, 2007.

[32] M. Yamamoto, K. Shimatani, Y. Ishige, and H. Takemura, "Verification of gait analysis method fusing camera-based pose estimation and an imu sensor in various gait conditions," *Scientific reports*, vol. 12, no. 1, p. 17 719, 2022.

[33] P. Peursum, H. H. Bui, S. Venkatesh, and G. A. West, "Classifying human actions using an incomplete real-time pose skeleton," in *PRICAI 2004: Trends in Artificial Intelligence: 8th Pacific Rim International Conference on Artificial Intelligence, Auckland, New Zealand, August 9-13, 2004. Proceedings 8*, Springer, 2004, pp. 971–972.

[34] S. J. Howarth and J. P. Callaghan, "Quantitative assessment of the accuracy for three interpolation techniques in kinematic analysis of human movement," *Computer methods in biomechanics and biomedical engineering*, vol. 13, no. 6, pp. 847–855, 2010.

[35] J. Smołka and M. Skublewska-Paszkowska, "Comparison of interpolation methods based on real human motion data," *Przegląd Elektrotechniczny*, vol. 90, no. 10, pp. 226–229, 2014.

[36] J. Camargo, A. Ramanathan, N. Csomay-Shanklin, and A. Young, "Automated gap-filling for marker-based biomechanical motion capture data," *Computer Methods in Biomechanics and Biomedical Engineering*, vol. 23, no. 15, pp. 1180–1189, 2020.

[37] S. Shan, S. Sun, and P. Dong, "Data driven intelligent action recognition and correction in sports training and teaching," *Evolutionary Intelligence*, pp. 1–9, 2023.

[38] M. Gambhir, ""match analysis" using notational analysis and data analytics in table tennis with interactive visualisation," in *PROCEEDINGS BOOK OF THE 16th ITTF SPORTS SCIENCE CONGRESS*, p. 280.

[39] Y. Huang, L. Churches, and B. Reilly, "A case study on virtual reality american football training," in *proceedings of the 2015 virtual reality international conference*, 2015, pp. 1–5.

[40] S. Cmentowski and J. Krueger, "Exploring the potential of vertical jump training in virtual reality," in *Extended Abstracts of the 2021 Annual Symposium on Computer-Human Interaction in Play*, 2021, pp. 179–185.

[41] R. M. Pascoal, A. de Almeida, and R. C. Sofia, "Activity recognition in outdoor sports environments: Smart data for end-users involving mobile pervasive augmented reality systems," in *Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers*, 2019, pp. 446–453.

[42] T. Clegg, D. M. Greene, N. Beard, and J. Brunson, "Data everyday: Data literacy practices in a division i college sports context," in *Proceedings of the 2020 CHI conference on human factors in computing systems*, 2020, pp. 1–13.

[43] N. Hamanishi, T. Miyaki, and J. Rekimoto, "Assisting viewpoint to understand own posture as an avatar in-situation," in *Proceedings of the 5th International ACM In-Cooperation HCI and UX Conference*, 2019, pp. 1–8.

[44] A. Khan, S. Mellor, R. King, *et al.*, "Generalized and efficient skill assessment from imu data with applications in gymnastics and medical training," *ACM Transactions on Computing for Healthcare*, vol. 2, no. 1, pp. 1–21, 2020.

[45] S. Safdarnejad, X. Liu, L. Udpa, B. Andrus, J. Wood, and D. Craven. "Sports videos in the wild (svw): A video dataset for sports analysis." (2015), [Online]. Available: `https://cvlab.cse.msu.edu/project-svw.html`.

[46] A. Atiković, A. N. Mujanović, J. Mehinović, E. Mujanović, and J. Bilalić, "Effects of a mini-trampoline exercise during 15 weeks for increasing the vertical jump performance.," *Sport Scientific & Practical Aspects*, vol. 15, no. 1, 2018.

[47] M. A. King and M. R. Yeadon, "Maximising somersault rotation in tumbling," *Journal of Biomechanics*, vol. 37, no. 4, pp. 471–477, 2004.

[48] A. Holzinger, O. Waclik, F. Kappe, S. Lenhart, G. Orasche, and B. Peischl, "Rapid prototyping on the example of software development in automotive industry: The importance of their provision for software projects at the correct time," in *Proceedings of the International Conference on e-Business*, 2011, pp. 1–5.

[49] J. Rick, P. Francois, B. Fields, R. Fleck, N. Yuill, and A. Carr, "Lo-fi prototyping to design interactive-tabletop applications for children," in *Proceedings of the 9th International Conference on Interaction Design and Children*, 2010, pp. 138–146.

[50] S. Wilkinson, "Focus group methodology: A review," *International journal of social research methodology*, vol. 1, no. 3, pp. 181–203, 1998.

[51] C. Courage and K. Baxter, *Understanding your users: A practical guide to user requirements methods, tools, and techniques*. Gulf Professional Publishing, 2005.

[52] M. Anastassova, C. Mégard, and J.-M. Burkhardt, "Prototype evaluation and user-needs analysis in the early design of emerging technologies," in *Human-Computer Interaction. Interaction Design and Usability: 12th International Conference, HCI International 2007, Beijing, China, July 22-27, 2007, Proceedings, Part I 12*, Springer, 2007, pp. 383–392.

[53] B. Jeong, C.-Y. Ko, Y. Chang, J. Ryu, and G. Kim, "Comparison of segmental analysis and sacral marker methods for determining the center of mass during level and slope walking," *Gait & Posture*, vol. 62, pp. 333–341, 2018.

[54] J. Lazar, J. H. Feng, and H. Hochheiser, *Research methods in human-computer interaction*. Morgan Kaufmann, 2017.

[55] A. Castleberry and A. Nolen, "Thematic analysis of qualitative research data: Is it as easy as it sounds?" *Currents in pharmacy teaching and learning*, vol. 10, no. 6, pp. 807–815, 2018.

[56] N. Ludolph, J. Plöger, M. A. Giese, and W. Ilg, "Motor expertise facilitates the accuracy of state extrapolation in perception," *PloS one*, vol. 12, no. 11, e0187666, 2017.

[57] W. H. Edwards, *Motor learning and control: From theory to practice*. Cengage Learning, 2010.

[58] M. Kaphle, "Simulations of human movements through temporal discretization and optimization," Ph.D. dissertation, KTH, 2007.

[59] 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.

[60] C. Fuchs and M. Obrist, "Hci and society: Towards a typology of universal design principles," *Intl. Journal of Human–Computer Interaction*, vol. 26, no. 6, pp. 638–656, 2010.

[61] N. B. Hansen, C. Dindler, K. Halskov, *et al.*, "How participatory design works: Mechanisms and effects," in *Proceedings of the 31st Australian conference on human-computer-interaction*, 2019, pp. 30–41.

[62] N. Seethapathi, S. Wang, R. Saluja, G. Blohm, and K. P. Kording, "Movement science needs different pose tracking algorithms," *arXiv preprint arXiv:1907.10226*, 2019.
[63] S. Ma, M. Varley, L.-K. Shark, and J. Richards, "Overcoming the information overload problem in a multiform feedback-based virtual reality system for hand motion rehabilitation: Healthy subject case study," *Virtual Reality*, vol. 16, pp. 325–334, 2012.
[64] J. Afonso, J. Garganta, and I. Mesquita, "Decision-making in sports: The role of attention, anticipation and memory," *Revista brasileira de cineantropometria & desempenho humano*, vol. 14, pp. 592–601, 2012.
[65] M. Baće, S. Ilijić, Z. Narancić, and L. Bistricić, "The envelope of projectile trajectories," *European journal of physics*, vol. 23, no. 6, p. 637, 2002.
[66] M. Starzak, M. Biegajło, M. Nogal, *et al.*, "The role of verbal feedback in the motor learning of gymnastic skills: A systematic review," *Applied Sciences*, vol. 12, no. 12, p. 5940, 2022.
[67] N. Marta, "Effectiveness of verbal feedback in complex motor skill learning," *Theory and Practice of Physical Culture*, no. 8, pp. 63–66, 2020.

Appendices

A

Interviews and User Evaluations

A.1. Interviews and User evaluations

A.1.1. Study Information and Consent Form

The the document provided to interviewees and participants for this research, it starts on the next page.

Interactive Technology for Gymnastics

-

Study Information and Consent

Authors: Niels van Dalen
Last edited: 17-01-2024

Who am I and What is this Study About?

Hi, I am Niels and I am studying Interaction Technology at the University of Twente, and am a great sports enthusiast as well. For my Master thesis I am doing research into uses for Interactive Technologies within sports. For this specific study I want to empathize with you on your progress of learning and performing aerial gymnastics, specifically the tucked in forward somersault.

What will taking part involve?

Participating in this research means taking part in a *focus group, interview or gymnastics session* in which we discuss various topics related to learning the somersault. You are expected to be able to formulate (elaborate) experiences and be honest about these things. If this concerns a gymnastics session you're expected to be competent in performing a somersault.

Why have you been invited to take part?

You have been invited because you meet the criteria for my research, which are:

- You are older than 12 years
- You have a gymnastics association membership (or had one last year)
- You are involved with learning and/or improving the tucked in forward somersault in a gymnastics discipline

If you do not meet any of these criteria please let me know and we can discuss the options.

Do you have to take part?

Participation is completely voluntary and stays voluntary throughout. You can reach out to the researcher for any question about participation in advance. Meaning you can refuse any requests or withdraw yourself from participation. Additionally, you can at all times request deletion of your data.

Data and results of the study

All gathered data will be labelled anonymously, yet be aware that certain data (such as a voice recordings or video) is personally identifiable. Data will be stored in the cloud using Google docs. The only person with access to this cloud is me (Niels van Dalen). Data will be retained until the end of 2024, after which it will be deleted. My University supervisor (Dees Postma) will, exclusively under my supervision, be able to view the data as well.

Who to contact for further information?

For any further questions you can contact the following people, in most effective order:

1) Niels van Dalen (main researcher) at n.g.a.vandalen@utwente.nl or 06 8182 1571
2) Dees Postma (thesis supervisor) at d.b.w.postma@utwente.nl
3) University of Twente Ethics committee at ethicscommittee-cis@utwente.nl

UNIVERSITY OF TWENTE.

Consent Form

YOU WILL BE GIVEN A COPY OF THIS INFORMED CONSENT FORM

Please tick the appropriate boxes	**Yes**	**No**
Taking part in the study		
I have read and understood the study information dated 25-05-2023, or it has been read to me. I have been able to ask questions about the study and my questions have been answered to my satisfaction.	○	○
I consent voluntarily to be a participant in this study and understand that I can refuse to answer questions and I can withdraw from the study at any time, without having to give a reason.	○	○
I understand that taking part in the study involves capturing personal data in the form of notes, recordings, and other means during the session. Recording may be partially or fully transcribed.	○	○
Use of the information in the study		
I understand that information I provide will be used for a master thesis report at the University of Twente, where affiliated people (supervisors etc.) will have access the final result of the work based on this data.	○	○
I understand that personal information collected about me that can identify me, such as my voice in recordings, will not be shared beyond the study team.	○	○
Future use and reuse of the information by others		
I give permission for written notes, audio and video recordings (including their transcriptions) and any other inputs I provide during the experiment to be anonymously archived in the cloud and on a local hard drive so it can be used for future research and learning.	○	○

Further participation

By providing contact details I give permission to the researcher to reach out to me for further questions. I am aware that at all times I can refuse any form of cooperation and request deletion of my contact details.

Optional:

Email address:

Phone Number:

Signatures

Name of participant Signature Date

_____ _____ _____

I have accurately read out the information sheet to the potential participant and, to the best of my ability, ensured that the participant understands to what they are freely consenting.

A.1.2. Interview Script

The script used to structure interview with, it starts on the next page.

Interview Script - preferably with Observations

Interview with gymnastics trainer - preferably accompanied with class observation

Introduction & info

- Thank you
- Who am I and why am i here? I am Niels and for my Master thesis I am researching the possibilities of personal interactive data exploration in a sport's learning context, specifically concerning ballistic movements.
- Goal: The goal of my interview is to have an idea of the context in gymnastics training related to the tucked in forward somersault and other comparable aerial movements. What are typical ways of learning and giving feedback on a somersault?
- Consent form + explain/answer questions

Warm up:

- Who are you?
- How long have you been involved with gymnastics (as a trainer/coach/athlete)?
- Favorite part/exercise/domain of gymnastics?

Body:

- How is the forward somersault positioned within gymnastics in general?
- What does a typical/common somersault look like?
- **What are common strategies/approaches to somersault learning?**
    - **What are your current technological aids?**
- **What are challenges/shortcoming in learning/training?**
- How do you go about giving feedback on somersaults?
- What are the demographics of somersault learners (according to coaches)
- **What keypoints do you think are important for feedback?**

Conclusion:

- **Summarize and confirm notes with interviewee**
- Additional relevant info you think I should know?
- May i contact you in a later stage of my prototyping perhaps?
- Thanks again!

A.1.3. Interview Notes

These are the notes taken during interviews with gymnastics trainers and athletes, the interviews were structured as denoted by the script in A.1.2. The interview notes were discussed and where necessary corrected in collaboration with interviewees in order to accurately reflect their views. In total 9 interviews were conducted.

P1

General
SV Wilhelmina, The Hague & X in Enschede
8 years experience
Groups: 4-7, 7-11 jaar DH: 4-12 jaar, meisjes
Recreational & Performance classes

Good somersault:
> big leap into tramp
> arms up
> highest point (good height achieved), sing arms down, pull in legs
> rotate
> when proper (feeling) extend
> land with bent knees, arms to the side and stand still

Ways of training for a somersault:
(on both mats/trampolines:)

1. Headrolls
    a. Make small, grab shins
2. Trampoline + higher mat for landing Zweefrol Tipsalto / high jump on feet on mat (takes longest time)
    > because: lot of different exercises, safety preparation, mental prep confidence
3. Salto: catch / lock arms / decrease support gradually (can also take long)

Common mistakes:

1. Gaining (not enough) height
2. Making small
    a. Angles too fast for the eye
    b. 'Holistic view of the blurry ball in the air' > small or big
        i. Knees to chest
        ii. Hands on shin
        iii. Chin on chest
3. Rotation timing
    a. At highest point upside down
4. Landing (stationary)
    a. Reflects point of straightening out! (fall backwards > earlier, fall forwards > later)
    b. Not only earlier & later but also speed thereof

> above is from turning teacher course, after you develop intuitive feeling for how to teach
It depends also a lot on the person! Different kids need different (amounts of) feedback

Keypoints considered relevant:
0,1,4,7,10,13, 14

P2

General:
Turner since child, Main trainer at Avanti enschede
Classes ages 4-25 years
Around 10-11 years > saltos (because: prerequisite skills for it, and basic movement ‘after’)

Good somersault:
> height
> swing arms
> tuck in small
> land stable

Ways of training for a somersault:
- Jumping trampoline > height
- Headroll compact
- ‘Hurksprong’ > salto posture without rotating
- Elevated mat headroll (increasing height)
- First attempt salto’s

More info: https://beterturnen.nl/
https://dutchgymnastics.nl/ (youtube)

Common mistakes:
Hard: timing (rotation)
Run-up: 100% energetic > balance between height and rotation

KPI: high & fast rotation, head pointed straight down ie, no lateral deviation (head kps possibly important)

Additional:
Huidige Apps: *video delay, coacheye* (dependant also on coach knowledge for effectiveness)
Added value: visual feedback for athletes (realisation),
Current way of feedback: show, help physically (aid turning oid), for higher level/perfectioning: visual (footage), also desired naturally by athletes

Relevant keypoints:
0, 1, 4, 7, 9, 12, 13 , 14
“when standing close it can be too hectic to also observe at the same time”

P3

General:
7 years experience as assistent trainer
From 17 years onwards classes with full responsibility > small children to pubers

Goeie salto:
Big jump into trampoline with both feet
Stretched liftoff
Pull in knees for rotation
(preferably) don't grab knees (this decreases points)
Keep compact for long > allows easier rotation
Landing: slightly bent knees, hands straight and forwards
→ throughout somersault: keep hands close to body and extended whenever possible

Landing: depends on discipline (trampoline=same spot, mat=forward yet centered)

Safety is part of learning process
- Also for prerequisites > important to realize consequences [of bad execution]

Ways of training for a somersault:
Higher levels - somersault is a basic element
Head rool > dive roll > roll to high element (mat) > salto chair > aid / diminishing

Common mistakes/challenges:
Control rotational velocity
No 'feeling' for a good somersault > how to convey this as trainer
Limited time per trainee
Backwards somersault hard to record
It is hard to give proper feedback (unspecified)

Additional:
Videos, reference objects (vertical pole), video edit app, rubber bands, apps are time consuming & focused on trainers themselves, gopro (suggestion)

Relevant keypoints:
0,1, 8, 9,10,11,12,13,14

P4

General:
Trainer for 23 years, Uturn, [other association] and for Eredivisie (professional)
Previously practicing, now retired (age=53)

Good somersault:
- Straight run up
- Balanced jump into/from trampoline (ie. no lateral weight)
- On takeoff: forward momentum is already initiated by: arm swing/slightly bending over upper body forward
- Gain sufficient height (not a req, but helps timing rotation and ‘catching’ the stop of momentum on landing (less rotational speed required)
- Upside down moment is before highest point

Ways of training for a somersault:
- Start with headrolls, downward headrolls to feet
- Elevated headrolls from trampoline
- Tucked jump (straight up)
- Salto with arm support

Challenges/Common mistakes:
- Assessing rotational velocity
- Departing ‘too’ straight up
- Straight legs on landing

Additional:
- Most people are taught wrong; body is not straight up on takeoff but slighlty tilted forward
- position of CoM important indicator for balance as a whole

Relevant keypoints:
0,1, 8, 9,10,11,12,13,14

P5

General:
Practicing gymnastics since 8 years (current 18), 2 years at UTurn
Retired from competitive/selection at 16 (because of studying)

Good somersault:
- ‘must feel good’
- Two legs into tramp
- Grab ankles/knees > keep arms in vicinity thereof
- Chin to chest

Ways of training for a somersault:
- Practice a lot
- Started headrolling
- Salto chair
- Into the foam pit from a trampoline

Challenges/Common mistakes:
- Jumping too far forward
- Not tucking tight enough

Additional:
- Thinks this applies to multiple aerial movements

Relevant keypoints:
0,1,2,4,5,7,10,11, 13, 14

P6

General:
Gymnastics since 17 (current 21), Uturn
recreational

Good somersault:
- ‘Strong’ runup
- Jump and grab shins, chin to chest
- Touch surface with toes and ‘glide’ into stable foot stance
- Hands are to the side or up at landing [vague?]
- Straight back at takeoff and landing, curved inwards in air

Ways of training for a somersault:
- Able to do it as kid, doesn’t remember exactly how learned

Challenges/Common mistakes:
- Jumping too straight (minitramp>mat)
- Over rotation (fear of landing on head)
- Misjudging jump into trampoline (too far/too close)

Additional:
- *“Wouldn’t say I’m very proficient at the somersault”*

Relevant keypoints:
0,1,9,10,11,12,13,14

P7

General:
Trainer at Uturn (2years), practicing since 6 years old
‘low’ level of competitiveness (does matches every now and then)

Good somersault:
- Takeoff with two legs in the center (cross) of trampoline
- Arms straight and up
- During (omhoogveren) swing arms down and bring chin down
- When feet leave trampoline, there is forward momentum, ‘rotating goes naturally’
- ‘Somewhere’ during rotations extend again > couldnt explain, this is ‘feeling’
- Land with slightly bent knees and straight back
- Arms are used to balance landing (laterally and also forward/backward correction)

Ways of training for a somersault:
- ‘No experience with teaching from scratch, most people here can do so and are merely able to improve form’
- As kid: headrolls with (pittenzakjes) between knees and feet and chest <> chin
- Jump and teacher grabs and throws legs over head while supporting at armpits > gradually diminished until only armpit support (rotation on own power)
- Jump saltos (trampoline) with rubber bands that decrease falling speed

Challenges/Common mistakes:
- Timing of rotation
- Landing too ‘harsh’ (kicking out too much (abrupt) with legs or not enough rotation)
- Adapting feedback to inidividuals, posture/length etc matters

Additional:
- Videotaping & pausing useful tool for reflection

Relevant keypoints:
0,1,3,4,6, 7,10,11, 13, 14

P8

General:
Gymnastics since 11 years (current age 25), focussing mostly on ‘rings’ (ringen) at competitive level

Good somersault:
- Fast runup (helps height)
- Hit center of trampoline
- Swing arms down and ‘make small’
- Strecht out halfway through (‘i think highest point’)
- Absorbing landing in squatting position (knees bent, back striaght, arms foward)

Ways of training for a somersault:
- Salto chair > practice jumping with forward rotation without falling
- Doing headrolls downwards onto feet
- [doesn’t remember more]

Challenges/Common mistakes:
- Mispositioning in trampoline
- Tuck/stretch timing
- Land with ‘strong’ posture (often: can be a bit ragdolly)

Additional:
- Somersault as is is rather standard, at older age this is a ‘step-up’ exercise
- Advice: test a lot with people

Relevant keypoints:
0, 1, 4, 7, 9, 10, 12, 13

P9

General:
Gymnastics since 14 (currently 18)
recreational level

Good somersault:
- ‘Envison’ your jump before you do it and be confident about it
- Jump into center of trampoline with both feet
- Throw your legs over your head
- Make small
- ‘Fish’ with your feet for the landing (when: by feeling)
- Landing: not too abrupt > controlled deceleration

Ways of training for a somersault:
- Tipsalto (somersault with hand contact to the landing surface)
- Tipsalto upwards (ie. from trampoline onto an elevated mat
- Above: start landing on but, build up to being able to land at feet
- Somersaults into a foam pit
- On trampoline with person support at side (either just the fall or actively aiding rotation)

Challenges/Common mistakes:
- Dealing with fear
- Proper translational jump length (ie. not too high and also not too far)
- Timing tucking/stretching
- Properly sticking the landing (due to insufficient strength/bad timing)

Additional:
- ‘Things I said might vary depending on somersault type and learning stage’
- ‘I feel that men and women have different challenges and common mistakes for the somersault’ (summary of elaboration: men are more powerful but less controlled, women are typically more controlled but less powerful)

Relevant keypoints:
0,1,2,4,5,7 8,10,11,13,14

Summary:

What does a typical/common somersault look like?

A good salto (trampoline onto mat):

- big leap into tramp (tilted slightly forward)
- arms up
- highest point, sing arms down, pull in legs
- rotate
- when proper (feeling) extend
- land with bent knees, arms to the side and stand still (or arms forward, safety)

What are common strategies/approaches to somersault learning?

General chronological progress:

- Headrolls (from negatively inclined to straight)
- Dive rolls (zweefroll)
- Gaining height for these rolls ie. onto mat (poss. With a jump)
- Salto with aid: salto chair (tuigje van boven), persons catching/guiding on the side
- Practicing jump (straight tucked jump)

Additional

- Circuit with partial exercises (see above.. Or this beterturnen link)
- Focus on tangible concepts/tricks/aids (ie. practice chin to chest/grabbing knees for later on enough rotational velocity > specific instruction with more abstract goal: improve rotational velocity)

What are your current technological aids?

- (annotable/drawable) video (frames)
- Apps: video delay, coach eye

Other Aids

- vertical pillar/pole in order to reference straight line(s)
- Rubber bands/salto chair etc.

What keypoints do you think are important for feedback?
Mentioned key points overall to be deemed important (i.e. mentioned by >50% of participants)

0	**Xxxxx xxxx**	**Nose**
1	**Xxxxx xxxx**	**Neck**
2	xx	RShoulder
~~3~~	~~x~~	~~RElbow~~
4	**Xxxxx x**	**RWrist**
5	xx	LShoulder
~~6~~	~~x~~	~~LElbow~~
7	**Xxxxx x**	**LWrist**
8	xxx	MidHip
9	**Xxxxx**	**RHip**
10	**Xxxxx xxx**	**RKnee**
11	**Xxxxx x**	**RAnkle**
12	**Xxxxx**	**LHip**
13	**Xxxxx xxx**	**LKnee**
14	**Xxxxx xx**	**LAnkle**
15		REye
16		LEye
17		REar
18		LEar
19		LBigToe
20		LSmallToe
21		LHeel
22		RBigToe
23		RSmallToe
24		RHeel
25		Background

A.1.4. Lo-Fi prototype evaluation protocol

Lo-Fi Prototype evaluation protocol

This document is a guide for the evaluation of Lo-Fi prototypes, it concerns personal notes and prototype specific questions.

Make very clear participants can be honest and criticism is appreciated. I have no connection or relation with these ideas (you don't offend me if you criticize)

General Evaluation Questions[1]

English	**Dutch**
Do you understand what is going on?	Begrijp je wat je voor je hebt?
Explain what you are seeing/doing	Leg uit wat je ziet/aan het doen bent
How do you feel about this?	Wat denk je als dit zo ziet?
Can you think of a scenario in which this would be helpful to you?	Kom jij in scenario's waarin jij hier wat aan zou hebben?

Time management guideline

Introduction & signing consent	4 min
Prototype evaluation	6x3min each = 18min
Concluding (final remarks, verifying notes etc.)	4 min
Total:	**26 minutes**

[1] As proposed by K. Baxter Understanding your Users

1. Gamification: Highscores

- Show highscores after a somersault
- Metrics: max height, rotational velocity, stick
- variations: numerical, symbolic (adult), symbolic (child)

Explain to user:

- You will get to see three video
- Share your thoughts

Check if answered:

- What was your thought when a high score was reached?
- How did you feel about the metrics? (the variables that were chosen?)

P1	
P2	
P3	
P4	
P5	
P6	

2. Numerical/Direct Feedback: Values at Location

For 5 key phases of somersault (we do 6 frames, 2 for landing)
Menu with:

- Velocity
    - Horizontal velocity
    - Vertical velocity
    - Rotational velocity
- Angle
    - Neck
    - Arms
    - Core-legs
    - knees

Explain to user:

- Pretend the screen is touch screen
- I will play the computer (i.e., respond to your input), please give me some time

- [General Evaluation Questions]

Check if answered:

- What do the angles/velocities mean?
- Can you valuably interpret the meaning of the data?
- Is this helpful in performing/learning/reflecting on/improving a somersault?

P1	
P2	
P3	
P4	
P5	
P6	

3. Interaction: Clickable body parts

"…moet je kijken naar de essentie van de aangedragen oplossing en je afvragen: Hoe zou OpenPose het idee van aantekeningen / vergelijkingen naar een volgend plan kunnen tillen. Het gaat per slot van rekening over de potentie van openpose"

- Make a mark = get stats (ie. circle knee = get angle)
- Reflect on values given (is this good, too big/small, what would be ideal?)
- Pause at tightest tuck > guess this moment, when is this moment supposed to be?

→ ik denk dat dit te ver gezocht is/buiten de scope

Focus on: how does the user interact with prototype

Clickable body parts:
- Phase frames
- Show angle/velocity of body part
- Multiple body parts → cummulative toggle
- Stats that encompass whole body?
    - Exclude
    - Give separately ('general' button or msthng)

Explain to user:
- This prototype works similar to previous one in terms of controls
- Tasks will be shown in screen
- [General Evaluation Questions]

    Check if answered:
- What method of selecting body angles would you prefer?

P1	
P2	
P3	
P4	
P5	
P6	

4. Guidance: Line Tracking

- Select a body part (from list is easiest i think)
- Video of somersault plays with body part highlighted and leaving behind a trail
- Optional: compare to known somersault of same person

Explain to user:

- This video prototype will show you a number of somersaults
- The left angle is tracked, this is shown by a line following it
- [General Evaluation Questions]

Check if answered:

- Which fade do they prefer? (current or best)

P1	
P2	
P3	
P4	
P5	
P6	

5. Data Visualisation: Visual corresponding to data

Explain to user:

- You will see a video of a somersault, imagine this was yours
- Under the video a metric is shown as a moving graph
- [General Evaluation Questions]

Check if answered:

- Are the visuals clear/easy to interpret?
- What can you tell about your somersault based on the graph?

P1	
P2	
P3	
P4	
P5	
P6	

6. Compare Yourself: Ghost Image

- Should not be random (but based on ranked previous saltos, similar etc..)
- Overlay your current somersault with previous one (or multiple)
- Be able to select
    - rating of somersault
    - Amount of somersaults

Explain to user:

- You will see a video of a somersault, imagine this was yours
- You will then see three comparisons
- [General Evaluation Questions]

Check if answered:

- Are the visuals clear/easy to interpret?
- Playback speed?

P1	
P2	
P3	
P4	
P5	
P6	

A.1.5. Hi-Fi prototype evaluation protocol and transcript

Hi-Fi Evaluation Script

Goal: learn about the value of OpenPose data for feedback on a somersault, taking an approach such that during evaluation the Hi-Fi's underlying relevant concepts can be distinguished and reflected on.

Bring:

- PC with prototypes and charger
- Microphone
- Paper & pens (fornotes/observations)

Intro:

- Explain goal and give indication of duration (aiming for 15min)
- Sign consent form
- Ask permission to record (voice?) > start recording here
- Do a brief introduction on their gymnastics experience:
    - Age
    - Years of experience
    - Roles within gymnastics (athlete/coach/judge etc.)

Body:

Briefly explain each prototype before continuing
There are no wrong answers and no need to be polite, honesty is best. I invite you to think out loud, speaking your mind during the evaluation.

Please pretend that you are trying to learn or improve the somersault during this interview
You can take any perspective you want in giving answers (athlete, coach, judge etc.)

Start taking observational notes here

General questions:

- Tell me about your first impression of this prototype
- What are things you like about the feedback
- What are things you dislike about the feedback
- How can the feedback be improved?
- Based on this feedback, can you explain what can be done better next time?
    - Can you also do this without the feedback? (i.e., just the video)
    - Does this feedback add value to the footage? (what/why not?)
    - Does this feedback improve the quality of your personal reflection?
- For which phase of the somersault did you consider the feedback to be (most) useful?
- What are reasons you would not like this type of feedback?

Prototype 1: Compare 0 with 8 (knee angle)
- What do you think of gridlines?
- Can you think aloud how you are interpreting the current angle value?
- Is a discrete moment in time (as is the case) sufficient for you to reflect on?

Prototype 2: Compare 1 with 7
- None

Prototype 3: Compare 3 with 0 (tracking nose initially)
- None

Prototype 4: Show 11
- What is a center of mass? (explain..)
- What can you say about the distance between dots?
- How do you feel about a comparison here?

- Concluding with: Is there anything else you want to mention about the feedback?

Thanks, stop recording

General Evaluation Questions

- Tell me about your first impression of this prototype
- What are things you like about the feedback
- What are things you dislike about the feedback
- How can the feedback be improved?

- Based on this feedback, can you explain what can be done better next time?
  - Can you also do this without the feedback? (i.e., just the video)
  - Does this feedback add value to the footage? (what/why not?)
  - Does this feedback improve the quality of your personal reflection?

- For which phase of the somersault did you consider the feedback to be (most) useful?
- What are reasons you would not like this type of feedback?

Time management guideline

Introduction, setting up recording & signing consent	2 min
Prototype evaluation	4x5min each = 20min
Concluding (final remarks, verifying notes etc.)	3 min
Total:	**25 minutes**

Hi-Fi Prototype Evaluation Transcript

Legenda of transcript and coding:

Punctuation

(additional context)	To clarify certain phrases and references
[...]	Irrelevant spoken words
R: What time is it?	As spoken by researcher, often questions

Coding:

Positive remark about, or sign of useful feedback	For example, being able to give precise points of improvement based on the prototype is found to be positive
Negative remark about, or sign of irrelevant feedback	For example, unclarity or doubts about the usefulness of feedback
Statement implying a more general concept	For example: it is important to consider what angle is relevant at what phase
Relevant question/condition/requirement/ or constraint of the prototype	For example: the grid would *only* help me if I was training on jump height specifically

Overlap in coding

This is an example sentence with overlap in coding	Sentences relating to multiple coding criterias are indicated like this.

P1:

General:
Female, 48 years, athlete and adjudicator

First prototype:
R: Please tell me about your first impressions of what you're looking at.
[...]
First thing I'm wondering is to what extent are we using this (knee) angle, it's different for landing but here getting the core angle (knee, hip, shoulder) is more relevant. That is because that (the extent of straight posture on takeoff) indicates how long you'll be gaining height before initiating the turn.

R: What are your thoughts on provided feedback?
I can think of various elements of gymnastics that could really benefit from these angles, especially for adjudication.

R: And for the somersault specifically?
What I would like to see here (frame is at landing) is not just your body angles, but angles relative to the ground here. [...] You told that in theory any angle could be shown and in that case you should consider every phase of the somersault individually, and think about which angles are relevant at that point in time. [...]

R: Thinking about these phases, what phases would benefit most from this feedback?
For the somersault the takeoff is rather important, there the angle of your whole body should in fact be considered, it is desirable to stay straight as long as possible. Here I do not really care about this (knee) angle. [...] Additionally, the angle of jumping into the trampoline is also important, since this is quite different depending on what kind of jumping tool you're using.

R: Lastly, I've added a grid, can you tell me something about what you think of that?
Well I'm not bothered by it…If you're reflecting on your jump height than it is useful for spatial referencing.

Second prototype:
R: Tell me about whether you can derive feedback from this comparison

The height is exactly the same [...], it seems in one of the two you're falling backwards on the landing.
R: Imagine this as being your somersaults, what can you improve for the falling one?
I see that for the extending you are too slow on that somersault, which leads to your feet being too much afront your body.
R: Conceptually speaking, tell me more about what you think of this feedback
I can imagine this is quite useful for situations in which you have a similar perception of somersaults that yield different outcomes. Because you can very clearly see the tipping point, up until the highest aerial point these somersaults seem identical, you can then distinguish where you're going wrong, without needing a trainer to tell you. For situations with less experienced trainers, or when you're on your own due to whatever reason. This forces you as an athlete to think about the comparison and reflect on how they (the somersaults) felt. Additionally, noticing something does not necessarily imply that you have the capability to also change this.

R: Based on the current feedback, do you think you are able to change your performance?
I think so yes, giving the current example, I can see that one should be rotating faster.

Third prototype:
The grey one is better.
R: Tell me why
The somersualt is higher and closer (to the trampoline), I can see that the chance of you sticking the landing is higher on the grey one since the line is less inclined towards your landing.
R: Can you say something about the jump?
I think for the blue one you're jumping too high (into the trampoline), after I can see that for blue you're already rotating on takeoff.
R: What do you think of these lines?
It is very clear in an instant. If you know the movement, if you have a feel for what it should be like then I think such lines are very insightful.
R: Interesting, because I'm not showing you any footage of the somersault (the comparison)
No you're not, there's no need to show it. In fact, as a judge this is what I always do. Because it's a swift movement, I can observe the takeoff and after I see such a line inside of my head. Landing corrections I can observe well again, I have enough time for that. But I mainly envision this line. [...] This doesn't happen deliberately, but looking at a video and then in my head I draw

this line based on things like body type and other aspects of the athlete. In fact I am doing exactly this, I think it's funny to see this drawn out like this because it makes me realize I've always been doing this in my head already. [...]
R: What can be improved?
I think this is very clear. I would not know what to improve.

Fourth prototype:
R: Tell me what is going through your mind.
I'm not sure if this is something I'd like to know. For me, this doesn't add a lot of value, I would not know how I can use this for feedback.

R: How could this be improved? What do you think of the center of mass concept?
Why did you choose this, what exactly did you think it was gonna provide in terms of feedback?
[...] I don't think I can make sense of something that revolves around the center of mass. The center of mass is different between women and men as well. I understand the concept technically speaking, but not in relation to improving your somersault. [...]

R: Concluding, are there any other things you'd like to mention for now?
I like the lines (third prototype) a lot. Usually things are an observation at a certain timeframe and you have to remember all things together, but when overlaying it (the prototype) over the whole somersault with also a comparison, then it is way easier to reflect with someone on what is going on. This is fine inside of the grid because this also helps you to more easily determine whether someone is jumping his own length (height wise) and other things. This is also something that is important for a somersault. [...]

P2:

General:
Female, 20 years, athlete and adjudicator

First prototype:
R: Tell me about what you're seeing.
I'm wondering to what extent we are using this angle (for reflection). For takeoff I'd prefer the core angle, and for landing I'd prefer this knee angle to be shown.

R: More conceptually speaking, what do you think of this feedback?
For [inauble gymnastics apparatus], this would be really nice. For the somersault I'd consider this less relevant. For the landing phase I think this angle can help, since it shows how you're leaning backwards too far. [...]

R: Consider the different phases of a somersault, are there phases in which you would value this feedback more or less?
The phase of takeoff is important, there you want to be stretched out. Only after rotation other angles become important such as the inclination towards landing (this is a body relative to ground angle).

R: Concluding, there is also a grid, what does this do for you?
Didn't really pay attention to this to be honest. Mainly for comparing with others this grid can be quite useful, also the vertical lines give a nice reference to where exactly you are landing as compared to just eyeballing that.

Second prototype:
R: Please tell me your thoughts.
It is clear to me to see for example the position of the arms. They look pretty similar otherwise. [...]

R: Imagine these somersaults being yours, how can you reflect in order to prevent yourself from falling based on what you're seeing.

I would get ‘an error’ here, because the first part of somersaults are similar, and you even initiate the rotation identically, until your landing. Thinking about this, I can in fact tell the difference. Comparing, the first part is the same and then after I can see the rotation of the second somersault is insufficient. It actually is quite precise to tell the difference. Comparing like this is nice since the differences are very well visible.

Third prototype:
R: The floor is yours.
The grey one is better, I can tell that right away. This is visible on first glance.

R: Can you tell me something about the run up and jump?
You’re jumping in the middle of the trampoline I feel, I can see you’re rotating already during takeoff here (referring to the blue line).

R: What do you think of these lines?
It’s clear right away. I can tell the grey one is better because it is shows me more height and a landing closer to the trampoline. [...] I don’t think I need to even see the other somersault at all (in order to reflect).

R: What can be improved?
This is very very clear honestly. Take it as a compliment. [...]

Fourth prototype:
R: What do you think?
Upon seeing this I think I’m getting a ‘complete error’, these dots… it is a lot to see…
In my opinion, I don’t think this is something I would consider relevant. I’d rather get to know my rotational axis. I would not know how to get valuable input from this for myself. [...]

R: What could be improved? When adding a comparison for example?
I’ve never really been concerned with weight distribution during somersaults.

R: Is there anything else you want to add?
The two lines (prototype three) are great in my opinion. That instantly gives me a lot of insight

into different jumps, different elements etc… [...]

P3

First prototype:
R: Please tell me what you think.
I think that is useful to see here (during jump/landing), not so much for the starting phase

R: What do you consider the phase you're talking about?
The run up mainly, the jump might be useful, but it's mainly about the aerial phase. When you're airborne, then it's about the knee angle. [...] I'm missing the shoulders (angle of). I would like to see the angle between knee, hip and shoulders. When you're jumping you're tilted rather forward, and then it would be more useful to see that (core) angle, also for children that are practicing somersaults. [...]

R: Let's look at the takeoff then, imagine the angle being displayed there for your core, let it be 175 degrees for example, does this add any value to you?
I think that would give clarity compared to others. If you would have two somersaults and compared the angles of them side by side, then it can very well show differences in execution. It'd then be nice to have some sorting of red areas (indicators of good/bad) for these angles as well.

R: You're saying this is more useful when compared?
Yes I think so. [...]

Second prototype:
R: What are your first impressions?
So it's two different somersaults right?

R: Yes
Here I can see that with extending you're too late on this one. It is funny to see it like this.

R: Can you tell me the differences between these somersaults?
I find that challenging. Because they partly overlap that makes it difficult for me to properly the one in the back (the ghost). [...] Only in the end when there really is a significant difference between the two is when it becomes quite clear to me.

R: How could this idea be improved according to you?
I would like to view them side by side, since you also have a grid already, this can be used in order to spatially compare them well.

R: So you would like them next to each other, where you compare the locations based on the same grid?
Yes. And what I'd also like is that when you perform a good somersault, when jumping and comparing again that you are then able to view the deviations compared to the good one. That there will be some sort of indicator to explicitly show me when the somersault is 'on track' and when things are going wrong. Perhaps with a kind of notification.

Third prototype:
R: (still explaining)
This is nice. This is very nice.

R; Why?
You said it tracked your nose here? And it's also possible to track your shoulders and knees?

R: Yes. (sets bodypart to shoulder)
Immediately you can see a difference between grey and blue. [...] White is higher than the blue one. I'd that makes it the better one here. It is prettier, it is higher, it (the landing) is closer to the trampoline, the shoulders are higher up which indicates to me that the rotation is better. Based on this I can see quite a lot about your somersault, also your jump into the trampoline: here (points to blue) you're jumping earlier, that makes you land deeper into the trampoline benefitting the height of your jump. And these things allow me distinguish where also things are going wrong for the other somersault.

Fourth prototype:

R: So tell me about it.

I think this is less useful, reason being… Previous one was more useful [...] Here you don't have an overview of the somersault as a whole without scrolling.

R: Let's say the whole somersault being able to be displayed with the center of mass, how would that be?

Personally, I'm a fan of seeing the center of mass during the trampoline contact. That's important for your takeoff. [...] Only then I think it's important, apart from that I think it's not so useful for reflection on a somersault.

R: Would a comparative feature make this more relevant?

I have difficulty imagining that. [...] (looks confused)

R: Concluding, is there anything you'd like to add to your statements about this?

Center of mass is only useful during trampoline contact, the third one (prototype) is very useful as it is an easy comparison with a lot of available information. The second one didn't strike me as much as useful, because it is harder to see the differences compared to the lines (third prototype). At the moment of deviation is when you really have to focus on spotting when that happens.

P4:

First prototype:
R: Tell me what you think about this somersault and the feedback.
I'm not quite sure whether showing these angles has added value over just showing the video itself. You can see it visually in front of you already, where it is harder to tell because of the pants waving, but in that case you can also wear tighter pants.

R: Does the angle value provide you with some useful information?
Wasn't paying attention to that to be honest. [...] Saying I can really 'do' something with this…, I'm not sure about that.

R: How can this be improved?
I can think of plenty of apparatus for which this would be very useful, for example more power based exercises. Additionally it'd be nice for a 'hoeksalto' (angled somersault) or stretched somersaults. For a tucked somersault I'd say not so useful.

Second prototype:
R: Share your thoughts please.
They are different. I should wear glasses it feels like I'm seeing double. [...] One somersaults appears to be opening faster than the other. Not sure if I'm seeing that correctly?

R: Looking at this, is this useful feedback?
I'd say that it should be more clear which somersault is which, for example with coloring of the ghost. Now it can be confusing to distinguish the two properly. It looks like motion blur at times. You could also put them next to each other. This makes it harder to compare spatially but more clear to view the somersault themselves. [...]

R: What would you then prefer in that case?
Depends on the context, next to each other comparing will be hard since you're missing a clear reference to compare them. [...] Overlapping can be hard to distinguish, but would be more useful when they are more distinct.

Third prototype:

R: What do you think?

Well here I can nicely see the differences, since both somersaults are clearly indicated by a color. Potential downside here can be that you're zooming in on a single bodypart so that makes its value depend on your goal with feedback. [...] What I'm thinking here is considering that it's good practice to jump 'over' your own body length, you can add a virtual reference and then compare with these lines whether you meet this requirement. Additionally, the lines can then show how you're doing going into the jump and then reflect on what works and what would not work well in achieving the height, for example. I like that it's simple. You can add a lot of measurements but in my experience it is pretty challenging already to take specific feedback into account for improvement of a performance. [...] I think one bodypart to focus on makes it easier to understand what is going on.

Fourth prototype:

R: Please tell me what you're thinking.

I think the video itself provides me with more information than the center of mass. [...] I think that for a somersault your center of mass is not a very important parameter.

R: Would comparing improve this?

I think comparing would be more useful, but in my eyes this does not account for the fact that I still don't think the center of mass is very useful. I think the rings (apparatus) would benefit greatly from being able to see your center of mass but apart from that I don't see a lot of benefit. The rings require a lot of strength and that makes your center of mass more important [...].

R: Concluding, do you have anything to add?

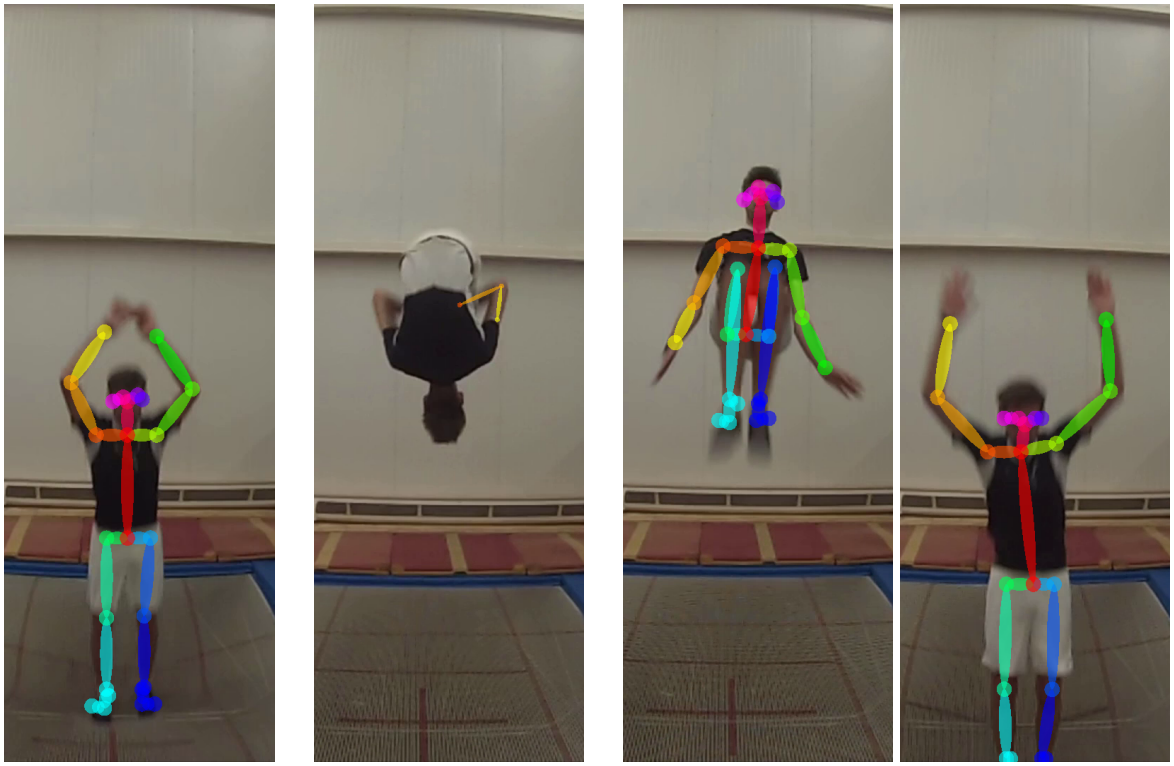
Well as I said [repetition of previously noted comments], apart from that, no.

B

OpenPose Performance

B.1. OpenPose's Performance Graphs for Front View

This section contains performance graphs of OpenPose for the 0 degree or front view camera angle somersaults. The graphs are assessed to be similar to those taken at a 45 degree or diagonal view which are shown in section 4.1.2.



(a) frame 0 **(b)** frame 21 **(c)** frame 42 **(d)** frame 62

Figure B.1: OpenPose render of a somersault (No. 88) from the dataset (front view)

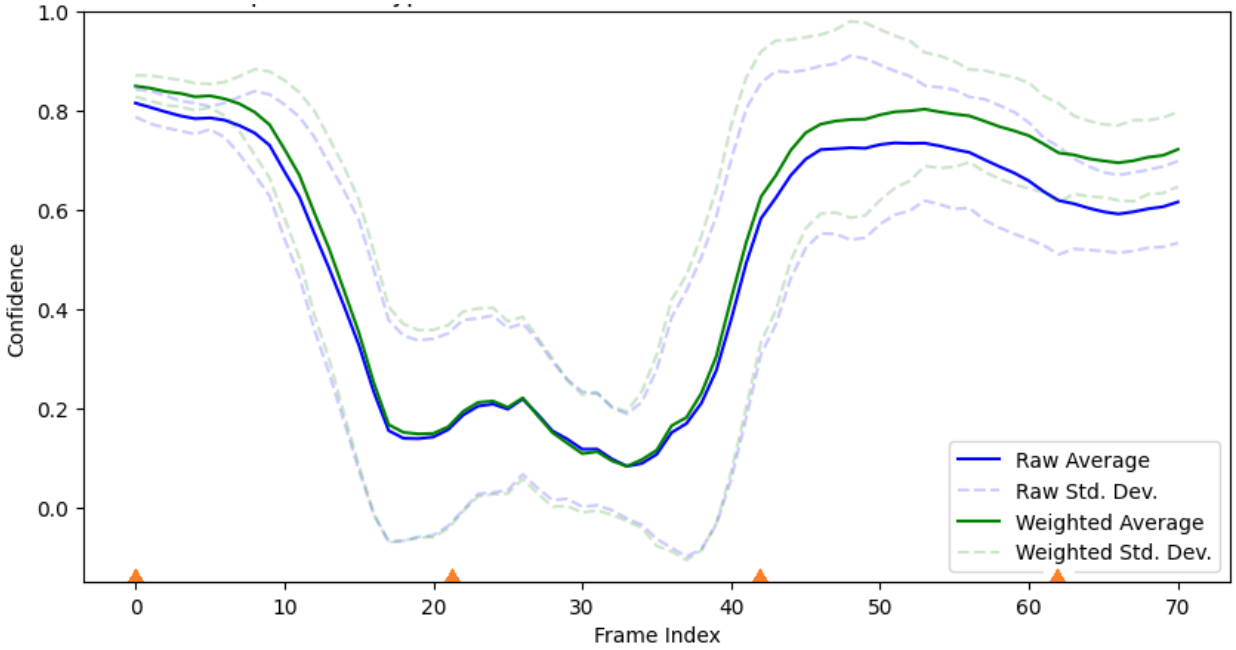


Figure B.2: OpenPose overall keypoint confidence of somersaults (n = 115) - front view. The orange markers represent the location of rendered frames.

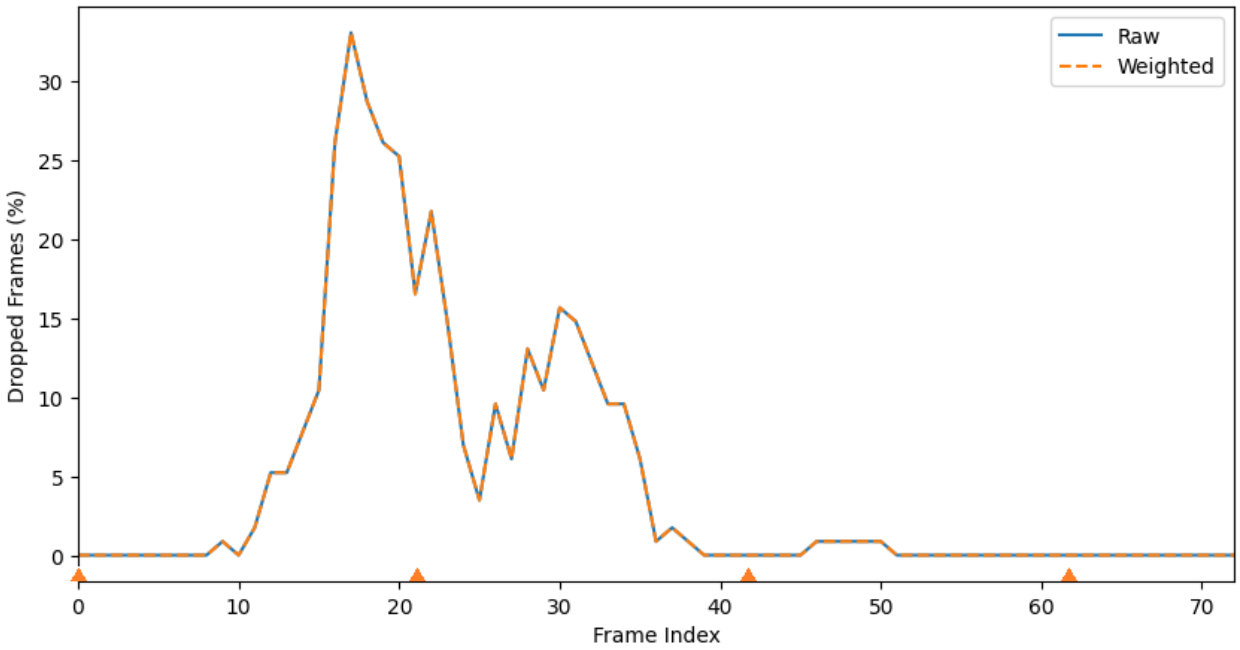


Figure B.3: Cumulative amount of frames dropped (i.e. no keypoints detected) of somersaults (n = 115) - front view. The orange markers represent the location of rendered frames.

C

Ideation Personas

Personas that were used as inspiration for design ideas. Note: these are fictional characters and their images AI generated.

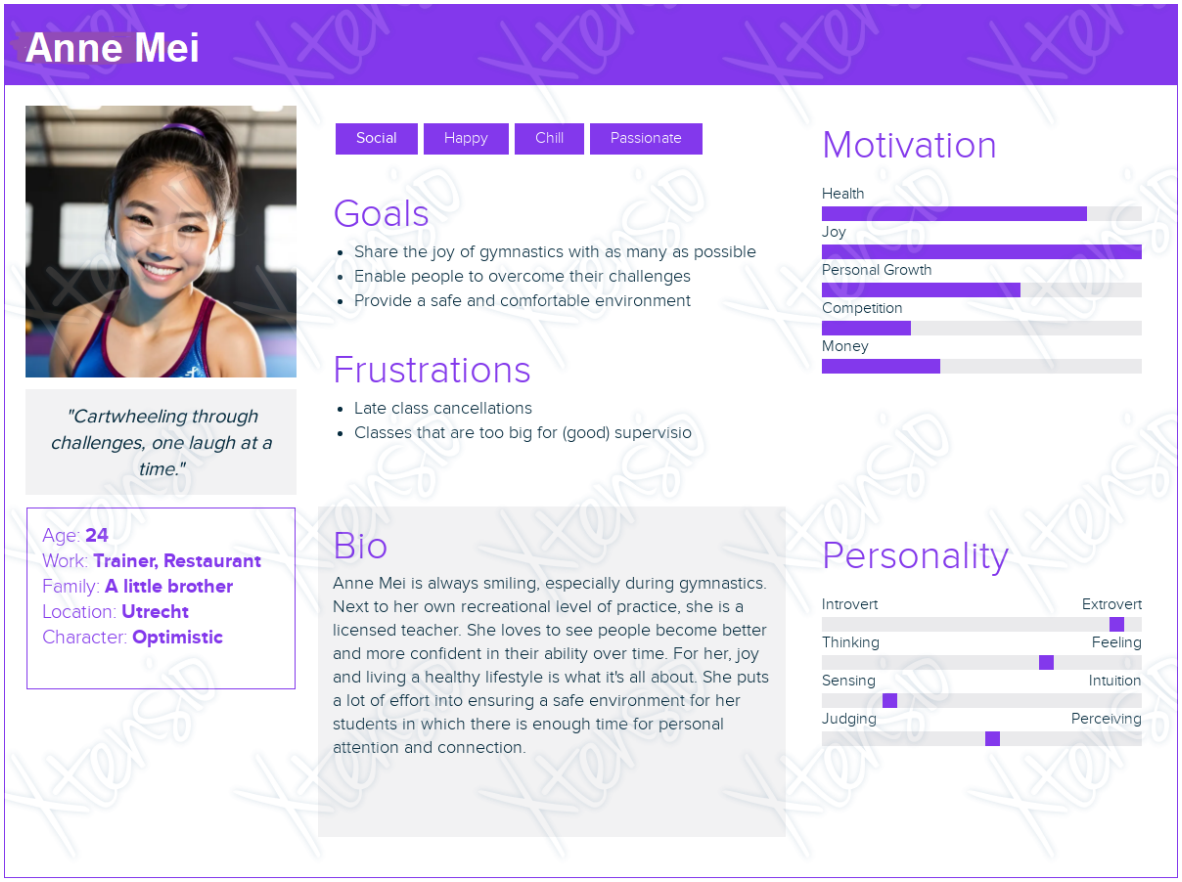


Figure C.1: Persona: Anne Mei

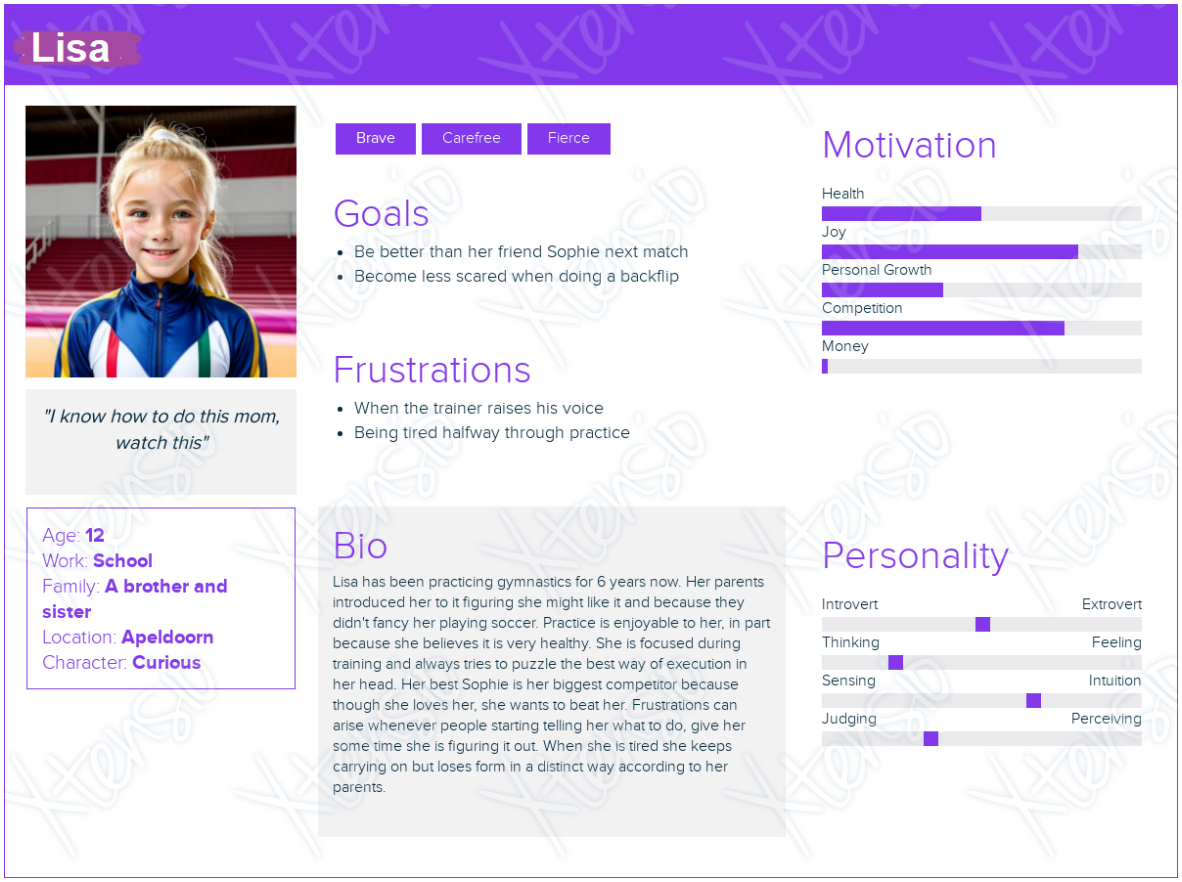


Figure C.2: Persona: Lisa

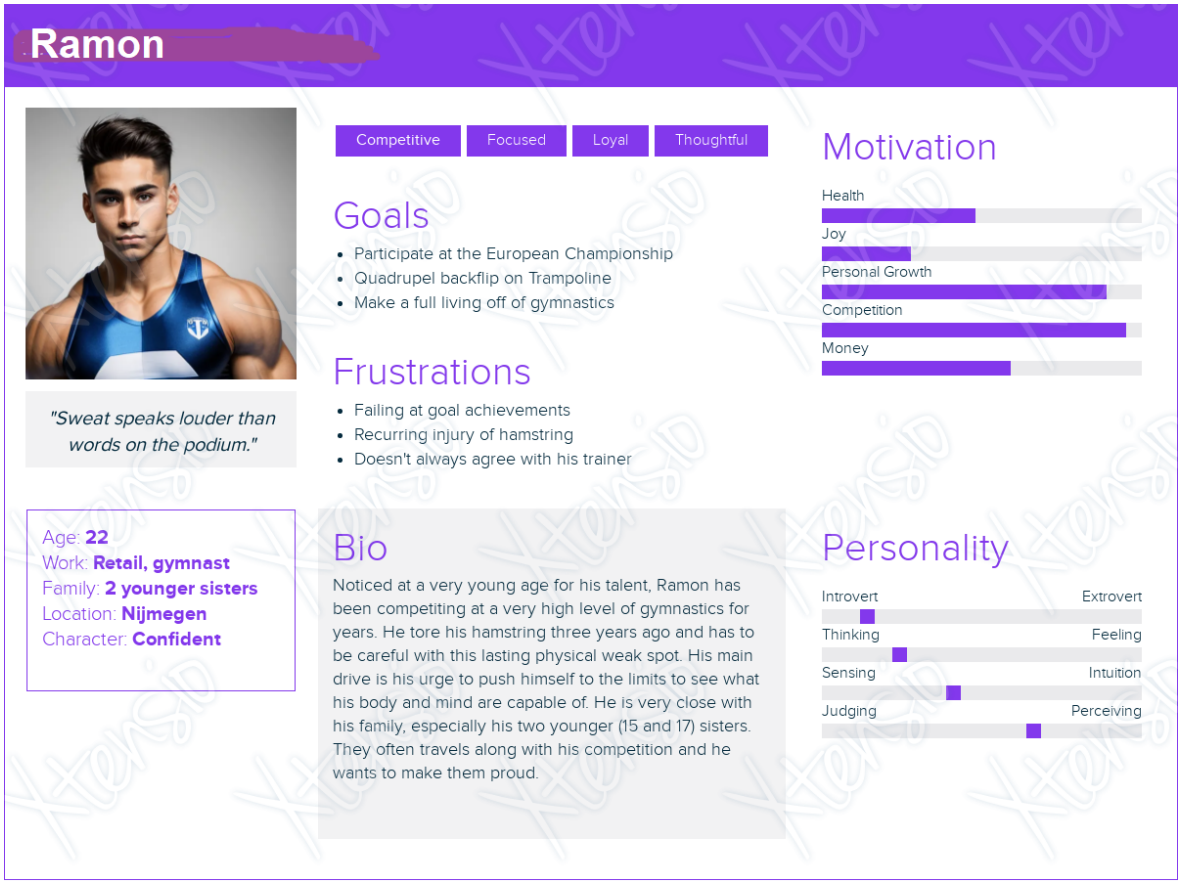


Figure C.3: Persona: Ramon

D

Source Code

The following code is written in Python, version 3.9.7 using the Pycharm IDE (build 231.9011.38, May 16, 2023). Context on the functionality and application is commented for each file.

main_v2.py :

```
# This python code uses OpenPose data in combination with a custom video player in order to
give somersault feedback.
# Written by Niels van Dalen, for a Master Thesis in Interaction Technology
# last edited: 24-04-2024

# The code is able to execute the following:
# - Find and scan all directories (under general directory root_path)
# - Load in all JSON and videos (mp4, avi) files  inside these directories
(directoryscanner.py)
# - Extract the keypoints from the JSONs (jsonprocessor.py)
# - Correct keypoint values for incomplete points/files (keypointcalculations.py)
# - Use and process keypoint values for 4 prototypes that provide feedback
#   atop of a controllable video window showing the somersault. (hifiprototypes.py)

# - Optional: plot keypoint confidence levels and amount of frame drops (visualisation.py)
#   These plots are indicative of the performance (accuracy) of OpenPose

# PLEASE NOTE:
# - Obey the naming conventions of JSON and Video files as commented in directoryscanner
# - Don't stare too long at the code for crippling depression will crawl upon you

# Settings for the program
interface = True # Figure 0, this is the interface
single_plot = False  # Figure 1, displays all found json files and their confidences per frame
(messy on large sets)
combined_plot = False  # Figure 2, takes the average of all found json files, and plots
together
dropped_frames = False  # Figure 3

#removed:
#relevant_keypoints_only = True # Filter the data on relevant keypoints only (12 keypoints
instead of all 25), see doc

import math
import matplotlib.pyplot as plt
from directoryscanner import DirectoryScanner
from jsonprocessor import JSONProcessor
from keypointcalculations import KeypointCalculations
from visualisation import Visualisation
from guidance import*
from directfeedback import*
from compareyourself import*
from hifiprototypes import*



def main():
    root_path = 'C:\\Users\\niels\\Documents\\Utwente\\Jaar ' \
                '7\\Graduation\\Technical\\OP\\openpose-1.6.0-binaries-win64-gpu-flir-
3d_recommended\\openpose' \
                '\\data_video_combined'
    print('\n')  # make some space

    directory_scanner = DirectoryScanner(root_path)
    directory_scanner.scan_directories() #scan_directories calls
extract_number_from_filename() for sorting
    subdir_list = directory_scanner.get_subdirectories()
    directory_scanner.import_videos()
    # Sort the subdir_list based on the numeric values in the subdirectory names. Wups it's in
main boohooo :((
    #subdir_list.sort(key=lambda subdir:
directory_scanner.extract_number_from_filename(subdir))
    # it works though whatcha gonna do? Update 16-01-2024: rip it did cause problems down the
line, changed and  moved to
    # directoryscanner.py

    json_processor = JSONProcessor(subdir_list)
```

```
    json_processor.process_json_files()

    # Filter out subdirectories with 0 JSON files
    valid_subdir_list = [subdir for subdir, num_frames in zip(subdir_list,
json_processor.number_of_frames) if
                         num_frames > 0]

    # For making confidence graphs enable these
    #visualisation = Visualisation(average_confidence, avg_frames,
keypoint_calculator.empty_frames_sum)
    # if single_plot:
    #     visualisation.single_plot_graph_per_subdirectory()
    # if combined_plot:
    #     visualisation.plot_combined_graph()
    # if dropped_frames:
    #     visualisation.dropped_frames_plot()
    # plt.show()  # prevent the plots from closing right away (cuz code = done --> plot = gone

    ################## Hi Fi Prototypes ##################
    hifiprototypes = OpenPoseData(subdir_list, directory_scanner.video_files)
    hifiprototypes.import_keypoint_values()
    # Get the keypoints for filtering
    keypoints = hifiprototypes.keypoint_values
    hifiprototypes.play_video(resize=True)

if __name__ == "__main__":
    main()
```

directoryscanner.py :

```python
# here we scan subdirectories under root_path (specified in main), sort contents and store in a
list for JSONProcessor
# extraction

import re
import os
import glob
import cv2
from scandir import walk

class DirectoryScanner:
    def __init__(self, root_path):
        self.root_path = root_path
        self.subdir_list = []
        self.leafdir_list = []
        self.video_files = []

    def extract_number_from_filename(self, filename):  # used for sorting by the folder number
in the name
        # Extract the numeric part from the filename
        match = re.search(r'(\d+)salto@deg(\d+)', filename)
        #WARNING: this forces name requirement of: (number)salto@deg(angle) and a single
parent directory containing
        # the somersaults (i.e. both a 'deg0' and 'deg45' directory will fuck up sorting
        if match:
            return int(match.group(1))
        return 0  # Return 0 if no numeric part is found

    def scan_directories(self):  # Scan all the subdirectories under root_path (specify in
main) and store in list
        for root, dirs, _ in walk(self.root_path):
            if root == self.root_path:
                continue
            self.subdir_list.append(root)

            # Sort the subdir_list based on the numeric values in the subdirectory names
            self.subdir_list.sort(key=lambda subdir:
self.extract_number_from_filename(subdir))
        #print((self.subdir_list))# prints the full dir list (ie.
C/users/niels...../jsonfilefolder)

    def get_subdirectories(self):  # return the list of subdirectories
        #print(self.subdir_list)
        return self.subdir_list

    def import_videos(self):
        for subdir in self.subdir_list:
            # Construct the full path for each directory
            video_path = os.path.abspath(subdir) #convert to absolute path
            # Use glob to get a list of video files with .avi or .mp4 extension
            self.video_files.extend(glob.glob(os.path.join(video_path, '*.avi')))
            self.video_files.extend(glob.glob(os.path.join(video_path, '*.mp4')))
        #print('[printed in directoryscanner.py> def import_videos] found video files:', '\n',
self.video_files,
        #       '\n')


        return self.video_files
```

jsonprocessor.py :

```
import os
import json
import numpy as np
from directoryscanner import DirectoryScanner



class JSONProcessor:  # In 'subdir_list' check for JSON files and extract (only) the keypoints in there
    def __init__(self, subdir_list, keypoint_attributes=3): # each keypoint has attribute x,y,c
        self.subdir_list = subdir_list
        self.json_files = []  # list for storing the json file names (print if unsure your files are being seen)
        self.number_of_frames = []  # list that stores the amount of json per folder, for finding an average frames
        self.keypoint_attributes = keypoint_attributes  # amount of keypoint attributes, here 3: an x, y and c value (c = confidence of estimation)
        self.frame_data = [[] for _ in self.subdir_list]  # LOL of all raw keypoint data
        self.entries_without_people_key = [[] for _ in self.subdir_list]
        self.amount_of_keypoints = 25  # Hardcode please reach out to 06 8182 1571 if u want to cry about it
        self.keypoint_values = []
        self.keypoint_values_T = []

    def process_json_files(self):
        directory_scanner = DirectoryScanner(self.subdir_list)

        # Create lists to store x, y, and c values, used for creation of the 'keypoint_values' np array
        x_values = []
        y_values = []
        c_values = []

        json_files_list = []

        for i, subdir in enumerate(self.subdir_list):  # Loop through all subdirectories under 'root_path' (see main)
            current_jsons = [pos_json for pos_json in os.listdir(subdir) if pos_json.endswith('.json')]
            self.json_files.append(current_jsons)  # every file ending in .json gets added to current_jsons
            subdir_name = os.path.basename(subdir)  # get current subdirectory name to make senseful prints
            # print(f"in directory '{subdir_name}':")  # print the subdirectories
            self.number_of_frames.append(len(self.json_files[i]))

            # print(len(self.json_files[i]), 'JSON files') #amount of json files in a folder
            # print(self.json_files[i], '\n')
            os.chdir(subdir)  # update cwd to current (next) subdir

            for j, json_file in enumerate(current_jsons):  # Loop through amount of json files in subdirectory i
                json_files_list.append(current_jsons)
                with open(json_file) as open_json:
                    json_str = json.load(open_json)
                    # print(json_str)
                    # print(current_jsons[j]) #DEBUG: Current jsons contains all json files (also empty ones)
                    # Create empty zeros lists corresponding to keypoint size (= 25)
                    xVals = [0] * self.amount_of_keypoints
                    yVals = [0] * self.amount_of_keypoints
                    cVals = [0] * self.amount_of_keypoints

                if 'people' in json_str and len(json_str['people']) > 0:  # This is always true also for empty files
```

```
                    if 'pose_keypoints_2d' in json_str['people'][0] and any(json_str['people']
[0]['pose_keypoints_2d']):
                        keypoints = json_str['people'][0][
                            'pose_keypoints_2d']  # if no 'pose_keypoints_2d' then error List
OOR
                        self.amount_of_keypoints = int(
                            len(keypoints) / self.keypoint_attributes)  # amount of keypoints
is
                        # the amount of data points / attributes, in this case should always
be 25 = 75 / 3 attributes (x,
                        # y,c). Also convert to int so for loop isn't gonna be a lil bitchboy

                        # order of keypoints in the json is x-y-z: x0, y1, c2, .... x72, y73,
c74
                        x = 0
                        y = 1
                        c = 2

                        for k in range(min(self.amount_of_keypoints, len(keypoints) //
self.keypoint_attributes)):
                            xVals[k] = keypoints[x]
                            yVals[k] = keypoints[y]
                            cVals[k] = keypoints[c]
                            x += self.keypoint_attributes
                            y += self.keypoint_attributes
                            c += self.keypoint_attributes

                        # Append x, y, and c values to the respective lists
                        x_values.extend(xVals)
                        y_values.extend(yVals)
                        c_values.extend(cVals)


                    else:  # If 'pose_keypoints_2d' key is missing or empty
                        # Extend x_values, y_values, and c_values with arrays of zeros of the
same length
                        xVals = [0] * self.amount_of_keypoints
                        yVals = [0] * self.amount_of_keypoints
                        cVals = [0] * self.amount_of_keypoints

                        # Append x, y, and c values to the respective lists
                        x_values.extend(xVals)
                        y_values.extend(yVals)
                        c_values.extend(cVals)

                        # Append the JSON file name to the list
                        self.json_files[i].append(json_file)

                        # Append zeros to frame_data for this empty JSON file (also: I love
programming)
                        json_file_data = {
                            'subdirectory': subdir_name,
                            'json_file': json_file,
                            'x_vals': xVals,
                            'y_vals': yVals,
                            'c_vals': cVals
                        }
                        self.frame_data[i].append(json_file_data)

                json_file_data = {
                    'subdirectory': subdir_name,
                    'json_file': json_file,
                    'x_vals': xVals,
                    'y_vals': yVals,
                    'c_vals': cVals
                }
```

```
                self.frame_data[i].append(json_file_data)  # store the data into the list of
dictionaries frame_data[]
                                                           # this includes the file name and
subdirectory

                #for data in self.frame_data[i]: #Print all the json files found
                #   print(data['json_file'])

        #print(self.frame_data[2]) #argument is subdirectory (don't be surprised if some are
empty you monkey cuz no
        # json)
                # From frame_data (see above), extract the hard x,y,c values to use for
further processing
                # Extracting relevant data from each dictionary

        # Convert lists to np arrays, NOTE:
        #keypoint_values is dimensioned as: [[x1...xn][y1..yn][c1...cn]
        #keypoint_values_T has shape: [[x1,y1,c1][][]...[xn,yn,cn]]
        #making both just in case

        # here we loop through frame_data to extact the values, this includes zeros that are
added in empty json files
        keypoint_data = [[entry['x_vals'], entry['y_vals'], entry['c_vals']] for sublist in
self.frame_data for entry in
                         sublist]
        self.keypoint_values = np.array(keypoint_data)
        self.keypoint_values_T = self.keypoint_values.T
        #print(self.keypoint_values)
        return self.keypoint_values, self.keypoint_values_T, self.amount_of_keypoints
```

visualisation.py :

```
import numpy as np
import matplotlib.pyplot as plt

class Visualisation:
    def __init__(self, data, avg_frames, empty_frames_sum):
        self.data = data
        self.avg_frames = int(np.ceil(avg_frames))
        self.empty_frames_sum = empty_frames_sum

    def single_plot_graph_per_subdirectory(self):
        plt.figure(1, figsize=(10, 5))
        non_empty_subdirs = [subdir_averages for subdir_averages in self.data if
any(subdir_averages)]

        if non_empty_subdirs:
            for subdir_averages in non_empty_subdirs:
                plt.plot(subdir_averages[:self.avg_frames])  # Use only up to avg_frames
            plt.xlabel('Frame')
            plt.ylabel('Average Confidence')
            plt.title('Average Confidence per Frame for Invididual Somersaults')
            plt.legend([f'Somersault {i}' for i in range(0, len(non_empty_subdirs) + 1)])
#title somersaults
        #plt.show(block = False)

    def plot_combined_graph(self): # Print an average of all confidences, together with a std
(note: doesn't show with
        # only 1 input folder since std = 0 then)
        max_length = min(self.avg_frames, max(len(subdir_averages) for subdir_averages in
self.data))
        all_confidences = np.array(
            [subdir_averages[:max_length] + [0] * (max_length -
len(subdir_averages[:max_length]))
             for subdir_averages in self.data]
        )
        mean_confidences = np.mean(all_confidences, axis=0)
        std_dev_confidences = np.std(all_confidences, axis=0)

        # Plot properties
        plt.figure(2, figsize=(10, 5))
        plt.ylim(-.15, 1)
        plt.plot(mean_confidences, label='Average')
        plt.fill_between(range(len(mean_confidences)),
                         mean_confidences - std_dev_confidences,
                         mean_confidences + std_dev_confidences,
                         color='gray', alpha=0.3, label='Standard Deviation')
        # Title & Labels
        plt.xlabel('Frame Index')
        plt.ylabel('Keypoint Confidence')
        plt.legend()
        plt.title("OpenPose's Overall Keypoint Confidence of a Somersault - Diagonal view (45
degrees)")
        #plt.show(block = False)

    def dropped_frames_plot(self):
        plt.figure(3, figsize=(10, 5))
        plt.plot(self.empty_frames_sum)
        # Title & Labels
        plt.xlabel('Frame Index')
        plt.ylabel('Amount of frames dropped (Normalized)')
        plt.legend()
        plt.title("Amount of frames dropped - Diagonal view (45 degrees)")
        #plt.show(block = False)
```

keypointcalculations.py :

```
import json

from visualisation import Visualisation


class KeypointCalculations:
    def __init__(self, json_processor, avg_frames):
        self.json_processor = json_processor
        self.total_data_points = sum([len(elem) for elem in json_processor.json_files]) \
                                 * json_processor.amount_of_keypoints *
json_processor.keypoint_attributes
        # print('total data points:', self.total_data_points)
        self.empty_frames_sum = self.find_empty_frames()
        self.average_confidence = self.get_average_confidence()  # Store the average
confidence
        self.visualisation = Visualisation(self.average_confidence, avg_frames,
self.empty_frames_sum)  # for passing
        # confidence to
        # Visualisation

    def get_average_confidence(self):  # Get the average confidence level (c value) per frame
        average_confidence = []  # List to store the average values of confidence per frame
        average_confidence_weighted = []  # filtered list based on relevant_keypoints

        relevant_keypoints_indices = [0,1,3,4,6,7,9,10,11,12,13,14] # List of relevant
keypoints, see doc for number
        # references

        for element_list in self.json_processor.frame_data:  # nr of loops equal to amount of
folders found
            confidence_avg_per_salto = []
            confidence_avg_per_salto_weighted = []

            for element in element_list:
                x_vals = element['x_vals'] # unused i should do this in jsonprocessor
                y_vals = element['y_vals'] # unused i should do this in jsonprocessor
                c_vals = element['c_vals']  # Extract the c_vals for each element

                #print('x', x_vals)
                #print('y', y_vals)
                #print('c', c_vals)

                # Extract the specific keypoints that are deemed relevant for a somersault:

                #print(c_vals_weighted)

                average_c = sum(c_vals) / len(c_vals)  # Calculate the average value of
frames' c_vals
                confidence_avg_per_salto.append(average_c) # concerns 1 salto frame by frames,
i.e. len =+- 70-80frames
            average_confidence.append(confidence_avg_per_salto)   # Store all averages in a
LoL - all subdirs


        #print('\n', 'list of confidences:', '\n', average_confidence, '\n')
        #print('cvals weighted length:' + str(len(c_vals_weighted)))
        #print(str(len(confidence_avg_per_salto)))
        #print(confidence_avg_per_salto_weighted)
        # print('grand total confidence:', sum(average_confidence) / len(average_confidence))

        # FIND OVERALL CONFIDENCE VALUE
        # print('avgconflen', len(average_confidence))
        # self.visualisation.single_plot_graph_per_subdirectory()
        return average_confidence

        # Empty frames contains zeros for all x,y,c. Chosing c because we already have it
```

```
    def find_empty_frames(self):
        empty_frames_sum = []  # List to store the total of empty frames by index, sum of
empty_frames_sequence
        empty_frames_sum_normalized = []

        # Determine the maximum number of frames across all subdirectories
        max_frames = max(len(subdir) for subdir in self.json_processor.frame_data)

        for i in range(max_frames):
            total_empty_frames = sum(1 for subdir_data in self.json_processor.frame_data if i
< len(subdir_data) and
                                     all(c == 0 for c in subdir_data[i]['c_vals']))
            empty_frames_sum.append(total_empty_frames)
        #print('efs:', empty_frames_sum)
        #print('efs length:', len(empty_frames_sum))

        # Normalisation of empty_frames_sum (list of dropped frame counts per index)
        if empty_frames_sum:
            min_val = min(empty_frames_sum)
            max_val = max(empty_frames_sum)
            if min_val != max_val:
                empty_frames_sum_normalized = [(x - min_val) / (max_val - min_val) for x in
empty_frames_sum]
        else:
            print('no empty frames found')

        return empty_frames_sum_normalized
```

hifiprototypes.py :

```python
# hifiprototypes.py
from jsonprocessor import JSONProcessor
from directoryscanner import DirectoryScanner
from scipy.signal import butter, filtfilt
import matplotlib.pyplot as plt

import numpy as np
import math
import cv2

class OpenPoseData:
    def __init__(self, subdir_list, video_files, keypoint_attributes=3):
        ############################################################
        ############################################################
        ############################################################
        # PROTOTYPE SETTINGS
        # Somersaults are selected by number, prototypes by booleans (0/1 or True/False)
        # Choose somersault (number between 0-11)
        self.somersault_number = 3  # somersault to evaluate
        self.somersault_ghost = 0 # somersault to compare against
        # Select which prototype you want to see, PB stands for Prototype Boolean
        # Note: these titles will likely be updated in the report in order to be more representative of the nature
        self.PBdirectfeedback = 0 #(hifi eval: 0 vs 8)
        self.PBcompareyourself = 0 #(hifi eval: 1 vs 7 )
        self.PBguidance = 1 #(hifi eval: 3 vs 0 )
        self.PBcenterofmass = 0 # #(hifi eval: 11) enable both guidance and CoM for this to happen (cv2 crashes
        # in separate CoM function)

        # Optional grid
        self.show_grid = 1
        self.gridspacing = 200  # grid square size in px

        self.Nose, self.Neck, self.RShoulder, self.RElbow, self.RWrist, self.LShoulder, self.LElbow, self.LWrist,
self.MidHip, self.RHip, self.RKnee, self.RAnkle, self.LHip, self.LKnee, self.LAnkle, self.REye, \
            self.LEye, self.REar, self.LEar, self.LBigToe, self.LSmallToe, self.LHeel, self.RBigToe, self.RSmallToe,
self.RHeel, self.Background = range(
            26)
        self.bodyparts = [self.Nose, self.Neck, self.RShoulder, self.RElbow, self.RWrist, self.LShoulder,
self.LElbow,
                          self.LWrist, self.MidHip, self.RHip, self.RKnee, self.RAnkle, self.LHip, self.LKnee,
self.LAnkle, self.REye, \
            self.LEye, self.REar]  # variable used to loop through all bodyparts

        # For PBdirectfeedback with bodypart 1 and 3 as legs of 2
        self.selected_bodypart1 = self.RHip
        self.selected_bodypart2 = self.RKnee
        self.selected_bodypart3 = self.RAnkle
        # For single body part tracking (i.e., guidance)
        self.selected_bodypart4 = self.RShoulder
        self.fitted_curve_points_resolution = 100
        self.poly_curvefit_degree = 5  # degree of polynomial fit, higher tends to overfit; not neccesarily bad here
but
        # will be susceptible to outliers (aka bs datapoints)
        self.show_rawvals = 0
        self.show_filteredvals = 1
        self.show_trendline = 0  # this feature is shit only enable for recreational purposes
        self.compare_with_ghost = 1  # provide a filtered line of the ghost's keypoint tracking

        ############################################################
        ############################################################
        ############################################################
        self.gridX, self.gridY = self.show_grid, self.show_grid
        # Create an instance of JSONProcessor
        self.processor = JSONProcessor(subdir_list, keypoint_attributes)
        # Call the process_json_files method to populate keypoint_values and keypoint_values_T
        self.processor.process_json_files()
        self.video_files = video_files
        self.resize = False
        self.frame_number = 0  # Initialize frame number
        # Import the keypoint values
        self.amount_of_keypoints = self.processor.amount_of_keypoints
        self.keypoint_values0 = None
        self.keypoint_values = self.processor.keypoint_values
        self.keypoint_values_T = self.processor.keypoint_values_T
        # Parameters for overlay:
        self.shapes_example = False
        self.line_thickness = 2
        # This is in (B, G, R) format, some colors to make coloring more pleasant and clear
        self.color_orange = (135, 185, 255)
        self.color_green = (100, 200, 100)
        self.color_blue = (201, 176, 118)
        self.color_white = (235, 235, 235)
```

```
        self.color_grey = (175, 175, 175)
        self.color_black = (0, 0, 0)
        self.color_pink = (225, 279, 200)
        self.alpha = 0.3  # amount of transparency of ghost on 0-1, highest at 1 in which only ghost is shown
        self.fill_shape = -1  # use to color fill a shape
        self.radius = 5  # to set a radius
        self.size = 250  # size of square
        self.x, self.y, self.c = 0, 1, 2
        # Assign values to bodyparts (0,1,2...25 in OpenPose's default order)
        np.set_printoptions(suppress=True)  # supress scientific notations during printing
        self.rating = [3,1,2,2,2,2,3,2,3,3,2,2] #UPDATE TO REFLECT TRUE RATINGS!
        self.filtered_x = None
        self.filtered_y = None
        self.filtered_x_ghost = None
        self.filtered_y_ghost = None
        self.filtered_data = {}

        self.numfr = 0 #variable used to loop trhough all frame for filtering UNUSED

    def import_keypoint_values(self):
        # Interpretation/contextualisation of keypoint values
        # Split array per somersault, number of frames differs per recording!
        split_at = [134, 198, 170, 175, 179, 184, 141, 164, 163, 171, 158, 152]
        self.keypoint_values = np.round(self.keypoint_values)
        self.keypoint_values_T = np.round(self.keypoint_values_T)

        # self.keypoint_values = self.keypoint_values[0:170,:,:]
        # print(np.shape(self.keypoint_values))
        # print(self.keypoint_values)
        # print(self.keypoint_values[133,:,:]) #access last frame of somersault_number = 0
        current_index = 0
        # loop through split_at and split keypoint_values at its indices
        for i in range(0, len(split_at)):  # for some reason enumerate is an incompetent lil sht so deal with it
            self.keypoint_values
            # separate keypoint_value variables with ascending numbers
            keypoints_subarray = f"keypoint_values{i}"
            # create index and update with each loop
            startIndex = current_index
            current_index += split_at[i]
            # for each keypoint_valueN sublist we assign the values based on the section of values from the full
array
            # this creates lists: keypoint_valuesN in which N = 0,1,2...N corresponding with the somersault number
            globals()[keypoints_subarray] = self.keypoint_values[startIndex:startIndex + split_at[i], :, :]
        return

    def on_trackbar_change(self, frame_number):
        # Callback function to jump to a specific frame based on trackbar position
        self.frame_number = frame_number
        # Set the video position to the selected frame
        self.cap.set(cv2.CAP_PROP_POS_FRAMES, self.frame_number)
        self.ghost.set(cv2.CAP_PROP_POS_FRAMES, self.frame_number)
        # Read the frame at the selected position, ret = boolean for successful read of frame
        ret, frame = self.cap.read()
        retG, frameG = self.ghost.read()

        if ret and retG:
            # Overlay DFB on the current frame (directfeedback prototype), this calls the functions for each
changing frame
            self.DFBoverlay(frame)
            self.guidance(frame)
            #if self.PBcenterofmass:
                #self.center_of_mass(frame)

            if self.PBcompareyourself:
                # blend both frames to show a ghost comparison and display its skill level
                # cv2.putText(frame, comparison_text, (20, 40), cv2.FONT_HERSHEY_SIMPLEX, 1,
                #             (self.color_white),
                #             self.line_thickness)
                comparison_text = 'Comparing somersault ' + str(self.somersault_number) + ' with ' + str(
                    self.somersault_ghost)
                rating_text = 'Ghost rating: ' + str(self.rating[self.somersault_ghost])
                cv2.putText(frame, comparison_text, (20, 40), cv2.FONT_HERSHEY_SIMPLEX, 1,
                            (self.color_white),
                            self.line_thickness)
                cv2.putText(frame, rating_text, (20, 80), cv2.FONT_HERSHEY_SIMPLEX, 1,
                            (self.color_white),
                            self.line_thickness)
                blended_frame = cv2.addWeighted(frame, 1 - self.alpha, frameG, self.alpha, 0)
                cv2.imshow('Video Player', blended_frame)  # update shown frame with trackbar

            else:
                cv2.imshow('Video Player', frame)  # update shown frame with trackbar
```

```
    def play_video(self, resize):
        # Check if a video is there
        # This whole function can be considered as a constructor of the video window, it is subsequently updated in
        # the 'on_trackbar_change' function. I love sloths as well they are very cute, especially as babies
        if not self.video_files:
            print("No video files found.")
            return
        self.cap = cv2.VideoCapture(self.video_files[self.somersault_number])  # Select 'main 'video based on input
number
        self.ghost = cv2.VideoCapture(self.video_files[self.somersault_ghost])

        # If unable to open video file
        if not self.cap.isOpened():
            print("Error: Unable to open video file:", self.video_files[self.somersault_number])
            return

        # Read the first frame
        ret, frame = self.cap.read()
        retG, frameG = self.ghost.read()

        # If the frame is read successfully, display it initially
        if ret:
            # Overlay shapes on the initial frame
            self.DFBoverlay(frame)
            self.guidance(frame)
            #self.center_of_mass(frame)

            # Display the initial frame
            cv2.imshow('Video Player', frame)  # initiliasation of frame, needed to avoid nullpointer at trackbar

        # Create a trackbar for changing frames
        cv2.createTrackbar('Frame', 'Video Player', 0, int(self.cap.get(cv2.CAP_PROP_FRAME_COUNT)) - 1,
                           self.on_trackbar_change)

        while True:
            key = cv2.waitKey(1)
            # Check if 'q' is pressed to exit the loop
            if key & 0xFF == ord('q'):
                break
            elif key == ord('.'):  # Move forward one frame on '.'
                self.frame_number += 1
                self.on_trackbar_change(self.frame_number)
            elif key == ord(','):  # Move backward one frame on ','
                self.frame_number -= 1
                self.on_trackbar_change(self.frame_number)
            cv2.setTrackbarPos('Frame', 'Video Player', self.frame_number)

        self.cap.release()
        cv2.destroyAllWindows()

    def DFBoverlay(self, frame):
        # optional grid:
        if self.gridX:
            # draw horizontal lines on screen with spacing specified by 'self.gridspacing'
            for y in range(0, frame.shape[0], self.gridspacing):
                cv2.line(frame, (0, y), (frame.shape[1], y), self.color_grey, 1)
        if self.gridY:
            # draw vertical lines on screen with spacing specified by 'self.gridspacing'
            for x in range(0, frame.shape[1], self.gridspacing):
                cv2.line(frame, (x, 0), (x, frame.shape[0]), self.color_grey, 1)

        # Select body part, calculate and display angles between
        if self.PBdirectfeedback:
            self.show_grid = True
            # Get the values that correspond to body parts of choice (choose in __init__ above)
            # we convert to int bad argument (why the fk tho i already round in import_keypoint_values #IIWIW)
            keypoint_valuesN = globals()[f'keypoint_values{self.somersault_number}']

            startX = int(keypoint_valuesN[self.frame_number][self.x][self.selected_bodypart1])
            startY = int(keypoint_valuesN[self.frame_number][self.y][self.selected_bodypart1])
            midX = int(keypoint_valuesN[self.frame_number][self.x][self.selected_bodypart2])
            midY = int(keypoint_valuesN[self.frame_number][self.y][self.selected_bodypart2])  # become midhip [8]
            endX = int(keypoint_valuesN[self.frame_number][self.x][self.selected_bodypart3])
            endY = int(keypoint_valuesN[self.frame_number][self.y][self.selected_bodypart3])

            if startX != 0 and startY != 0 and midX != 0 and midY != 0:
                # Draw lines between body parts (knee to midhip to neck)
                cv2.line(frame, (startX, startY), (midX, midY), (self.color_white), self.line_thickness)
            if midX != 0 and midY != 0 and endX != 0 and endY != 0:
                cv2.line(frame, (midX, midY), (endX, endY), (self.color_white), self.line_thickness)
            # print(self.frame_number)
```

```
            # Draw circles at the points for clarity of location
            cv2.circle(frame, (startX, startY), self.radius, (self.color_blue), self.fill_shape)
            cv2.circle(frame, (midX, midY), self.radius, (self.color_green), self.fill_shape)
            cv2.circle(frame, (endX, endY), self.radius, (self.color_orange), self.fill_shape)

            # Calculate angle if all points have non-zero coordinates
            if startX != 0 and startY != 0 and midX != 0 and midY != 0 and endX != 0 and endY != 0:
                # Calculate vectors
                vec1 = [startX - midX, startY - midY]
                vec2 = [endX - midX, endY - midY]

                # Calculate dot product and magnitudes
                dot_product = vec1[0] * vec2[0] + vec1[1] * vec2[1]
                magnitude_vec1 = math.sqrt(vec1[0] ** 2 + vec1[1] ** 2)
                magnitude_vec2 = math.sqrt(vec2[0] ** 2 + vec2[1] ** 2)

                # Calculate angle in degrees, we round this (no significance and more clarity)
                angle = round(math.degrees(math.acos(dot_product / (magnitude_vec1 * magnitude_vec2))))

                # Draw angle text on frame
                angle_text = f"Angle: {angle:d} degrees"
            else:
                # If any point is undetected (ie value = 0), show angle 'unknown'
                angle_text = "Angle: Unknown"

            # Display angle text on the frame
            cv2.putText(frame, angle_text, (midX + 200, midY), cv2.FONT_HERSHEY_SIMPLEX, 1,
                        (self.color_green), self.line_thickness, cv2.LINE_AA)
        return

    def compare_yourself(self, frame):
        # classic troll, function is in on_trackbar_change
        return

    def guidance(self, frame): #Note: this also includes CoM calculations
        if self.PBguidance:

            # in this function we filter the keypoints values of somersault and ghost with a butter LP, used for
Guidance
            keypoint_values = globals()[f'keypoint_values{self.somersault_number}']  # use the current
keypoint_values
            keypoint_values_ghost = globals()[
                f'keypoint_values{self.somersault_ghost}']  # use the current keypoint_values

            # extract x,y from keypoint_values (clarity)
            x_coords = np.array(keypoint_values[:, self.x, self.selected_bodypart4])
            y_coords = np.array(keypoint_values[:, self.y, self.selected_bodypart4])

            # extract x,y from keypoint_values (clarity)
            x_coords_ghost = np.array(keypoint_values_ghost[:, self.x, self.selected_bodypart4])
            y_coords_ghost = np.array(keypoint_values_ghost[:, self.y, self.selected_bodypart4])

            # Mask zero values (needed to exclude undetected points)
            mask = x_coords != 0  # Create a mask for non-zero values in x_coords
            x_masked = x_coords[mask]
            y_masked = y_coords[mask]

            # Mask zero values
            mask_ghost = x_coords_ghost != 0  # Create a mask for non-zero values in x_coords
            x_masked_ghost = x_coords_ghost[mask_ghost]
            y_masked_ghost = y_coords_ghost[mask_ghost]

            # Define parameters for the lowpass filter
            fs = 60  # Sample rate, change this according to your data
            cutoff_frequency = 4  # Desired cutoff frequency of the filter [5 default
            order = 5  # Filter order

            # Compute the Nyquist frequency
            nyquist_frequency = 0.5 * fs
            # Normalize the cutoff frequency
            normalized_cutoff_frequency = cutoff_frequency / nyquist_frequency
            # Design a lowpass Butterworth filter
            b, a = butter(order, normalized_cutoff_frequency, btype='low', analog=False)

            # Apply lowpass filter to x and y coordinates separately
            self.filtered_x = np.round(filtfilt(b, a, x_masked)).astype(int)
            self.filtered_y = np.round(filtfilt(b, a, y_masked)).astype(int)

            self.filtered_x_ghost = np.round(filtfilt(b, a, x_masked_ghost)).astype(int)
            self.filtered_y_ghost = np.round(filtfilt(b, a, y_masked_ghost)).astype(int)
            # Initialize a list to store the coordinates of 'RKnee' keypoints
```

```
            keypoints_list = []

            # Loop through all frames to collect 'RKnee' keypoints
            for kps in range(len(keypoint_values)):
                # Get the 'RKnee' keypoint coordinates for the current frame

                startX = int(keypoint_values[kps][self.x][self.selected_bodypart4])
                startY = frame.shape[0] - int(
                    keypoint_values[kps][self.y][self.selected_bodypart4])  # y values are flipped (else
                # its upside down)

                # Check if the coordinates are valid (non-zero)
                if startX != 0 and startY != 0:
                    # Flip the y-coordinate to adjust for the coordinate system difference
                    startY = frame.shape[0] - startY
                    # Add x,y vals to a keypoints list
                    keypoints_list.append((startX, startY))
            if self.show_rawvals:
                # Draw all the keypoints on the current frame
                for point in keypoints_list:
                    cv2.circle(frame, point, self.radius, (self.color_orange), -1)  # Orange color
                    # Draw connecting lines between keypoints
                    if len(keypoints_list) > 1:
                        for i in range(len(keypoints_list) - 1):
                            cv2.line(frame, keypoints_list[i], keypoints_list[i + 1], (self.color_orange),
                                     self.line_thickness)

            keypoint_partition = (len(keypoints_list) / 4)
            keypoint_partition = int(
                keypoint_partition)  # has to be on a separate line because the dark forces are rising
            # print(type(keypoint_values))
            keypoints_list = keypoints_list[keypoint_partition:-25]

            # Fit a polynomial curve to the keypoints
            if len(keypoints_list) > 1:
                x = np.array([point[0] for point in keypoints_list])
                y = np.array([point[1] for point in keypoints_list])

                # Fit a polynomial curve of degree 3 (you can adjust the degree as needed)
                z = np.polyfit(x, y, self.poly_curvefit_degree)
                p = np.poly1d(z)

                # Evaluate the polynomial at several points to draw the fitted curve
                fitted_curve_points = [(int(x_val), int(p(x_val))) for x_val in
                                       np.linspace(min(x), max(x), self.fitted_curve_points_resolution)]

                # Draw the fitted curve (colored black)
                if self.show_trendline:
                    for i in range(len(fitted_curve_points) - 1):
                        cv2.line(frame, fitted_curve_points[i], fitted_curve_points[i + 1], (self.color_black),
                                 self.line_thickness)

                if self.PBcenterofmass == False:
                    # Draw the filtered keypoints (filtered_x, filtered_y)
                    if self.filtered_x is not None and self.filtered_y is not None and self.show_filteredvals:
                        for i in range(
                                len(self.filtered_x) - 1):  # doesn't matter x,y have same length but i choose x
cuz I'm Xtremed
                            # Draw a line between the current point and the next point
                            cv2.line(frame, (self.filtered_x[i], self.filtered_y[i]), (self.filtered_x[i + 1],
self.filtered_y[
                                i + 1]), self.color_blue, self.line_thickness)

                    # Draw the filtered keypoints of the ghost (filtered_x_ghost, filtered_y_ghost)
                    if self.filtered_x_ghost is not None and self.filtered_y_ghost is not None and
self.compare_with_ghost:
                        for i in range(len(self.filtered_x_ghost) - 1):
                            # Draw a line between the current point and the next point
                            cv2.line(frame, (self.filtered_x_ghost[i], self.filtered_y_ghost[i]),
                                     (self.filtered_x_ghost[i + 1], self.filtered_y_ghost[
                                         i + 1]), self.color_grey, self.line_thickness)
                            rating_text = 'Ghost rating: ' + str(self.rating[self.somersault_ghost])
                            comparison_text = 'Comparing somersault ' + str(self.somersault_number) + ' with ' +
str(
                                self.somersault_ghost)
                            cv2.putText(frame, comparison_text, (20, 40), cv2.FONT_HERSHEY_SIMPLEX, 1,
                                        (self.color_white),
                                        self.line_thickness)
                            cv2.putText(frame, rating_text, (20, 80), cv2.FONT_HERSHEY_SIMPLEX, 1,
                                        (self.color_white),
                                        self.line_thickness)
```

```
########################################################################################################################

########################################################################################################################

########################################################################################################################

########################################################################################################################

########################################################################################################################
                                                    #CENTER OF MASS
########################################################################################################################

########################################################################################################################

########################################################################################################################

########################################################################################################################

########################################################################################################################


    # This is being done here cuz cv2 gives unsolvable (by a third degree programmer) scalar to vector error on
    # drawing line in a separate function. Enable both PB.guidance and PB.centerofmass for CoM prototype


            if self.PBcenterofmass:
                # in this function we filter the keypoints values of somersault and ghost with a butter LP, used for
Guidance
                keypoint_values = globals()[
                    f'keypoint_values{self.somersault_number}']  # use the current keypoint_values
                # print('lenkpval:', len(keypoint_values))
                loop = 18  # it gives error at 18+ because of some ancient curse on the filtfilt function, don't
need more than index17
                for i in range(len(self.bodyparts)):
                    # print('i=', i)
                    # extract x,y from keypoint_values (clarity)
                    self.x_coords = np.array(keypoint_values[:, self.x, i])
                    self.y_coords = np.array(keypoint_values[:, self.y, i])

                    # print(self.x_coords)

                    # Mask zero values (needed to exclude undetected points by OpenPose)
                    mask = self.x_coords != 0  # Create a mask for non-zero values in self.x_coords
                    x_masked = self.x_coords[mask]
                    y_masked = self.y_coords[mask]

                    # Define parameters for the lowpass filter
                    fs = 60  # Sample rate, this is 60Hz (fps)
                    cutoff_frequency = 4  # f-cut is chosen for best performing value (ie, no overfit, significant
filtering)
                    order = 5  # Filter order, higher order is insignificant

                    # Compute the Nyquist frequency
                    nyquist_frequency = 0.5 * fs
                    # Normalize the cutoff frequency
                    normalized_cutoff_frequency = cutoff_frequency / nyquist_frequency
                    # Design a lowpass Butterworth filter
                    b, a = butter(order, normalized_cutoff_frequency, btype='low', analog=False)

                    # Apply lowpass filter to x and y coordinates separately
                    self.filtered_x = np.round(filtfilt(b, a, x_masked)).astype(int)
                    self.filtered_y = np.round(filtfilt(b, a, y_masked)).astype(int)

                    # Optional plotting to confirm filtering process, plots all bodyparts raw vs filtered 1 by 1
(close w/ q)
                    # plt.figure(figsize=(12, 6))
                    # plt.gca().invert_yaxis()
                    # plt.plot(x_masked, y_masked, label='Raw')
                    # plt.plot(self.filtered_x, self.filtered_y, label='Processed')
                    # plt.xlabel('Frame Index')
                    # plt.ylabel('Coordinate Value')
```

```
                    # plt.title(f'Keypoint Coordinates and Filtered Coordinates ({self.bodyparts[i]})')
                    # plt.legend()
                    # plt.grid(True)
                    # plt.show()

                    # Store filtered_x and filtered_y in dynamically named dictionary with keys 'filtered_x_N' and
                    # 'filtered_y_N' where N is an integer corresponding to the bodypart indexation by OpenPose
                    filtered_x_key = f'filtered_x_{i}'
                    filtered_y_key = f'filtered_y_{i}'

                    if filtered_x_key not in self.filtered_data:
                        self.filtered_data[filtered_x_key] = []
                    if filtered_y_key not in self.filtered_data:
                        self.filtered_data[filtered_y_key] = []

                    self.filtered_data[filtered_x_key].append(list(self.filtered_x))
                    self.filtered_data[filtered_y_key].append(list(self.filtered_y))

                    ################################### END OF GUIDANCE ######################

                # Calculation of Center of Mass (CoM)
                # Segmentation method is applied where we use the CoM of segments and sum them weighted

                # Assumptions in these calculations:
                # - Segments are not precisely represented by OpenPose, the closest keypoints are used to represent
segment
                # - Weight of hands and feet are not accounted for, since they are not represented well by OpenPose
and
                #   contribute little to the overall CoM compared to other segments
                # - Weight of segments is linearly distributed (i.e., CoM of a segment is at its center)
                # First we extract all keypoint values for body segments:
                #
                #         # Keypoints approximation                Segment (bodypart with known mass)
Mass Percentage
                #         # ---------------------------------------------------------------------------------------
--------------
                #         # Neck - Reye                            Head & Neck
7.6
                #         # Neck - RHip/Midhip                     Trunk
45.2
                #         # RShoulder- RElbow                     Upper arm (R)                                2.9
                #         # LShoulder - LElbow                      Upper Arm (L)
2.9
                #         # RElbow - RWrist                         Forearm (R)
2.4
                #         # LElbow - LWrist                         Forearm (L)
2.4
                #         # RHip-RKnee                              Thigh (R)
11.9
                #         # LHip-LKnee                             Thigh (L)
11.9
                #         # RKnee-RAnkle                            Shank (R)
6.0
                #         # LKnee -LAnkle                         Shank (L)                                    6.0

                #                                                                                          total:
99%

                #         NOTE: Total is 99% due to certain body parts not included in figures (such as hands) and
rounding.
                #         Rounding is done since the keypoint accuracy is considered more significant than segment
figures


                # Reference for accessing the dict: self.filtered_data[key][row 1][frame value]
                # [key]: f'filtered_x_{self.Neck}' in which example self.Neck is integer corr. 2 bodypart
                # [row 1]: every dict key has only 1 row so should be 0 always
                # [frame value]: this specifies the frame and should be equal to used loop variable 'i'

                # Calculations of CoM Segments

                # General formula for CoM with uniform weight distrubutions for a body formed by points in space
                # (a,b) and (c,d):

                # CoM_x = ((m1*a)+(m2*c))/(m1+m2)
                # with m1=m2 (uniform weight) this simplifies, and similar approach for y we get:

                # CoM_x = (a+c)/2
                # CoM_y = (b+d)/2

                # Where in our case a,b,c,d are x,y coordinates (in px) of body segments.
                # We calculate a list of CoM'ses, for each segment and then take the weighted sum to find CoM_total:
```

```
                # CoM_total = sum(weight

                # Neck_x, Neck_y, REye_x, REye_y, RHip_x, RHip_y, MidHip_x, MidHip_y, RShoulder_x, RShoulder_y,
RElbow_x, \
                # RElbow_y, LShoulder_x, LShoudler_y, LElbow_x, LElbow_y, RWrist_x, RWrist_y, LWrist_x, LWrist_y, \
                # RKnee_x, RKnee_y, LHip_x, LHip_y, LKnee_x, LKnee_y, RAnkle_x, RAnkle_y, LAnkle_x, LAnkle_y = \
                #     [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [],\
                #     [], [], [], [], [], [], [], [], [], [], [], []


                # Because we can't directly access the dict we first store x,y coordinates
                # ! For the aestethic code geeks the following section might be perceived as shocking !
                # Initialization of lists such that they are empty with appropriate length
                Neck_x = [[] for _ in range(len(self.filtered_data[filtered_x_key][0]))]
                Neck_y = [[] for _ in range(len(self.filtered_data[filtered_x_key][0]))]

                # Initialize other lists in a similar manner
                REye_x = [[] for _ in range(len(self.filtered_data[filtered_x_key][0]))]
                REye_y = [[] for _ in range(len(self.filtered_data[filtered_x_key][0]))]

                RHip_x = [[] for _ in range(len(self.filtered_data[filtered_x_key][0]))]
                RHip_y = [[] for _ in range(len(self.filtered_data[filtered_x_key][0]))]

                MidHip_x = [[] for _ in range(len(self.filtered_data[filtered_x_key][0]))]
                MidHip_y = [[] for _ in range(len(self.filtered_data[filtered_x_key][0]))]

                RShoulder_x = [[] for _ in range(len(self.filtered_data[filtered_x_key][0]))]
                RShoulder_y = [[] for _ in range(len(self.filtered_data[filtered_x_key][0]))]

                RElbow_x = [[] for _ in range(len(self.filtered_data[filtered_x_key][0]))]
                RElbow_y = [[] for _ in range(len(self.filtered_data[filtered_x_key][0]))]

                LShoulder_x = [[] for _ in range(len(self.filtered_data[filtered_x_key][0]))]
                LShoulder_y = [[] for _ in range(len(self.filtered_data[filtered_x_key][0]))]

                LElbow_x = [[] for _ in range(len(self.filtered_data[filtered_x_key][0]))]
                LElbow_y = [[] for _ in range(len(self.filtered_data[filtered_x_key][0]))]

                RWrist_x = [[] for _ in range(len(self.filtered_data[filtered_x_key][0]))]
                RWrist_y = [[] for _ in range(len(self.filtered_data[filtered_x_key][0]))]

                LWrist_x = [[] for _ in range(len(self.filtered_data[filtered_x_key][0]))]
                LWrist_y = [[] for _ in range(len(self.filtered_data[filtered_x_key][0]))]

                RKnee_x = [[] for _ in range(len(self.filtered_data[filtered_x_key][0]))]
                RKnee_y = [[] for _ in range(len(self.filtered_data[filtered_x_key][0]))]

                LHip_x = [[] for _ in range(len(self.filtered_data[filtered_x_key][0]))]
                LHip_y = [[] for _ in range(len(self.filtered_data[filtered_x_key][0]))]

                LKnee_x = [[] for _ in range(len(self.filtered_data[filtered_x_key][0]))]
                LKnee_y = [[] for _ in range(len(self.filtered_data[filtered_x_key][0]))]

                RAnkle_x = [[] for _ in range(len(self.filtered_data[filtered_x_key][0]))]
                RAnkle_y = [[] for _ in range(len(self.filtered_data[filtered_x_key][0]))]

                LAnkle_x = [[] for _ in range(len(self.filtered_data[filtered_x_key][0]))]
                LAnkle_y = [[] for _ in range(len(self.filtered_data[filtered_x_key][0]))]

                CoM_total_x = [[] for _ in range(len(self.filtered_data[filtered_x_key][0]))]
                CoM_total_y = [[] for _ in range(len(self.filtered_data[filtered_x_key][0]))]


                for i in range(len(self.filtered_data[filtered_x_key][0])):
                    Neck_x[i].append(self.filtered_data[f'filtered_x_{self.Neck}'][0][i])
                    Neck_y[i].append(self.filtered_data[f'filtered_y_{self.Neck}'][0][i])

                    # # Append values to REye lists
                    # REye_x[i].append(self.filtered_data[f'filtered_x_{self.REye}'][0][i])
                    # REye_y[i].append(self.filtered_data[f'filtered_y_{self.REye}'][0][i])

                    # Append values to RHip lists
                    RHip_x[i].append(self.filtered_data[f'filtered_x_{self.RHip}'][0][i])
                    RHip_y[i].append(self.filtered_data[f'filtered_y_{self.RHip}'][0][i])

                    # Append values to MidHip lists
                    MidHip_x[i].append(self.filtered_data[f'filtered_x_{self.MidHip}'][0][i])
                    MidHip_y[i].append(self.filtered_data[f'filtered_y_{self.MidHip}'][0][i])

                    # Append values to RShoulder lists
                    RShoulder_x[i].append(self.filtered_data[f'filtered_x_{self.RShoulder}'][0][i])
                    RShoulder_y[i].append(self.filtered_data[f'filtered_y_{self.RShoulder}'][0][i])
```

```
            # Append values to RElbow lists
            RElbow_x[i].append(self.filtered_data[f'filtered_x_{self.RElbow}'][0][i])
            RElbow_y[i].append(self.filtered_data[f'filtered_y_{self.RElbow}'][0][i])

            # Append values to LShoulder lists
            LShoulder_x[i].append(self.filtered_data[f'filtered_x_{self.LShoulder}'][0][i])
            LShoulder_y[i].append(self.filtered_data[f'filtered_y_{self.LShoulder}'][0][i])

            # Append values to LElbow lists
            LElbow_x[i].append(self.filtered_data[f'filtered_x_{self.LElbow}'][0][i])
            LElbow_y[i].append(self.filtered_data[f'filtered_y_{self.LElbow}'][0][i])

            # Append values to RWrist lists
            RWrist_x[i].append(self.filtered_data[f'filtered_x_{self.RWrist}'][0][i])
            RWrist_y[i].append(self.filtered_data[f'filtered_y_{self.RWrist}'][0][i])

            # Append values to LWrist lists
            LWrist_x[i].append(self.filtered_data[f'filtered_x_{self.LWrist}'][0][i])
            LWrist_y[i].append(self.filtered_data[f'filtered_y_{self.LWrist}'][0][i])

            # Append values to RKnee lists
            RKnee_x[i].append(self.filtered_data[f'filtered_x_{self.RKnee}'][0][i])
            RKnee_y[i].append(self.filtered_data[f'filtered_y_{self.RKnee}'][0][i])

            # Append values to LHip lists
            LHip_x[i].append(self.filtered_data[f'filtered_x_{self.LHip}'][0][i])
            LHip_y[i].append(self.filtered_data[f'filtered_y_{self.LHip}'][0][i])

            # Append values to LKnee lists
            LKnee_x[i].append(self.filtered_data[f'filtered_x_{self.LKnee}'][0][i])
            LKnee_y[i].append(self.filtered_data[f'filtered_y_{self.LKnee}'][0][i])

            # Append values to RAnkle lists
            RAnkle_x[i].append(self.filtered_data[f'filtered_x_{self.RAnkle}'][0][i])
            RAnkle_y[i].append(self.filtered_data[f'filtered_y_{self.RAnkle}'][0][i])

            # Append values to LAnkle lists
            LAnkle_x[i].append(self.filtered_data[f'filtered_x_{self.LAnkle}'][0][i])
            LAnkle_y[i].append(self.filtered_data[f'filtered_y_{self.LAnkle}'][0][i])

            # Calculate CoM for x and y coordinates
            # Calculate CoM for x and y coordinates
            CoM_curr_x = int
            CoM_curr_y = int


            CoM_curr_x = sum(Neck_x[i],MidHip_x[i])
            CoM_curr_y = sum(Neck_y[i],MidHip_y[i])


            CoM_total_x[i] = (int(CoM_curr_x)/2)
            CoM_total_y[i] = (int(CoM_curr_y)/2)


    # CoM Trunk:
        # Calculate angle of the trunk (angle of inclination)
        angle_rad = np.arctan2(MidHip_y - Neck_y, MidHip_x - Neck_x)  # Angle in radians of Trunk

        # Calculate center of mass for the trunk
        CoM_Trunk_x = (Neck_x + MidHip_x) / 2  # Average x-coordinate
        CoM_Trunk_y = (Neck_y + MidHip_y) / 2  # Average y-coordinate

        # Get the CoM, taking inclination into account
        CoM_Trunk = (CoM_Trunk_x - CoM_Trunk_y * np.tan(angle_rad), 0)


    #Following body parts are calculated in the same way

    # CoM Upper Arm (R):
        angle_rad = np.arctan2(RShoulder_y - RWrist_y, RShoulder_x - RWrist_x)  # Angle in radians of Trunk
        CoM_UpArmR_x = (RElbow_x + RShoulder_x) / 2  # Average x-coordinate
        CoM_UpArmR_y = (RElbow_y + RShoulder_y) / 2  # Average y-coordinate
        CoM_UpArmR = (CoM_UpArmR_x - CoM_UpArmR_y * np.tan(angle_rad), 0)


    # CoM Upper Arm (L):
        angle_rad = np.arctan2(LShoulder_y - LElbow_y, LShoulder_x - LElbow_x)  # Angle in radians of Trunk
        CoM_UpArmL_x = (LElbow_x + LShoulder_x) / 2  # Average x-coordinate
        CoM_UpArmL_y = (LElbow_y + LShoulder_y) / 2  # Average y-coordinate
        CoM_UpArmL = (CoM_UpArmL_x - CoM_UpArmL_y * np.tan(angle_rad), 0)
```

```
        # CoM ForeArm (R):
                angle_rad = np.arctan2(RElbow_y - RWrist_y, RElbow_x - RWrist_x)  # Angle in radians of Trunk
                CoM_ForeArmR_x = (RWrist_x + RElbow_x) / 2  # Average x-coordinate
                CoM_ForeArmR_y = (RWrist_y + RElbow_y) / 2  # Average y-coordinate
                CoM_ForeArmR = (CoM_ForeArmR_x - CoM_ForeArmR_y * np.tan(angle_rad), 0)

        # CoM ForeArm (L):
                angle_rad = np.arctan2(LElbow_y - LWrist_y, LElbow_x - LWrist_x)  # Angle in radians of Trunk
                CoM_ForeArmL_x = (LWrist_x + LElbow_x) / 2  # Average x-coordinate
                CoM_ForeArmL_y = (LWrist_y + LElbow_y) / 2  # Average y-coordinate
                CoM_ForeArmL = (CoM_UpArmL_x - CoM_UpArmL_y * np.tan(angle_rad), 0)

        # CoM Thigh (R):
                angle_rad = np.arctan2(RHip_y - RKnee_y, RHip_x - RKnee_x)  # Angle in radians of Trunk
                CoM_ThighR_x = (RHip_x + RKnee_x) / 2  # Average x-coordinate
                CoM_ThighR_y = (RHip_y + RKnee_y) / 2  # Average y-coordinate
                CoM_ThighR = (CoM_ThighR_x - CoM_ThighR_y * np.tan(angle_rad), 0)

        # CoM Thigh (L):
                angle_rad = np.arctan2(LHip_y - LKnee_y, LHip_x - LKnee_x)  # Angle in radians of Trunk
                CoM_ThighL_x = (LHip_x + LKnee_x) / 2  # Average x-coordinate
                CoM_ThighL_y = (LHip_y + LKnee_y) / 2  # Average y-coordinate
                CoM_ThighL = (CoM_ThighL_x - CoM_ThighL_y * np.tan(angle_rad), 0)

        # CoM Shank (R):
                angle_rad = np.arctan2(RAnkle_y - RKnee_y, RAnkle_y - RKnee_x)  # Angle in radians of Trunk
                CoM_ShankR_x = (RKnee_x + RAnkle_x) / 2  # Average x-coordinate
                CoM_ShankR_y = (RKnee_y + RAnkle_y) / 2  # Average y-coordinate
                CoM_ShankR = (CoM_ShankR_x - CoM_ShankR_y * np.tan(angle_rad), 0)

        # CoM Shank (L):
                angle_rad = np.arctan2(LAnkle_y - LKnee_y, LAnkle_y - LKnee_x)  # Angle in radians of Trunk
                CoM_ShankL_x = (LKnee_x + LAnkle_x) / 2  # Average x-coordinate
                CoM_ShankL_y = (LKnee_y + LAnkle_y) / 2  # Average y-coordinate
                CoM_ShankL = (CoM_ShankL_x - CoM_ShankL_y * np.tan(angle_rad), 0)

                # Total CoM calculation:
                # CoM_total = Sum(CoM_segmentA * Weight_segmentA)
                CoM_segments = [CoM_Trunk, CoM_UpArmR, CoM_UpArmL, CoM_ForeArmR, CoM_ForeArmL, CoM_ThighR,
CoM_ThighL,
                                CoM_ShankR, CoM_ShankL]
                CoM_segment_weights = [45.2, 2.9,2.9,2.4,2.4,11.9,11.9,6.0,6.0]

                # Loop through the segments and their weights, add all parts to find the CoM x,y coordinates
                for segment in CoM_segments:
                    CoM_total_x = CoM_total_x + (CoM_segments[segment[x]]*CoM_segment_weights[segment[x]])
                    CoM_total_y = CoM_total_y + (CoM_segments[segment[x]]*CoM_segment_weights[segment[x]])

                # Plot CoM for reference
                # plt.figure(5)
                # plt.plot(CoM_total_x, CoM_total_y)
                # plt.gca().invert_yaxis()
                # plt.show()

                # Show the center of Mass together with the CoM of previous frames

                tail = 250 # number of CoM tailing elements ('frames back' to display CoM of)

                # Calculate brightness factor for each CoM in the tail
                brightness_values = np.linspace(255, 0, tail + 1, dtype=np.uint8)

                # Loop through previous frames within the tail range
                for i in range(max(0, self.frame_number - tail), self.frame_number + 1):
                    # Calculate brightness factor based on position in tail
                    brightness_index = self.frame_number - i
                    brightness = brightness_values[brightness_index]

                    # Create color using brightness value
                    color = (int(brightness), int(brightness), int(brightness))

                    # Draw CoM for the current frame with adjusted brightness
                    cv2.circle(frame, (int(CoM_total_x[i]), int(CoM_total_y[i])), self.radius, color, -1)

            return
```

quantitative_analysis.py :

```python
# Quantitative Evaluation of the Performance of OpenPose on Somersaults for Different Camera Angles
# Written by Niels van Dalen
# Last edited: 28-07-2023
# The code does the following (structured by Class):
import math
# 1 - DirectoryScanner
# Searches all subdirectories under a defined main/top-level directory (root_path).
# Each subdirectory should contain normalized JSONs of a single video (as output by OpenPose)

# 2 - JSONProcessor
# For each subdirectory, gets all JSONs and extract keypoints for each individual file (which corresponds to one frame)
# Each json file is stored as a dict (frame_data) containing its subdirectory, file name, and x,y,c values.

# 3 - KeypointCalculations
# Evaluate the performance of openpose over the frames, this consists of two parallel metrics:
#     a) over raw points and their performance
#     b) over the points that are weighted according to their relevance for a somersault (REFERENCE)
#
#     The evaluation properties are:
#     - Detection rate (number of found points / theoretical maximum of points)
#     - Dropped frames (frames without any detection points)
#     - Accuracy of keypoints for 1 somersault (compared to hand drawn (reference))

# COMMENT LEGENDA:
# LOL = list of lists
# KP = keypoint

# Libraries
import re  # numeric sorting
import os
import json
import numpy as np
from scandir import walk
import matplotlib.pyplot as plt
from itertools import groupby


class DirectoryScanner:
    def __init__(self, root_path):
        self.root_path = root_path
        self.subdir_list = []
        self.leafdir_list = []

    def extract_number_from_filename(self, filename):  # used for sorting by the folder number in the name
        # Extract the numeric part from the filename (N) using regular expressions
        match = re.search(r'(\d+)salto@deg0', filename)
        if match:
            return int(match.group(1))
        return 0  # Return 0 if no numeric part is found

    def scan_directories(self):  # Scan all the subdirectories under root_path (specify in main) and store in list
        for root, dirs, _ in walk(self.root_path):
            if root == self.root_path:
                continue
            self.subdir_list.append(root)
            # print(self.subdir_list) # prints the full dir list (ie. C/users/niels...../jsonfilefolder)

    def get_subdirectories(self):  # return the list of subdirectories
        return self.subdir_list
```

```
class JSONProcessor:  # In 'subdir_list' check for JSON files and extract (only) the keypoints
in there
    def __init__(self, subdir_list, keypoint_attributes=3):
        self.subdir_list = subdir_list
        self.json_files = []  # list for storing the json file names (print if unsure your
files are being seen)
        self.number_of_frames = []  # list that stores the amount of json per folder, for
finding an average frames
        self.keypoint_attributes = keypoint_attributes  # amount of keypoint attributes, here
3: an x, y and c value (c = confidence of estimation)
        self.frame_data = [[] for _ in self.subdir_list]  # LOL of all raw keypoint data
        self.entries_without_people_key = [[] for _ in self.subdir_list]
        self.amount_of_keypoints = 25  # Hardcode please reach out to 06 8182 1571 if u want
to cry about it

    def process_json_files(self):
        directory_scanner = DirectoryScanner(self.subdir_list)
        for i, subdir in enumerate(self.subdir_list):  # Loop through all subdirectories under
'root_path' (see main)
            current_jsons = [pos_json for pos_json in os.listdir(subdir) if
pos_json.endswith('.json')]
            self.json_files.append(current_jsons)  # every file ending in .json gets added to
current_jsons
            subdir_name = os.path.basename(subdir)  # get current subdirectory name to make
senseful prints
            # print(f"in directory '{subdir_name}':")  # print the subdirectories
            self.number_of_frames.append(len(self.json_files[i]))

            # print(len(self.json_files[i]), 'JSON files') #amount of json files in a folder
            # print(self.json_files[i], '\n')
            os.chdir(subdir)  # update cwd to current (next) subdir

            for j, json_file in enumerate(current_jsons):  # Loop through amount of json files
in subdirectory i
                with open(json_file) as open_json:
                    json_str = json.load(open_json)
                    # print(json_str)
                    # print(current_jsons[j]) #DEBUG: Current jsons contains all json files
(also empty ones)
                    # Create empty zeros lists corresponding to keypoint size (= 25)
                    xVals = [0] * self.amount_of_keypoints
                    yVals = [0] * self.amount_of_keypoints
                    cVals = [0] * self.amount_of_keypoints

                if 'people' in json_str and len(json_str['people']) > 0:  # This is always
true also for empty files

                    if 'pose_keypoints_2d' in json_str['people'][0] and any(json_str['people']
[0]['pose_keypoints_2d']):
                        keypoints = json_str['people'][0][
                            'pose_keypoints_2d']  # if no 'pose_keypoints_2d' then error List
OOR
                        self.amount_of_keypoints = int(
                            len(keypoints) / self.keypoint_attributes)  # amount of keypoints
is
                        # the amount of data points / attributes, in this case should always
be 25 = 75 / 3 attributes (x,
                        # y,c). Also convert to int so for loop isn't gonna be a lil bitchboy

                        # order of keypoints in the json is x-y-z: x0, y1, c2, .... x72, y73,
c74
                        x = 0
                        y = 1
                        c = 2

                        for k in range(min(self.amount_of_keypoints, len(keypoints) //
```

```
self.keypoint_attributes)):
                            xVals[k] = keypoints[x]
                            yVals[k] = keypoints[y]
                            cVals[k] = keypoints[c]
                            x += self.keypoint_attributes
                            y += self.keypoint_attributes
                            c += self.keypoint_attributes

                json_file_data = {
                    'subdirectory': subdir_name,
                    'json_file': json_file,
                    'x_vals': xVals,
                    'y_vals': yVals,
                    'c_vals': cVals
                }
                self.frame_data[i].append(json_file_data)  # store the data into the dict
frame_data[]

        # print(self.number_of_frames)  # list containing the amount of frames (jsons) in a
folder, used for avg
        # calculation
        # print(self.frame_data[0][1])
        # weighted_frame_data = self.frame_data
        # print(weighted_frame_data)
        # print(self.frame_data['x_vals'])


class KeypointCalculations:
    def __init__(self, json_processor, avg_frames):
        self.json_processor = json_processor
        self.total_data_points = sum([len(elem) for elem in json_processor.json_files]) \
                                 * json_processor.amount_of_keypoints *
json_processor.keypoint_attributes
        # print('total data points:', self.total_data_points)
        self.empty_frames_sum = self.find_empty_frames()
        self.average_confidence, self.average_confidence_weighted =
self.get_average_confidence()  # Store the average
        # confidence
        self.visualisation = Visualisation(self.average_confidence,
self.average_confidence_weighted, avg_frames,
                                           self.empty_frames_sum)  # for
        # passing
        # confidence to
        # Visualisation

    def get_average_confidence(self):  # Get the average confidence level (c value) per frame
        enable_weighted_keypoints = 0;  # Change to 1 to run over the most relevant keypoints
        average_confidence = []  # List to store the average values of confidence per frame
        average_confidence_weighted = []

        relevant_keypoints = [0,1,3,4,6,7,9,10,11,12,13,14]  # List of relevant keypoints, see
doc for number
        # references

        for element_list in self.json_processor.frame_data:  # nr of loops equal to amount of
folders found
            confidence_avg_per_salto = []
            confidence_avg_per_salto_weighted = []

            for element in element_list:
                c_vals = element['c_vals']  # Extract the c_vals for each element

                # Extract the specific keypoints that are deemed relevant for a somersault:
                c_vals_weighted = [c_vals[i] for i in relevant_keypoints]

                average_c = sum(c_vals) / len(c_vals)  # Calculate the average value of
frames' c_vals > 1
```

```
                average_c_weighted = sum(c_vals_weighted) / len(c_vals_weighted)

                confidence_avg_per_salto.append(average_c)  # - 1 salto
                confidence_avg_per_salto_weighted.append(average_c_weighted)

            average_confidence.append(confidence_avg_per_salto)  # Store all averages in a LoL
- all subdirs
            average_confidence_weighted.append(confidence_avg_per_salto_weighted)

        # print('\n', 'avg confidences:', '\n', average_confidence, '\n')
        #print('\n', 'avg confidences weighted:', '\n', average_confidence_weighted, '\n')

        # print('grand total confidence:', sum(average_confidence) / len(average_confidence))
        # FIND OVERALL CONFIDENCE VALUE
        # print('avgconflen', len(average_confidence))
        # self.visualisation.single_plot_graph_per_subdirectory()

        # print(c_vals_weighted)
        # print('\n', c_vals)
        return average_confidence, average_confidence_weighted

    # def find_empty_frames(self):



class Visualisation:
    def __init__(self, average_confidence, average_confidence_weighted, avg_frames,
empty_frames_sum):
        self.data = [average_confidence, average_confidence_weighted]
        self.avg_frames = int(np.ceil(avg_frames))
        self.empty_frames_sum = empty_frames_sum

    def single_plot_graph_per_subdirectory(self):
        non_empty_subdirs = [subdir_averages for subdir_averages in self.data if
any(subdir_averages)]
        if non_empty_subdirs:
            plt.figure(0)
            for subdir_averages in non_empty_subdirs:
                plt.plot(subdir_averages[:self.avg_frames])  # Use only up to avg_frames
            plt.xlabel('Frame')
            plt.ylabel('Average Confidence')
            plt.title('Average Confidence per Frame for Different Subdirectories')
            plt.legend([f'Somersault {i}' for i in range(1, len(non_empty_subdirs) + 1)])
            plt.show()

    def plot_combined_graph(self):
        # for some clarity
        raw = 0
        weighted = 1

        max_length = min(min(len(sublist) for sublist in self.data[raw]), min(len(sublist) for
sublist in self.data[
            weighted])) # Calculate an consistent array length for all somersaults (which have
different # of frames)

        # Calculate the mean and standard deviation for all raw values
        mean_raw_confidences = np.mean([raw[:max_length] for raw in self.data[raw]], axis=0)
        std_dev_raw_confidences = np.std([raw[:max_length] for raw in self.data[raw]], axis=0)

        # Calculate the mean and standard deviation for all weighted values
        mean_weighted_confidences = np.mean([weighted[:max_length] for weighted in
self.data[weighted]], axis=0)
        std_dev_weighted_confidences = np.std([weighted[:max_length] for weighted in
self.data[weighted]], axis=0)

        # Calculation of Netto confidence values (used in report, variables not used for
making plot)
```

```
        #netto_confidence_raw = sum(mean_raw_confidences) / len(mean_raw_confidences)
        #netto_confidence_weighted = sum(mean_weighted_confidences) /
len(mean_weighted_confidences)
        #print('netto conf raw:', netto_confidence_raw)
        #print('netto conf weighted:', netto_confidence_weighted)

        # Plot properties
        plt.figure(figsize=(10, 5))  # Make it horizontally strechted
        plt.ylim(-0.15, 1)  # For the normalized plot set some boundaries (-.15 cuz else Std
dev gets cut off)

        # Plot raw values with dashed lines at standard deviation values
        plt.plot(mean_raw_confidences, label='Raw Average', color='blue') # Average
        plt.plot(mean_raw_confidences - std_dev_raw_confidences, linestyle='dashed',
color='blue', alpha=0.2) # lower std
        plt.plot(mean_raw_confidences + std_dev_raw_confidences, linestyle='dashed',
color='blue', alpha=0.2, # upper std
                 label='Raw Std. Dev.')

        # Plot weighted values with dashed lines at standard deviation values
        plt.plot(mean_weighted_confidences, label='Weighted Average', color='green')
        plt.plot(mean_weighted_confidences - std_dev_weighted_confidences, linestyle='dashed',
color='green',
                 alpha=0.2) # lower std
        plt.plot(mean_weighted_confidences + std_dev_weighted_confidences, linestyle='dashed',
color='green',
                 alpha=0.2, label='Weighted Std. Dev.') # upper std

        # Title & Labels
        plt.xlabel('Frame Index')
        plt.ylabel('Confidence')
        plt.legend()
        plt.title("OpenPose Keypoint Confidence of Somersaults (n = 115) - Diagonal View") #
Change depending on which
        # data is being processed (see 'root_path' in main)
        plt.show()

    def dropped_frames_plot(self):
        # plt.figure(2)
        plt.figure(figsize=(10, 5))

        plt.plot(self.empty_frames_sum)
        # Title & Labels
        plt.xlabel('Frame Index')
        plt.ylabel('Amount of frames dropped (Normalized)')
        # plt.legend()
        plt.title("OpenPose Normalized Amount of Frames Dropped (n = 115) - Front view")
        plt.show()

def main():
    root_path = 'C:\\Users\\niels\\Documents\\Utwente\\Jaar ' \
                '7\\Graduation\\Technical\\OP\\openpose-1.6.0-binaries-win64-gpu-flir-
3d_recommended\\openpose' \
                '\\outputdata\\normalized\\gymxl\\deg45'
    print('\n')  # make some space

    directory_scanner = DirectoryScanner(root_path)
    directory_scanner.scan_directories()
    subdir_list = directory_scanner.get_subdirectories()
    # Sort the subdir_list based on the numeric values in the subdirectory names. Wups it's in
main boohooo :((
    # it works though whatcha gonna do?
    subdir_list.sort(key=lambda subdir:
directory_scanner.extract_number_from_filename(subdir))

    json_processor = JSONProcessor(subdir_list)
    json_processor.process_json_files()
```

```
    # Filter out subdirectories with 0 JSON files
    valid_subdir_list = [subdir for subdir, num_frames in zip(subdir_list,
json_processor.number_of_frames) if
                         num_frames > 0]
    # Calculate average number of frames for these valid subdirectories (ie. leaf directories
with json files)
    avg_frames = sum(json_processor.number_of_frames) / len(valid_subdir_list)

    avg_frames = math.ceil(sum(json_processor.number_of_frames) / len(valid_subdir_list))

    keypoint_calculator = KeypointCalculations(json_processor, avg_frames)
    average_confidence, average_confidence_weighted =
keypoint_calculator.get_average_confidence()
    empty_frames_sum = keypoint_calculator.find_empty_frames()
    # print(empty_frames)

    visualisation = Visualisation(average_confidence, average_confidence_weighted, avg_frames,
                                  keypoint_calculator.empty_frames_sum)
    # visualisation.single_plot_graph_per_subdirectory()
    #visualisation.plot_combined_graph()  # graph of all somersaults combined aka average plot
    visualisation.dropped_frames_plot()


if __name__ == "__main__":
    main()
```