BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

WEB APPLICATION FOR SETTING UP AN ERGONOMIC POSITION ON A BICYCLE

WEBOVÁ APLIKACE PRO NASTAVENÍ ERGONOMICKÉHO POSEDU NA JÍZDNÍM KOLE

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR **JIŘÍ VLASÁK**
AUTOR PRÁCE

SUPERVISOR **doc. Ing. MARTIN ČADÍK, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2024

Abstract

Do tohoto odstavce bude zapsán výtah (abstrakt) práce v anglickém jazyce.

Abstrakt

Do tohoto odstavce bude zapsán výtah (abstrakt) práce v českém (slovenském) jazyce.

Keywords

Sem budou zapsána jednotlivá klíčová slova v anglickém jazyce, oddělená čárkami.

Klíčová slova

Sem budou zapsána jednotlivá klíčová slova v českém (slovenském) jazyce, oddělená čárkami.

Reference

VLASÁK, Jiří. *Web application for setting up an ergonomic position on a bicycle.* Brno, 2024. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor doc. Ing. Martin Čadík, Ph.D.

Web application for setting up an ergonomic position on a bicycle

Declaration

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana X... Další informace mi poskytli... Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Jiří Vlasák
January 19, 2024

Acknowledgements

V této sekci je možno uvést poděkování vedoucímu práce a těm, kteří poskytli odbornou pomoc (externí zadavatel, konzultant apod.).

Contents

Chapter 1

Introduction

Zadar

Chapter 2

Bikefit

This chapter explains the standard process of bikefitting, the motivation behind it and compares several commonly used software systems used for bikefitting all over the world.

2.1 An Introduction to Bikefitting

Cycling is massively popular activity world-wide. However, having incorrectly set position on the bike can lead to unnecessary pain and injuries. Having a position that does not suit the rider can also have a drastic effect on performance.

Due to these reasons, experts, known as bikefitters help cyclists with the setup of saddle position, handlebar position and sometimes even choosing the right parts such as saddle, handlebars or cranks.

Bikefit sessions are done mostly in person and while most professional cycling teams have a bikefitting specialist that helps to set the bikes for their athletes, they are often too costly for amateur cyclists.

This section draws from Phil Burt's Bike Fit 2nd Edition: Optimise Your Bike Position for High Performance and Injury Avoidance book [5].

2.2 Bikefit Measurements

This section describes the most common measurements used in bikefitting. These measurements can be accessed from a video using automatic bike fitting software and are simple enough to adjust by the rider themselves.

2.2.1 Saddle Height

The saddle height is a fundamental measurement that significantly impacts a rider's comfort and pedaling efficiency. It is argued to be the most important measurement in bikefitting and should be the first measurement to be adjusted [5]. Bad saddle height can even cause problems commonly associted with other measurements such as knee pain, lower back pain, neck pain, saddle sores, etc.

It is determined by considering the rider's knee angle at the bottom of the pedal stroke. Knee angle is the angle between the hip and the ankle measured at the knee joint. Burt recommends a knee angle of 35-40 degrees for average riders and even up to 30 degrees for professional cyclists [5].

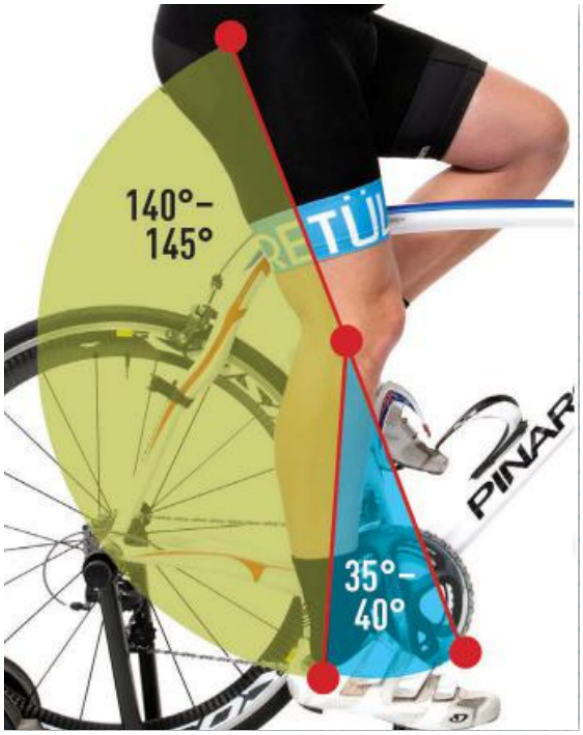

Figure 2.1: Knee extension angle of 140-145 degrees, which is often referred to as 35-40 degrees (being the angle of deviation from straight leg). Is optimal for the average rider. Taken from [5].

Higher saddle height can help to better recruitment of glutens and hamstrings, which can lead to more power output. However, it requires more flexibility and can lead to injury if the rider is not flexible enough. Similarly too low saddle height can increase the compressive forces on the knee and lead to pain and injury.

Proper saddle height is therefore a balance between power output and comfort. It is also important to note that the saddle height is not the only factor that affects the knee angle. The saddle fore and aft position and the cleat position also affect the knee angle.

2.2.2 Saddle Setback

Saddle setback or saddle fore and aft position refers to the horizontal position of the saddle with respect to the bottom bracket.

Setback is most often measured at the 3 o'clock position of the crank. At this position, the rider's knee should be directly above the pedal spindle. Having the knee too far back it is harder to generate power. Having the knee too far forward can lead to knee pain due to increased forces on the kneecap [5].

Saddle setback also affects the rider's balance on the bike. Having the saddle too far forward can lead to the rider's weight being too far forward, which can lead to hand pain because of too much weight on the handlebars. Having the saddle too far back can lead to the rider's weight being too far back, which can lead to make the front wheel feel light and make the bike harder to control.

Setback also affects the rider's hip angle. Hip angle is the angle between the shoulder and the knee measured at the hip. Having the saddle more forward can lead to a more open

hip angle, which can lead to more power output and more space between the rider's torso and legs at the top of the pedal stroke. This is why time trial bikes have a more forward saddle position.

2.2.3 Handlebar Height and Reach

Handlebar height measures how high the handlebars are in relation to the saddle. Handlebar reach measures how far the handlebars are in relation to the saddle.

Handlebar height influences mainly torso angle and to and also shoulder angle. Torso angle is the angle between the shoulder and the level plane measured at the hip. It is also known as the back angle. Shoulder angle is the angle between the hip and wrist measured at the shoulder. Handlebar reach influences mainly shoulder angle.

Handlebar height can be adjusted by changing the number of spacers under the stem or by changing the stem itself. Handlebar reach can be adjusted by changing the stem length or by changing the handlebars themselves.

While optimal handlebar height and reach are highly individual, there are some general guidelines. For example, a more upright position is more comfortable and is therefore recommended for longer rides. A more aggressive position is more aerodynamic and is therefore recommended for racing. Burt recommends back angle of about 45 degrees for average riders and shoulder angle of about 90 degrees with the elbow slightly bent [5]. For faster riders, the back angle can be lowered up to 30 degrees with more open shoulder angle. For more upright riders, the back angle can be increased up to 55 degrees with more closed shoulder angle.

2.3 Existing Solutions for Automated Bikefitting

2.3.1 MyVeloFit

MyVeloFit is a web application that uses pose estimation model to predict the joint locations for a side view video of the user pedaling their bike on an indoor trainer. Based on location of these joints, joint angles are then computed. On the basis of these angles and their relation to average angles, suggestions are made for adjusting the position of the saddle and handlebars.

The fitting process starts with the rider filling out questionaire about their mobility. This is then used to adjust the recommended angle ranges. For example: if the user has lower shoulder mobility, recommended ranges for shoulder angle will be increased so the user is not stretched forward so much.

After creating the user profile, user can create a fit session for one specific bike. In the process, the user selects their fit goal (performance, comfort, or balanced) and the type of bike they are using (road, gravel, mtb, triathlon, hybrid, or stationary). This also changes the recommended angle ranges.

Predicted keypoints

MyVeloFit predicts 6 joint locations for the camera facing side of the body. Most common pose estimation models predict similar keypoints. However, keypoints commonly used to adjust the position of the saddle, such as the heel and the fifth metatarsal of the foot, are missing.

- Ankle
- Knee
- Hip
- Shoulder
- Elbow
- Wrist

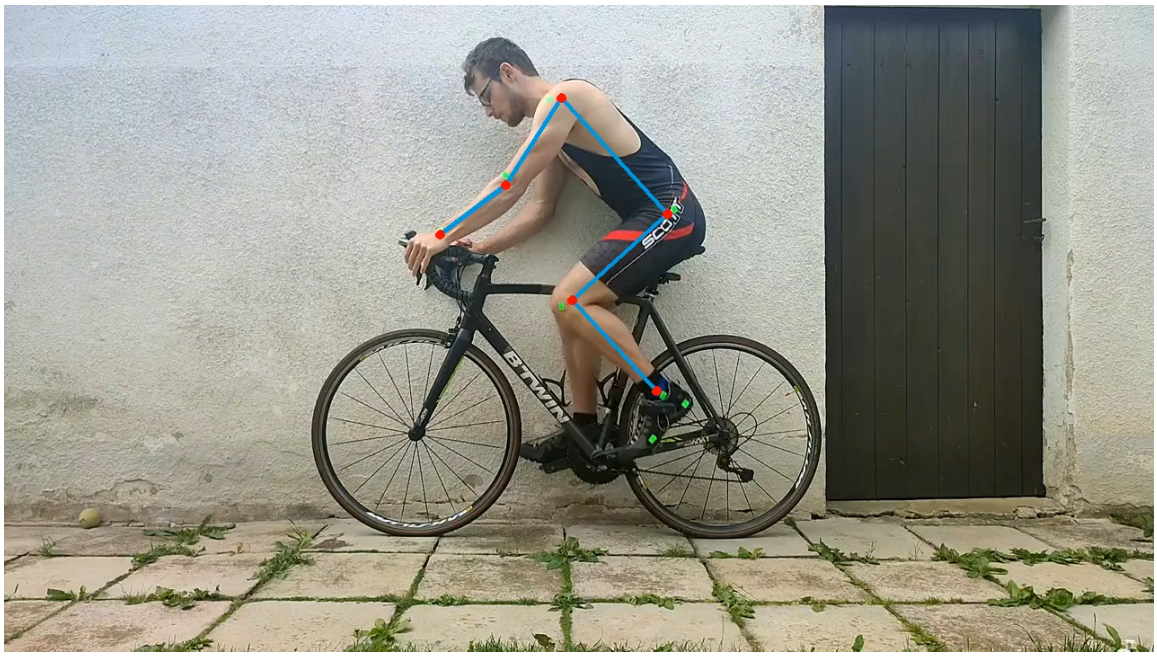

Figure 2.2: Side view image with predicted keypoints in MyVeloFit.

From the joint angles for every frame, some are selected for computing the joint angles at the top of the pedal stroke, front of the pedal stroke and bottom. Every position uses different angle ranges and even which angles are taken into account.

Based on the angles computed for parts of the pedal stroke, MyVeloFit then makes suggestions for saddle height, saddle fore and aft, handlebar height and handlebar reach.

Overall, MyVeloFit is relatively easy to use and its joint predictions are fairly accurate. However, it has few disadvantages:

- Only the most basic keypoints are used.
- Every video is converted to 30 FPS and cut down to 10 seconds.
- Video processing and keypoint predictions are slow (3-5 minutes).
- Requires subscription to get joint angles and recommended changes. Either a one time payment of 35 US dollars for access for 1 person and 1 bike for 2 weeks or 75 US dollars annually for unlimited number of bikes and people.

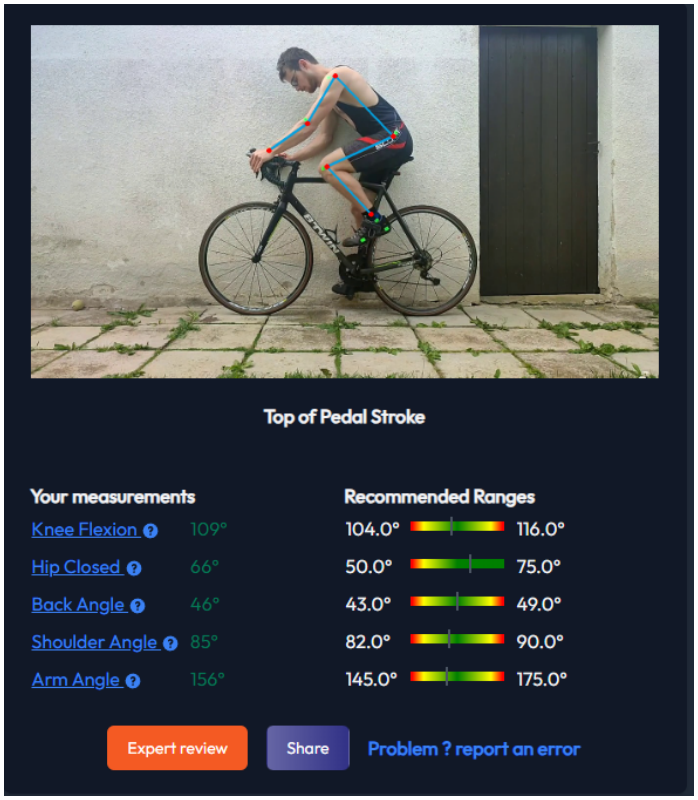

Figure 2.3: Predicted joint angles at the top of the pedal stroke in MyVeloFit.

2.3.2 Retül

Retül is a bike fitting system employing 3D motion capture technologys. It utilizes infrared LED markers placed on specific body points to track the rider's movements dynamically while cycling. The led markers are tracked by multiple infrared cameras placed around the rider. The cameras surprisingly capture only 18 frames per second. Despite this research [23] shows that the system is relatively reliable compared to 3d motion capture systems with higher frame rates.

Retül uses 8 markers placed on both sides of the rider's body. These markers are placed on the following locations:

- Fifth metatarsal of the foot
- Heel
- Ankle
- Knee
- Hip
- Shoulder
- Elbow
- Wrist

Figure 2.4: Recommended changes to the bike position based on the angles computed by MyVeloFit.

The markers are placed by the fitter on the rider's body. Accurate placement of the markers is crucial for the system to work properly. Even small deviations can lead to inaccurate results.

Figure 2.5: Placement of the markers used by Retül. (Screenshot from instructional instructional video by Retül [22].)

Retül's fitting process involves setting up the bike on a trainer equipped with the system. During the session, the rider performs various motions and pedal strokes while the Retül system captures real-time data on joint angles and movements.

Data Captured by Retül includes a wide range of joint angles and movements such as knee angles at top of the pedal stroke and bottom of the pedal stroke, hip angles throughout the pedal stroke, shoulder, elbow, and wrist positions in relation to handlebar reach and drop, as well as ankle and foot movement concerning cleat positioning and alignment.

The normal ranges for these angles were constructed based on the data collected from thousands cyclists. However, these cyclists were not necessarily optimally fitted to their bikes. Therefore, the normal ranges may not be based on the optimal position for the rider.

Based on the captured data, Retül compares the rider's position to the normal ranges. Based on this comparison, the fitter can make changes to the bike position.

Despite the fact that Retül is a very popular bike fitting system, it has some important disadvantages:

- Costly equipment and setup requirements, limiting accessibility to some individuals or smaller bike shops.
- The need for trained Retül bike fitters to interpret and implement fitting recommendations effectively.
- Requires in-person fitting sessions. These sessions can be time-consuming and costly.

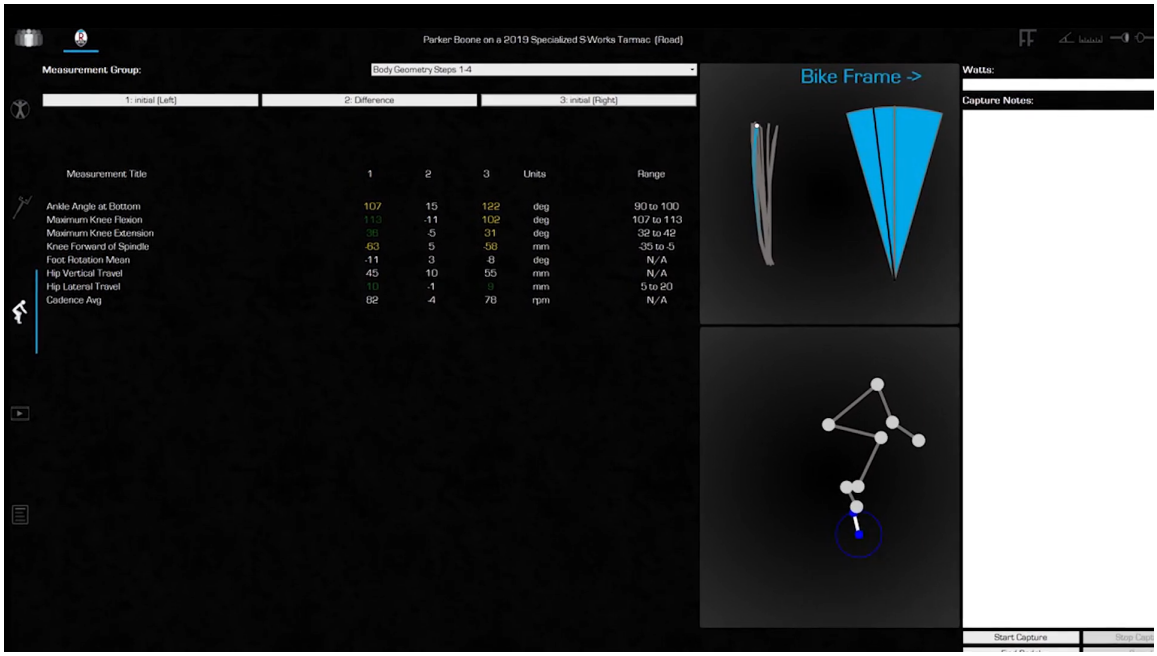

Figure 2.6: Retül's software showing the captured data. (Screenshot from instructional video by Retül [21].)

2.3.3 BikeFast Fit Elite

BikeFast Fit Elite is an iOS and Mac OS application that uses pose estimation model to predict the joint locations for a side view video of the user pedaling their bike on an indoor trainer. Compared to MyVeloFit, it uses additional keypoints for the fifth metatarsal of the foot and the heel.

Similarly to MyVeloFit, it suggest changes to the saddle height and fore and aft position but it does not suggest changes to the handlebar position, arguing that the handlebar position is based on individual goals and flexibility.

Additionally, it also provides front view knee tracking to address possible knee wobble and asymmetry.

The app costs 19.99 US dollars and does not require a subscription. However, it is only available for iOS and Mac OS. Also it only captures 3.5 seconds of video.

2.3.4 Kinovea

2.3.5 Posiclist

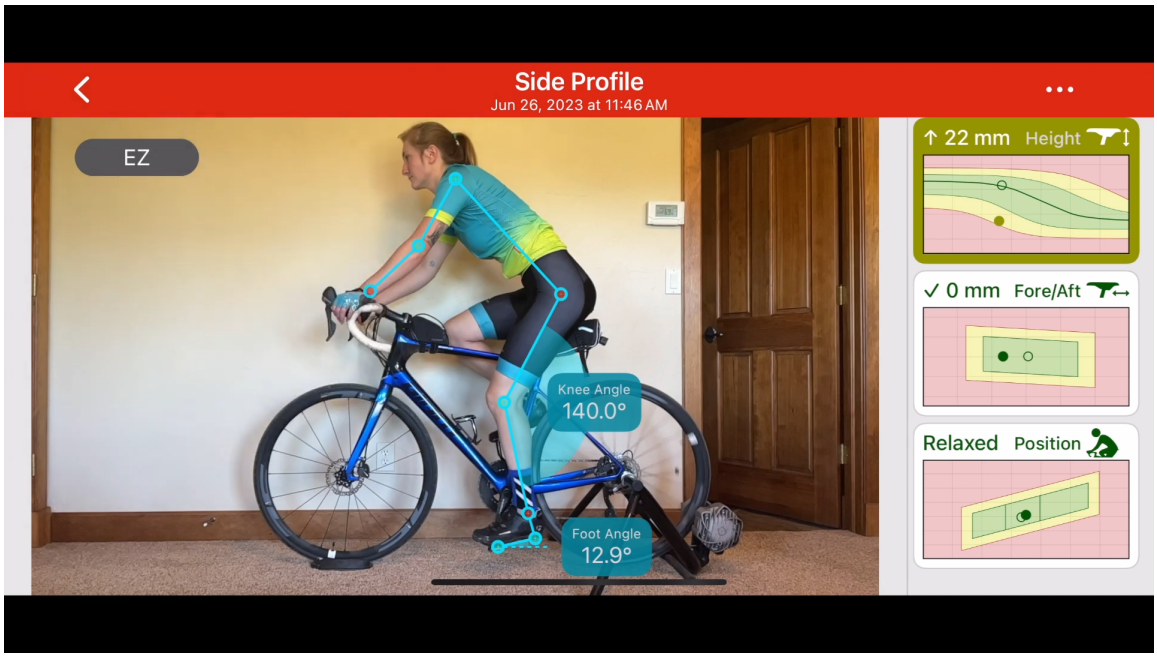

Figure 2.7: Side view image with predicted keypoints in BikeFast Fit Elite. (Screenshot from promotional video [4] by BikeFast Fit Elite.)

Chapter 3

Pose Estimation Models

This chapter describes some of the good performing pose estimation algorithms and compares their performance on the task of side view pose estimation of cyclists.

Pose estimation is a computer vision task that involves predicting the locations of keypoints on a person in an image or a video. The keypoints are usually the joints of the person, such as the ankle, knee, hip, shoulder, elbow, wrist, etc. The pose estimation models are usually trained on datasets that contain images or videos with annotated keypoints. The models are then evaluated on the same datasets.

There are multiple approaches to pose estimation. The most common approaches are the top-down approach and the bottom-up approach. The top-down approach first detects people in the image and then predicts the keypoints for each person. The bottom-up approach first predicts the keypoints and then groups them into poses using hand writte post processing algorithms. Another approach is the one-stage approach, which predicts the keypoints directly from the image without the need for post processing. This work evaluates three pose estimation models using the top-down approach (RTMPose, HRNet and ViPNAS), described in sections 3.1, 3.2 and 3.4 and one pose estimation model using the one-stage approach (YOLOX-Pose), described in section 3.3.

Not all of the models predict the same keypoints. The models predict either 17 keypoints, as defined in the COCO dataset [18], 26 keypoints, as defined in the Halpe26 dataset [8], or 133 keypoints, as defined in the COCO-WholeBody dataset [14].

3.1 RTMPose

RTMPose [13] is a pose estimation model from the authors of the MMPose framework [7]. It is designed to bridge the gap between the excellent performance of 2D pose estimation on public benchmarks and its application in the industrial community, which still suffers from heavy models and high latency.

The RTMPose models are designed to be lightweight and fast. The authors claim, that the RTMPose-m achieves 75.8% AP on COCO with 90+ FPS on an Intel i7-11700 CPU and 430+ FPS on an NVIDIA GTX 1660 Ti GPU. The RTMPose-s model, achieving 72.2% AP on COCO was also tested on a mobile device with the Snapdragon 865 chip, running at 70+ FPS.

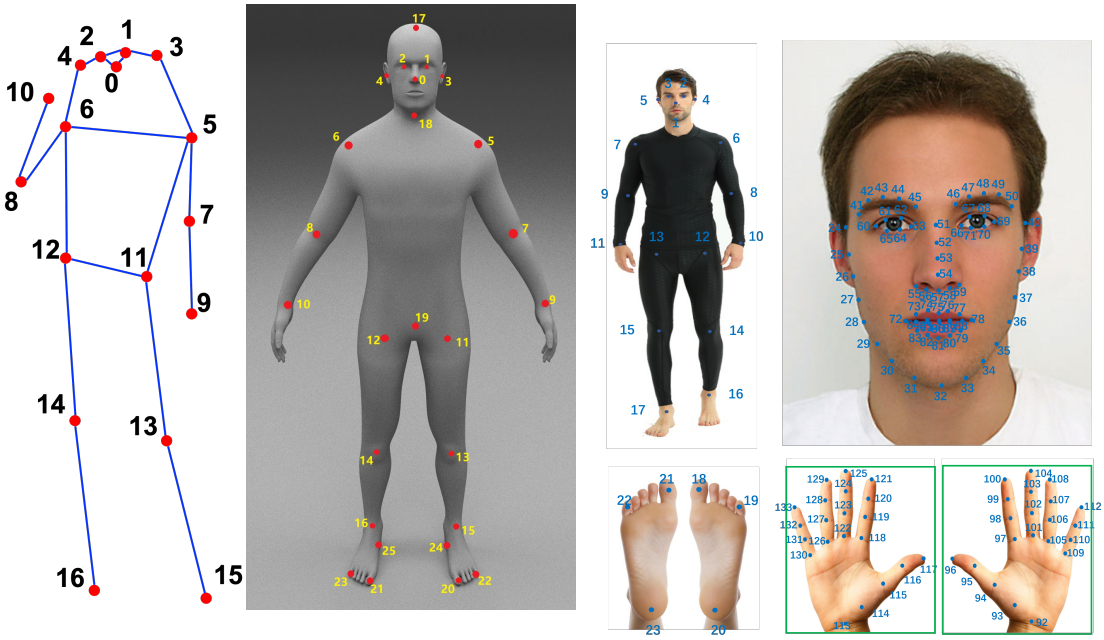

Figure 3.1: Sets of keypoints used by the MS-COCO dataset (image taken from [18]), the Halpe26 dataset (image taken from [8]) and the COCO-WholeBody dataset (image taken from [8]).

3.1.1 Architecture

The RTMPose models are based on the top-down pose estimation approach. They use a two-stage prediction, where the first stage is a person detector and the second stage is a pose estimator. The person detector is used to crop the image to the bounding box of the person. The pose estimator then predicts the keypoints for the cropped image. The authors claim that this approach is more accurate than the bottom-up approach, while still being faster in cases where the number of people in the image is lower than 6.

The CSPNeXt backbone is used in the RTMPose models. This backbone is primarly designed for object detection. Authors claim that backbones designed for image classification, are not optimal for dense tasks such as pose estimation, object detection, semantic segmentation, etc. Some backbones using high-resolution feature maps or advanced transformer architectures achieve good results, but suffer from high computional cost, high latency or difficulties in deployment. The CSPNeXt backbone is designed to have a good balance of speed and accuracy.

For prediction of the keypoints, the RTMPose models utilize the SimCC [16] algorithm. SimCC reformulates human pose estimation as two classification tasks for horizontal and vertical coordinates. To reduce quantization error, SimCC can use a larger number of bins for the classification tasks. This can lead to better accuracy, while still being faster than post-processing methods commonly used with tradional heatmap based pose estimation models.

To better use spatial information, the RTMPose models use a Gated Attention Unit (GAU) [11] module. The GAU has faster speed, lower memory cost and better accuracy than the commonly used self-attention module, proposed in the Transformer architecture [27].

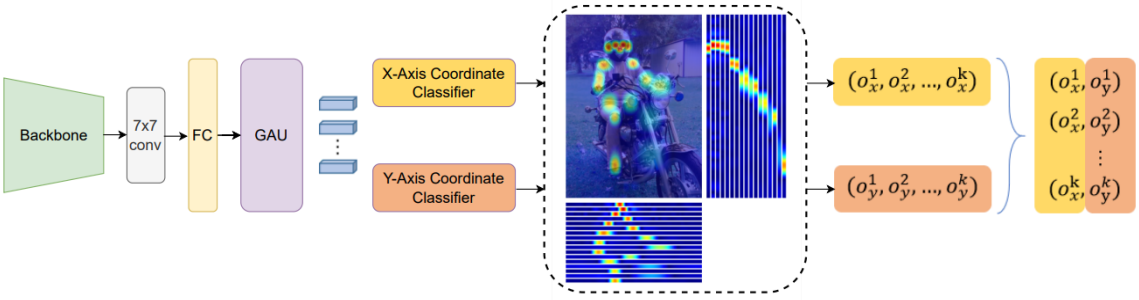

Figure 3.2: Architecture of the RTMPose models, which comprises of a CSPNeXt backbone, a convolutional layer, a fully-connected layer, and a Gated Attention Unit (GAU) designed to enhance K keypoint representations. Subsequently, the process of 2D pose estimation is treated as two separate classification tasks for the x-axis and y-axis coordinates. This involves predicting the horizontal and vertical positions of keypoints. Taken from [13].

3.1.2 Model Versions

This work evaluates the following RTMPose models:

RTMPose Body8 models: These models are trained and evaluated on the Body8 dataset consisting of 8 pose estimation datasets (AI Challenger [28], MS-COCO [18], CrowdPose [15], MPII [2], sub-JHMDB [12], Halpe [8], PoseTrack18 [1] and OCHuman [30]). They predict 17 keypoints, as defined in the COCO dataset.

RTMPose Halpe26 models: These models are also trained and evaluated on the Body8 dataset. However, they predict 26 keypoints, as defined in the Halpe26 dataset.

RTMPose WholeBody models: These models are trained and evaluated on the COCO-WholeBody [14] and UBody [17] datasets. They predict 133 keypoints, as defined in the COCO-WholeBody dataset.

3.2 HRNet

The HRNet (High-Resolution Network) [24] pose estimation architecture is characterized by its emphasis on maintaining high-resolution representations throughout the network. Unlike conventional methods that downsample the input image early in the network, HRNet adopts a multi-resolution approach, preserving detailed information essential for accurate pose estimation.

A notable feature of the HRNet pose estimation architecture is its ability to capture both local and global context effectively. The network consists of parallel branches, each processing different resolutions of the input image, and these branches are interconnected, facilitating the exchange of information between different scales. This design allows HRNet to address challenges associated with scale variations in human poses, ensuring robust performance in capturing both fine details and overall pose structure.

This work evaluates the HRNet-W32 model, which uses heatmap to predict the keypoints. The HRNet-W32 model is trained on the COCO dataset and predicts 17 keypoints, as defined in the COCO dataset.

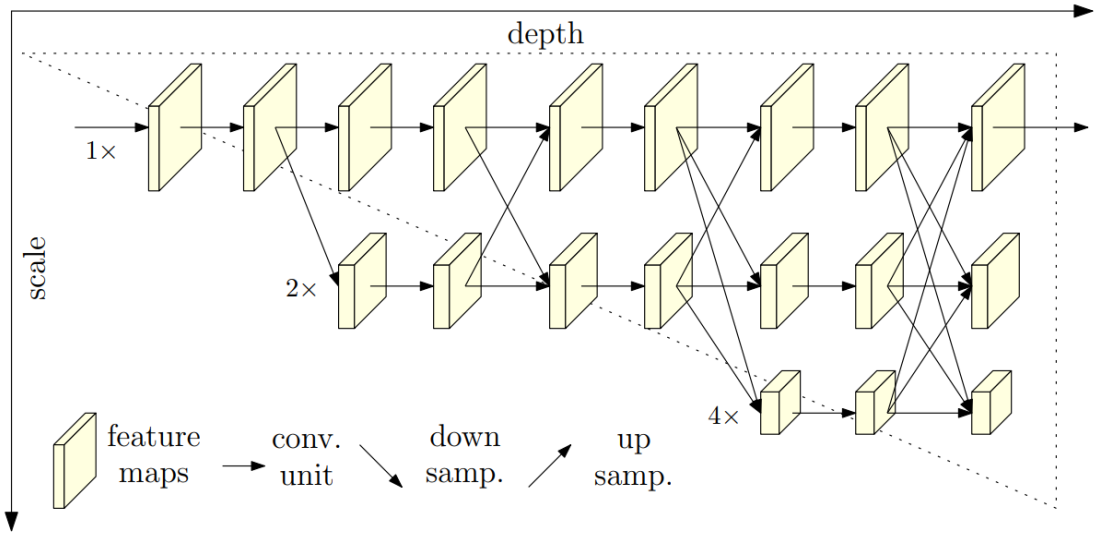

Figure 3.3: Architecture of the HRNet models. The HRNet models consists of three parallel branches, each processing different resolutions of the input image. These branches are interconnected, facilitating the exchange of information between different scales. Taken from [24].

3.3 YOLOX-Pose

YOLOX-Pose is a one-stage human pose estimation model that distinguishes itself from traditional top-down and bottom-up methods. The authors claim that YOLOX-Pose is the first one-stage human pose estimation model that achieves comparable performance to the state-of-the-art two-stage methods.

The one-stage approach has some advantages over bottom-up and top-down approaches. Bottom-up approaches first detect keypoints and then group them into poses. The grouping step is not learned and needs to be hand-crafted. This can lead to inaccurate results, especially in cases where the keypoints from different people are close together.

Top-down approaches first detect people and then predict keypoints for each person. This approach is favored by SOTA models, but the processing time grows linearly with the number of people in the image. This can lead to slow processing times in cases where there are many people in the image. SOTA methods also often use heavy detectors, which further increases the processing time.

The original YOLO-pose [20] model uses the anchor-based YOLOv5 detector [26]. The YOLOX-Pose model, which is evaluated in this work, uses the YOLOX detector [9]. The YOLOX detector is based on the YOLOv5 detector, but uses anchor-free approach, which does not require the anchor boxes to be predefined. This allows the YOLOX detector to be more flexible. The YOLOX detector generally outperforms the YOLOv5 detector on the COCO dataset.

The YOLOX-Pose model is trained on the COCO dataset and predicts 17 keypoints, as defined in the COCO dataset.

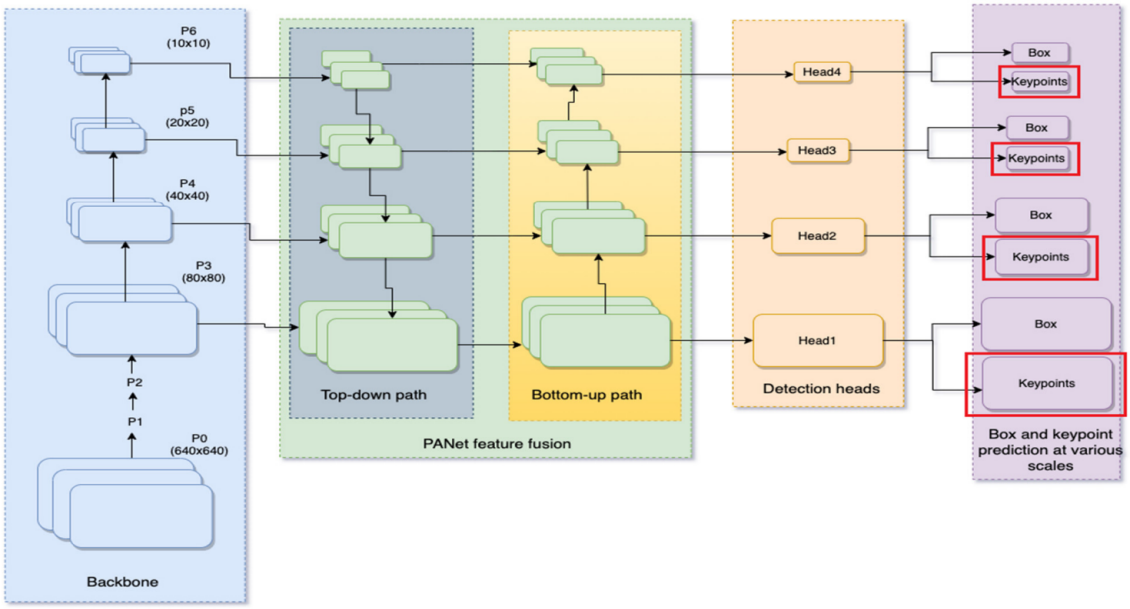

Figure 3.4: Architecture of the original YOLO-Pose model, which was based on the anchor-based YOLOv5 detector. The YOLOX-Pose model, which is evaluated in this work, is based on the anchorless YOLOX detector. Taken from [20].

3.4 ViPNAS

The ViPNAS (Video Pose Estimation via Neural Architecture Search) [29] model focuses on video pose estimation, employing two networks: S-ViPNet and T-ViPNet.

The S-ViPNet is used on invidual frames and keyframes in the video to predict the keypoints. The lightweight T-ViPNet network is used on non-keyframes to propagate the pose estimations. It extracts features from the current frame and also fuses heatmaps from the previous frame. The fused features are then used to predict the keypoints.

To find the optimal architectures for the S-ViPNet and T-ViPNet networks, the authors used neural architecture search (NAS). The search space for the S-ViPNet network is called the spatial-level search space. It consists of 5 dimensions (depth, width, kernel size, group and attention). The search space for the T-ViPNet network is called the temporal-level search space. It searches for the optimal fusion operation (addition, multiplication or concatenation) and the best stage of features to fuse.

This work evaluates the modified S-ViPNet model with the MobileNetV3 [10] backbone and SimCC [16] algorithm instead of predicting heatmaps. The model is trained on the COCO dataset and predicts 17 keypoints, as defined in the COCO dataset.

3.5 Results on COCO

To compare the potential of the methods, the results of the models on the COCO dataset are shown in table 3.1. All of the methods were trained only on the COCO dataset and predict the same keypoints. The results are taken from MMPose documentation [7].

The RTMPose models achieve strong results, with the RTMPose-l-256x192 model achieving the best results. The HRNet-W32 model also performs well but it is a much larger and

slower model. However, the other lightweight models such as YOLOX-Pose and ViPNAS-MobileNetV3 perform significantly worse than even the medium and small RTMPose models. Overall the RTMPose models show very promising results compared to the other models.

Model	Input Size	AP	AP-50	AP-75	AR	AR-50
rtmpose-l	**256x192**	**0.758**	**0.906**	**0.826**	**0.806**	0.942
pose_hrnet_w32	256x192	0.749	0.906	0.821	0.804	**0.945**
rtmpose-m	256x192	0.746	0.899	0.817	0.795	0.935
rtmpose-s	256x192	0.716	0.892	0.789	0.768	0.929
simcc_S-ViPNAS-MobileNetV3	256x192	0.695	0.883	0.772	0.755	0.927
yoloxpose_m	640x640	0.695	0.899	0.766	0.733	0.926
rtmpose-t	256x192	0.682	0.883	0.759	0.736	0.920

Table 3.1: Results of the models on the COCO dataset. The results are sorted by the Average precision (AP). The AP describes the average precision over Object Keypoint Similarity (OKS) thresholds from 0.5 to 0.95 with a step size of 0.05. The AR describes the average recall over OKS thresholds from 0.5 to 0.95 with a step size of 0.05. The AP-50 and AR-50 are the same as AP and AR, but only for OKS threshold of 0.5. Similarly AR-75 is average recall with OKS threshold of 0.75. The results are taken from MMPose documentation [7].

Chapter 4

Evaluation on Pose Estimation of Cyclists

The goal of this section is to evaluate the performance of pose estimation models on the task of side view pose estimation of cyclists and find the best model for use in the bikefit application.

The models are evaluated on a custom dataset, which is described in section 4.1. The evaluation metrics are described in section 4.2. The results are shown in section 4.3.

The models are ran using the MMPose framework [7], with the Inferencer API and the PyTorch backend. Topdown models (models that require inputs cropped to the bounding box of the person) use the RTMDet-nano model [19] from the MMDetetection framework [6] for person detection.

Since no models predict the fifth metatarsal of the foot, the small toe landmark is used instead for the models that predict it.

4.1 Evaluation Dataset

To accurately measure the performance of a pose estimation model, it is necessary to have a dataset with ground truth annotations. Since there is no such dataset for side view pose estimation of cyclists, new small evaluation dataset was created. It consists of 12 videos in 3 different enviroments with various camera angles and riding positions (riding on hoods, drops, tops and aero position on the drops). The videos were shot in FullHD resolution at 30 FPS.

Before shooting the videos, orange colored markers were placed on the camera-facing side of the rider's body. These markers were placed on the following locations:

- Fifth metatarsal of the foot
- Heel
- Ankle
- Knee
- Hip
- Shoulder

- Elbow
- Wrist

To annotate the videos, the markers were tracked using the first version of the bikefit application described in TODO. The application tracks the markers in a lower resolution. To get the ground truth annotations, the marker positions were then scaled to the original resolution of the video.

4.2 Evaluation Metrics

In assessing the performance of pose estimation models, widely used metrics, such as Mean Average Precision (mAP), or Object Keypoint Similarity (OKS) may not adequately gauge accuracy in scenarios involving video data where all keypoints of interest are consistently visible and accurately predicted. Similarly, relying solely on the L2 distance for evaluation proves inadequate due to the inherent variations in scale among different videos.

As an alternative approach, a novel metric called **Bounding Box Normalized Distance (BBND)** has been employed. This metric involves calculating the ratio of the Euclidean distance to the average dimension of the bounding box, scaled by a factor of 100, so it can be expressed as a percentage:

$$\text{BBND} = \frac{\text{dist}}{\frac{\text{bbox.w}+\text{bbox.h}}{2}} \times 100$$

Here, in the BBND formula:

BBND: Bounding Box Normalized Distance, which is the evaluation metric being proposed.

dist: Euclidean distance between the predicted keypoints and the ground truth keypoints.

bbox.w: Width of the bounding box encompassing the keypoints.

bbox.h: Height of the bounding box encompassing the keypoints.

This adjustment allows for a scaled representation of the distance metric, effectively normalizing it with respect to the average bounding box dimension. BBND offers easy to interpret results, with a lower value indicating a more accurate prediction. It also allows for a more direct comparison of the performance of different models, as the metric is not affected by the scale of the video.

4.3 Results

At first the models are evaluated using only the keypoints that all models predict. These keypoints are the ankle, knee, hip, shoulder, elbow and wrist. The results are shown in table 4.1. The best results are achieved by the rtmpose-l-256x192 model, but most of the bigger RTMPose models outperform the other models. This is in contrast to the results on the COCO dataset, where HRNet-W32 model performed similarly to the RTMPose-l model. The decrease in performance of the HRNet-W32 model is probably caused by the fact that it was trained only on the COCO dataset, while the RTMPose models were trained on multiple datasets and are therefore better used to pose estimation for cycling.

Figure 4.1: Examples of the evaluation dataset. The first video shows the rider pedaling indoors on the hoods. The second video shows the rider pedaling outdoors in the aero position on the hoods. The orange stickers are used as ground truth annotations.

The best model achieves a mean BBND of 2.59%. This means that the average distance between the predicted keypoints and the ground truth keypoints is 2.59% of the average bounding box dimension.

The good results of the RTMPose models are not surprising, since they were trained on multiple datasets, while the other models were trained only on the COCO dataset. The RTMPose models trained to predict the COCO keypoints perform slightly better than the halpe26 models, which were trained to predict 26 keypoints and significantly better than the wholebody models, which were trained to predict 133 keypoints.

Model	**Ankle**	**Knee**	**Hip**	**Shoulder**	**Elbow**	**Wrist**	**Mean**
rtmpose-l_body8-256x192	3.43	3.79	**2.15**	3.39	**1.21**	1.58	**2.59**
rtmpose-l_body8-384x288	3.26	**3.53**	3.33	**2.93**	1.44	**1.43**	2.65
rtmpose-l-halpe26-256x192	3.31	3.75	2.74	3.42	1.27	1.85	2.72
rtmpose-m_body8-384x288	3.40	3.84	3.27	3.26	1.34	1.59	2.78
rtmpose-m-halpe26-384x288	3.76	3.68	3.96	3.02	1.34	1.65	2.90
rtmpose-m-halpe26-256x192	3.56	3.92	2.97	3.34	1.71	2.11	2.94
rtmpose-m_body8-256x192	4.10	4.19	2.84	3.50	1.58	2.08	3.05
rtmpose-s_body8-256x192	5.19	4.40	2.83	3.93	1.95	2.20	3.42
rtmpose-s-halpe26-256x192	4.74	4.38	2.89	3.94	2.30	2.47	3.45
td-hm_hrnet-w32_coco-256x192	**3.17**	4.46	4.31	4.17	2.37	2.30	3.46
yoloxpose_m_coco-640	3.77	5.68	3.66	4.20	2.06	2.55	3.65
rtmpose-t-halpe26-256x192	5.09	4.96	3.19	4.86	2.95	2.65	3.95
rtmpose-l-wholebody-256x192	3.65	5.16	5.01	4.52	3.41	2.81	4.09
rtmpose-t_body8-256x192	6.17	4.98	3.16	4.70	2.99	2.75	4.13
simcc_vipnas-mbv3_coco-256x192	6.39	6.00	4.53	4.36	3.01	2.86	4.53
rtmpose-m-wholebody-256x192	5.57	6.16	4.94	4.78	3.62	3.44	4.75

Table 4.1: Evaluation results on the custom bike fitting dataset, using only MS-COCO keypoints (excluding heel and fifth metatarsal of the foot). Each column represents the mean Bounding Box Normalized Distance for the given keypoint. The results are sorted by the mean Bounding Box Normalized Distance, defined in section 4.2. The best results are highlighted in bold. The model names are derived from their MMPose config files.

The models which also output heel and top of the foot keypoints are evaluated in table 4.2. These models have a slighty higher mean BBND than the models that only output the MS-COCO keypoints. This is probably because there are more datasets with MS-COCO keypoints than datasets with the heel and foot keypoints. The best model is rtmpose-l_halpe26-256x192 with a mean BBND of 3.33%. However, we can see that the heel and especially the fifth metatarsal of the foot are not predicted very well. This is probably because the heel and the fifth metatarsal of the foot are not visible in most of the training datasets. Another reason could be that the models do not output position of the fifth metatarsal of the foot, but only the position of the small toe and these two keypoints are not always in the same position.

Model	Foot	Heel	Overall Mean
rtmpose-l_halpe26-256x192	**5.96**	4.34	**3.33**
rtmpose-m_halpe26-384x288	6.36	**4.22**	3.50
rtmpose-m_halpe26-256x192	5.97	5.34	3.62
rtmpose-s_halpe26-256x192	6.01	6.23	4.12
rtmpose-t_halpe26-256x192	6.78	6.45	4.62
rtmpose-l_wholebody-256x192	6.49	6.23	4.66
rtmpose-m_wholebody-256x192	8.13	8.12	5.59

Table 4.2: Evaluation results on the custom bike fitting dataset, including the heel and fifth metatarsal of the foot. Results for the other keypoints can be found in table 3.1. Each column represents the mean Bounding Box Normalized Distance for the given keypoint. The results are sorted by the overall mean Bounding Box Normalized Distance (spanning all keypoints), defined in section 4.2. The best results are highlighted in bold. The model names are derived from their MMPose config files.

Chapter 5

Training Pose Estimation Model for Bikefitting

5.1 Dataset

5.2 Training Setup

5.3 Model Compression

5.3.1 Pruning

5.3.2 Quantization

5.4 Results

Chapter 6

Architechture and Implementation of the Bikefit Application

This chapter describes the web application for bike fitting. Implementing the application as a web application has several advantages over implementing it as a mobile or desktop application. The web application can be used on any device with a web browser. This allows the application to be used on mobile devices, tablets, laptops and desktop computers. The user also does not need to download and install anything.

Most web applications are implemented using the client-server architecture with the video processing and other more complicated tasks (such as pose estimation) being performed on the server. This has several disadvantages. The server needs to be powerful enough to handle the load of multiple users. This can be quite expensive for heavy models and a lot of users. The user data also needs to be sent to the server, which means the user needs to be online when using the application and the users can be concerned about their privacy.

To avoid these disadvantages, most of the bussiness logic of the application (including video processing, keypoint estimation and generation of the suggestions) is implemented on the client side.

The application is implemented using the Svelte framework[1] . Svelte is a component framework similar to React[2] or Vue[3]. It is used to create reactive user interfaces. Svelte compiles the application to vanilla JavaScript, which means the application does not need to include a large framework library. This makes the application smaller and faster. Svelte also has a built-in state management system, which is used to store the application state.

To process the video frame by frame, the application uses the WebCodecs API. The details of the implementation are described in section 6.1.

To estimate the keypoints, multiple approaches were tested. First approach uses colored markers, which are placed on the parts of the body, whose position needs to be estimated (foot, heel, ankle, knee, hip, shoulder, elbow, wrist). The markers are then detected using color thresholding and tracked using a method similar to SORT. This approach is implemented using the OpenCV.js library [25]. The details of the implementation are described in section 6.2.

[1] https://svelte.dev/
[2] https://react.dev/
[3] https://vuejs.org/

The second approach uses the RTMPose pose estimation model and a person detector. The models run in the browser using the Tensorflow.js library[4]. The models are described in section 3.1 and evaluated in chapter 4. To improve the performance of the pose estimation models on the task of side view pose estimation of cyclists, the models are trained on a custom dataset. The details of the training are described in section 5. The details of the implementation are described in section 6.3.

6.1 Video Upload and Processing

At first, the video needs to be uploaded to the application. The video is uploaded using the HTML5 file input. The file is then converted to a URL using the `URL.createObjectURL()` method, because the WebCodecs API requires a URL as input.

6.1.1 Accessing the Video Frames

To accurately estimate the keypoints, the application needs to access all of the video frames and process them. This is not trivial to do in the browser. The browser does not allow the application to access the video frames directly.

To process the video frame by frame, the application uses the WebCodecs API[5]. The WebCodecs API is a low-level API for encoding and decoding audio and video. It gives developers access to the invididual frames of the video. It is the only way to access the frames of the video in the browser. Simple and commonly used method is to simultaneously play the video and draw the video frames to a canvas element and then access the pixels of the canvas. However, this method only does not work well if the processing is computationally expensive, since the video is played in real time. This leads to skipping of the frames and not all of the frames are processed. The WebCodecs API allows the application to process the video at its own pace.

Because the API is relatively new, it is not yet supported by all browsers. Figure 6.1 shows the browser support for the WebCodecs API as of January 2024. The API is supported by all browsers based on Chromium (Google Chrome, Microsoft Edge, Opera, Brave, etc.). It is Partially supported by Safari and Safari on iOS (video only) and not supported by Firefox. Overall based on CanIUse data[6], the API is supported by 87.6% of the users, where 74.5% of the users have full support and 13.1% of the users have partial support.

Due to the low-level design of the WebCodecs API, it is not easy to use. Moreover, the API is quite new and not very well documented so there is not a lot of examples. The developer also needs to take care of demuxing the media containers. This work uses the getVideoFrames.js library[7] from josephrocca. The library internally uses the WebCodecs API and MP4Box.js[8] for demuxing mp4 files, but provides a simple interface for accessing the video frames. The library is used to get the video frames and then the frames are processed as described in sections 6.2 and 6.3. Listing 6.1 shows an example usage of the getVideoFrames.js library. It defines three callback functions. The `onFrame` callback is

[4] https://www.tensorflow.org/js
[5] https://developer.mozilla.org/en-US/docs/Web/API/WebCodecs_API
[6] https://caniuse.com/webcodecs
[7] https://github.com/josephrocca/getVideoFrames.js
[8] https://github.com/gpac/mp4box.js/

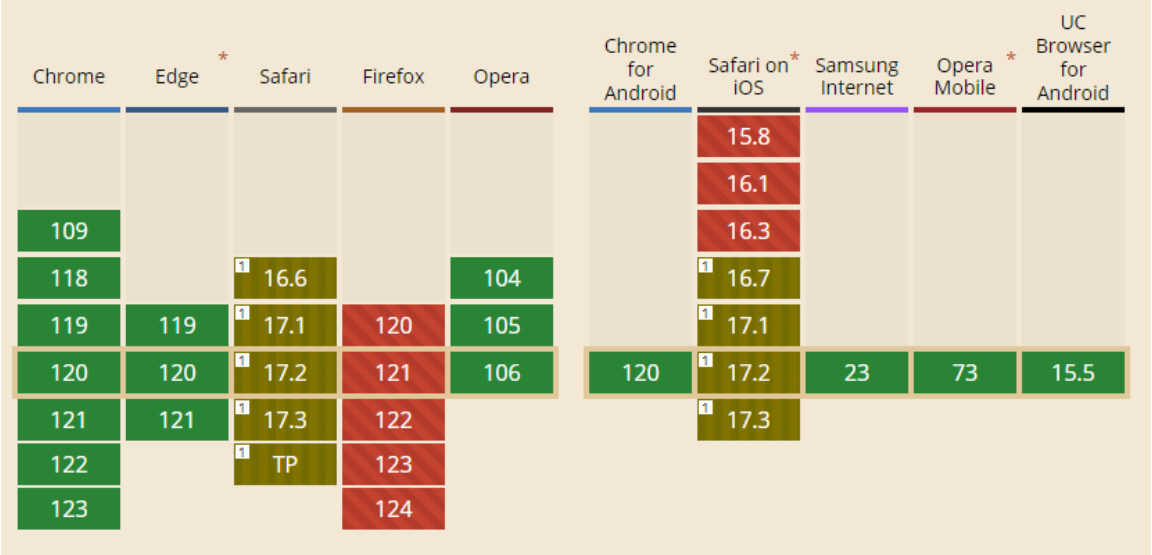

Figure 6.1: Browser support as of January 2024 for the WebCodecs API. Taken from https://caniuse.com/webcodecs.

called for each frame of the video. The `onConfig` callback is called when the video metadata is loaded. The `onFinish` callback is called when all of the frames are processed.

```
<canvas id="canvasEl"></canvas>
<br>
<input type="file" accept="video/mp4" onchange="start(this.files[0])">
<script type="module">
  import getVideoFrames from "https://deno.land/x/get_video_frames@v0.0.10/mod.js"

  let frameCount = 0;

  window.start = async function(file) {
    let ctx = canvasEl.getContext("2d");

    // `getVideoFrames` requires a video URL as input.
    // If you have a file/blob instead of a videoUrl, turn it into a URL like this:
    let videoUrl = URL.createObjectURL(file);

    await getVideoFrames({
      videoUrl,
      onFrame(frame) { // `frame` is a VideoFrame object: https://
          developer.mozilla.org/en-US/docs/Web/API/VideoFrame
        ctx.drawImage(frame, 0, 0, canvasEl.width, canvasEl.height);
        frame.close();
        frameCount++;
      },
      onConfig(config) {
        canvasEl.width = config.codedWidth;
        canvasEl.height = config.codedHeight;
      },
      onFinish() {
        console.log("finished!");
        console.log("frameCount", frameCount);
      },
    });
```

```
32
33          URL.revokeObjectURL(file); // revoke URL to prevent memory leak
34        }
35      </script>
```

Listing 6.1: Example usage of the getVideoFrames.js library. Taken from https://github.com/josephrocca/getVideoFrames.js/.

6.2 Marker-based Tracking

To estimate the keypoints, the application uses colored markers, which are placed on the parts of the body, whose position needs to be estimated (foot, heel, ankle, knee, hip, shoulder, elbow, wrist). The markers are at first localized manually by the user and their color is then used to detect them in the video. The markers are then tracked using a method similar to SORT. The detection and tracking is implemented using the OpenCV.js library [25].

6.2.1 Manual Localization of the Markers

At first, the user is asked to manually localize the markers by clicking on them in the first frame of the video. Figure 6.2 shows an example of the manual localization of the markers. The colors of these markers are saved and used for thresholding and detection of the markers in the following frames. The user is asked to first localize the marker on the foot, then the marker on the heel, then the marker on the ankle, etc.

After clicking on a marker, multiple steps are performed:

1. The x,y coordinates of the marker are saved.
2. If there is already some marker localized and therefore some color thresholds defined and markers detected, the detected markers are searched if there is some marker close to the clicked marker. If there is, the location of the clicked marker is updated to the location of the detected marker. This is done so that user does not have to precisely click on the marker. This is especially useful when the markers are small and the user is using a mobile device with a small screen.
3. Based on the x,y coordinates of the marker, the color of the marker is extracted from the image.
4. The color of the marker is used to update the color thresholds used to detect the markers. More on this in section 6.2.2.

6.2.2 Detection

After obtaining the markers locations and colors, the markers are detected in the following frames. The detection is performed in the following steps:

1. The frame is converted to the CIELAB color space.
2. The frame is thresholded using the minimum and maximum A and B channel values of the localized markers and some margin. The margin is used to account for the changes in the lighting conditions. Empirically, the margin of 15 works well. Figure 6.3 shows an example of the thresholded image.

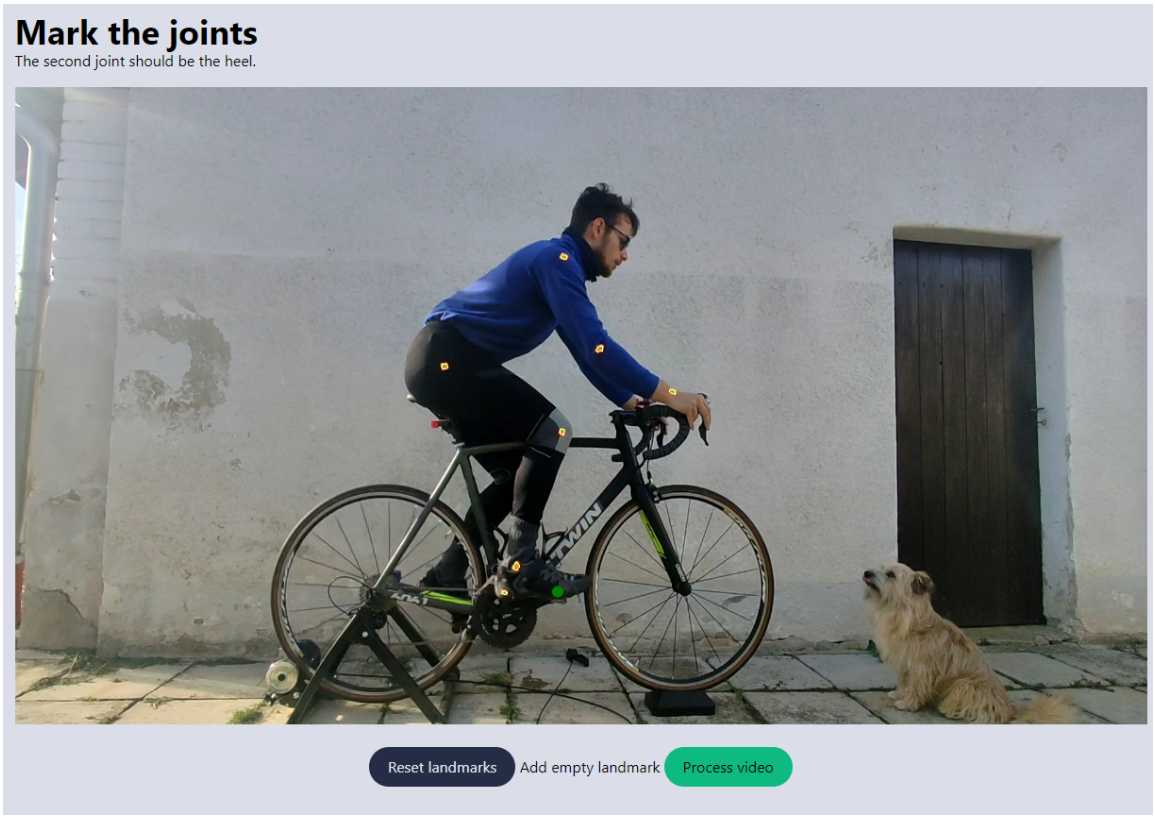

Figure 6.2: Example of the manual localization of the markers. The user is asked to click on the markers in the first frame of the video. The contours are highlighted in white. The centroids of the contours are highlighted in green.

3. Contours are found in the thresholded image. The `cv.findContours()` method is used for this.

4. The centroids of the contours are calculated using the `cv.moments()` method.

5. The centroids are used as noisy detections.

The CIELAB color space is used because it is designed to approximate human vision. The A and B channels are used because they represent the color of the marker. The L channel is not used because it represents the lightness of the color and is not very useful for detecting the markers due to the changes in the lighting conditions. Experiments with using the H channel of the HSV color space were also performed, but there were some problems with the detection of the markers on the edges of the color spectrum (H=0 and H=255). The CIELAB color space does not have this problem, because it uses two channels for the color.

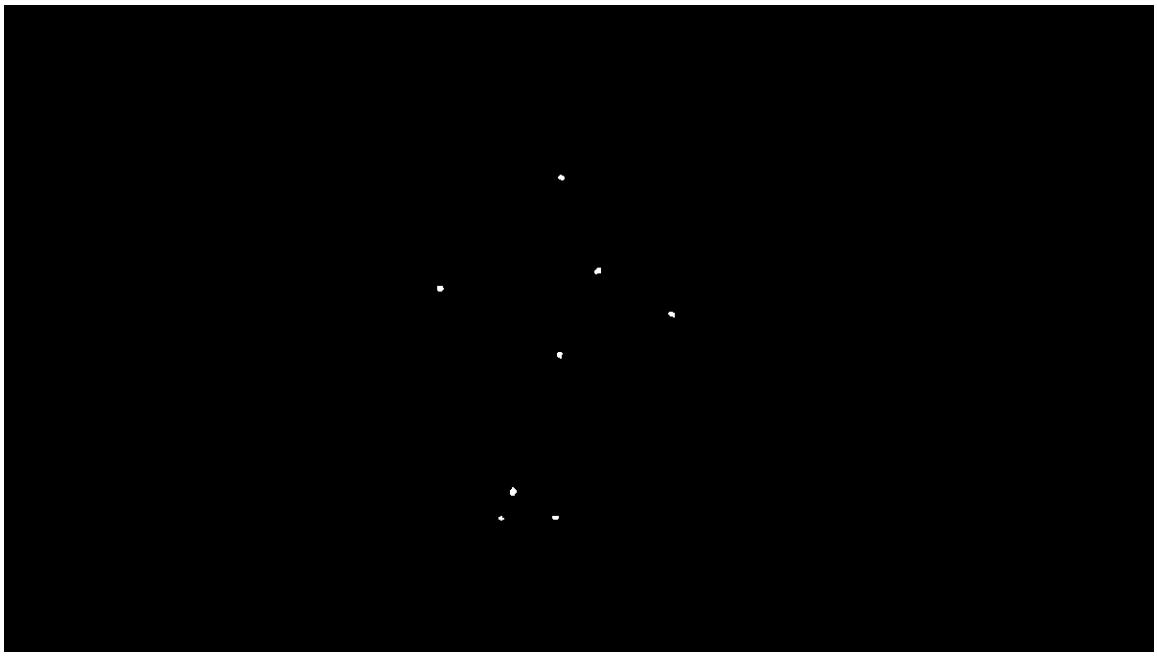

Figure 6.3: Example of the image thresholded using the minimum and maximum A and B channel values of the localized markers and a margin of 15. The markers are then detected as the contours in the thresholded image.

6.2.3 Tracking

The detected markers are then tracked using a simple method similar to SORT [3]. The tracking algorithm works as follows:

1. Every possible assignment of the detected markers and the markers in the previous frame is created. To limit the number of possible assignments, the assignments are created only if the distance between the markers is less than 50 pixels.

2. For every found assignment, the cost of the assignment is calculated as the sum of the distances between the markers in the assignment.

3. To make tracking more robust, the cost of the assignment is increased by the difference of the angles between the markers in the assignment. This is done to prevent the markers from switching places.

4. Assignment with the most markers assigned and the lowest cost is selected.

Experiments with using the Kalman filter were also performed. Unfortunately I was not able to get the Kalman filter to work properly. The Kalman filter was not able to predict the position of the markers and the tracking was not very robust. The simple method described above works better.

6.2.4 Limitations

6.3 Pose Estimation

6.4 Video Player

6.5 Presentation of the Results and Recommendations

6.6 User Authentication and Data Storage

Chapter 7

Experiments

Chapter 8

Conclusion

Bibliography

[1] Andriluka, M., Iqbal, U., Insafutdinov, E., Pishchulin, L., Milan, A. et al. PoseTrack: A Benchmark for Human Pose Estimation and Tracking. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[2] Andriluka, M., Pishchulin, L., Gehler, P. and Schiele, B. 2D Human Pose Estimation: New Benchmark and State of the Art Analysis. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2014.

[3] Bewley, A., Ge, Z., Ott, L., Ramos, F. and Upcroft, B. Simple online and realtime tracking. In: IEEE. *2016 IEEE international conference on image processing (ICIP)*. 2016, p. 3464–3468.

[4] Bike Fast Fit. *Bike Fast Fit Elite v2 - Basic Bike Fitting*. 2023. Available at: https://www.youtube.com/watch?v=4xhFhvIM7R4.

[5] Burt, P. *Bike Fit 2nd Edition: Optimise Your Bike Position for High Performance and Injury Avoidance*. Bloomsbury Publishing, 2022.

[6] Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y. et al. MMDetection: Open MMLab Detection Toolbox and Benchmark. *ArXiv preprint arXiv:1906.07155*. 2019.

[7] Contributors, M. *OpenMMLab Pose Estimation Toolbox and Benchmark* [https://github.com/open-mmlab/mmpose]. 2020.

[8] Fang, H.-S., Li, J., Tang, H., Xu, C., Zhu, H. et al. AlphaPose: Whole-Body Regional Multi-Person Pose Estimation and Tracking in Real-Time. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2022.

[9] Ge, Z., Liu, S., Wang, F., Li, Z. and Sun, J. Yolox: Exceeding yolo series in 2021. *ArXiv preprint arXiv:2107.08430*. 2021.

[10] Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B. et al. Searching for mobilenetv3. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, p. 1314–1324.

[11] Hua, W., Dai, Z., Liu, H. and Le, Q. Transformer quality in linear time. In: PMLR. *International Conference on Machine Learning*. 2022, p. 9099–9117.

[12] Jhuang, H., Gall, J., Zuffi, S., Schmid, C. and Black, M. J. Towards understanding action recognition. In: *International Conf. on Computer Vision (ICCV)*. December 2013, p. 3192–3199.

[13] Jiang, T., Lu, P., Zhang, L., Ma, N., Han, R. et al. RTMPose: Real-Time Multi-Person Pose Estimation based on MMPose. *ArXiv preprint arXiv:2303.07399*. 2023.

[14] Jin, S., Xu, L., Xu, J., Wang, C., Liu, W. et al. Whole-Body Human Pose Estimation in the Wild. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2020.

[15] Li, J., Wang, C., Zhu, H., Mao, Y., Fang, H.-S. et al. CrowdPose: Efficient Crowded Scenes Pose Estimation and a New Benchmark. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[16] Li, Y., Yang, S., Liu, P., Zhang, S., Wang, Y. et al. Simcc: A simple coordinate classification perspective for human pose estimation. In: Springer. *European Conference on Computer Vision*. 2022, p. 89–106.

[17] Lin, J., Zeng, A., Wang, H., Zhang, L. and Li, Y. One-Stage 3D Whole-Body Mesh Recovery with Component Aware Transformer. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, p. 21159–21168.

[18] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P. et al. Microsoft COCO: Common Objects in Context. In: Fleet, D., Pajdla, T., Schiele, B. and Tuytelaars, T., ed. *Computer Vision – ECCV 2014*. Cham: Springer International Publishing, 2014, p. 740–755. ISBN 978-3-319-10602-1.

[19] Lyu, C., Zhang, W., Huang, H., Zhou, Y., Wang, Y. et al. Rtmdet: An empirical study of designing real-time object detectors. *ArXiv preprint arXiv:2212.07784*. 2022.

[20] Maji, D., Nagori, S., Mathew, M. and Poddar, D. Yolo-pose: Enhancing yolo for multi person pose estimation using object keypoint similarity loss. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, p. 2637–2646.

[21] Retül Technology. *XY Fit Step 4: Saddle Fore-Aft*. 2021. Available at: https://www.youtube.com/watch?v=hZgI87DUUbU.

[22] Retül Technology. *XY Fit Steps 1-8: Markers and Harness*. 2021. Available at: https://www.youtube.com/watch?v=k-FsejpibKE.

[23] Ribeiro Branco, G., De Michelis Mendonça, L., Alves Resende, R. and Pivetta Carpes, F. Does the Retül System provide reliable kinematics information for cycling analysis? *Journal of Science and Cycling*. Dec. 2022, vol. 11, no. 3, p. 76–84. DOI: 10.28985/1322.jsc.15. Available at: https://www.jsc-journal.com/index.php/JSC/article/view/759.

[24] Sun, K., Xiao, B., Liu, D. and Wang, J. Deep High-Resolution Representation Learning for Human Pose Estimation. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[25] Taheri, S., Vedienbaum, A., Nicolau, A., Hu, N. and Haghighat, M. R. Opencv. js: Computer vision processing for the open web platform. In: *Proceedings of the 9th ACM Multimedia Systems Conference*. 2018, p. 478–483.

[26] Ultralytics. *YOLOv5: A state-of-the-art real-time object detection system* [https://docs.ultralytics.com]. 2021. Accessed: 14.01.2024.

[27] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L. et al. Attention is all you need. *Advances in neural information processing systems*. 2017, vol. 30.

[28] Wu, J., Zheng, H., Zhao, B., Li, Y., Yan, B. et al. Ai challenger: A large-scale dataset for going deeper in image understanding. *ArXiv preprint arXiv:1711.06475*. 2017.

[29] Xu, L., Guan, Y., Jin, S., Liu, W., Qian, C. et al. Vipnas: Efficient video pose estimation via neural architecture search. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, p. 16072–16081.

[30] Zhang, S.-H., Li, R., Dong, X., Rosin, P., Cai, Z. et al. Pose2Seg: Detection Free Human Instance Segmentation. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.