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

APP SUPPORTING STRENGTH SPORTS TRAINING
APLIKACE PRO PODPORU TRÉNINKU SILOVÝCH SPORTŮ

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR **Bc. RICHARD KLEM**
AUTOR PRÁCE

SUPERVISOR **prof. Ing. ADAM HEROUT, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2024

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Master's Thesis Assignment

155096

Institut:	Department of Computer Graphics and Multimedia (DCGM)
Student:	**Klem Richard, Bc.**
Programme:	Information Technology and Artificial Intelligence
Specialization:	Machine Learning
Title:	**App Supporting Strength Sports Training**
Category:	Mobile applications
Academic year:	2023/24

Assignment:

1. Analyze and describe the aspects of training of the selected strength sports, focusing on the importance of speed of exercise performance.
2. Get familiar with mobile application development.
3. Acquire and potentially annotate appropriate data for evaluating and potentially learning automatic algorithms for processing sports training videos.
4. Find suitable algorithms for processing the acquired dataset and experiment with them.
5. Design and iteratively improve the user interface of an application to support strength sports training based on user testing.
6. Integrate sub-elements of the solution into the application, test it with users, and refine it towards perfection.
7. Evaluate the results and propose future work; create a poster and a short video for presenting the project.

Literature:

- Nuncio Signore: Velocity-based Training: How to Apply Science, Technology, and Data to Maximize Performance, Human Kinetics, Athletes, 2022
- Ian Jeffreys, Jeremy Moody: Strength and Conditioning for Sports Performance, Strength and Conditioning for Sports Performance, Taylor & Francis, 2016
- Elisa Păduraru, Fundamentals of Creating a Great UI/UX, Creative Tim, 2022
- Tidwell et al.: Designing Interfaces: Patterns for Effective Interaction Design, O'Reilly, 2020
- Steve Krug: Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability, ISBN: 978-0321965516
- Steve Krug: Rocket Surgery Made Easy: The Do-It-Yourself Guide to Finding and Fixing Usability, ISBN: 978-0321657299
- Richard Szeliski: Computer Vision: Algorithms and Applications, Springer, 2nd, 2022

Requirements for the semestral defence:
items 1 and 2, considerable progress on items 3, 4, and 5

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor:	**Herout Adam, prof. Ing., Ph.D.**
Head of Department:	Černocký Jan, prof. Dr. Ing.
Beginning of work:	1.11.2023
Submission deadline:	17.5.2024
Approval date:	5.4.2024

Abstract

This thesis presents a novel approach for strength sports performance analysis using a human pose estimation machine learning model. The implemented solution employs the RTMPose model to estimate keypoints, then derive the barbell position from the wrist coordinates, and compute performance metrics without requiring specific camera angles or visible weight plates. The proposed method enhances traditional resistance training by providing feedback and performance metrics such as mean velocity. The solution was proved effective in both gym and home environments, even without barbells. Extensive experiments demonstrate the robustness and wide usability of the solution. Comparison with the professional system Qualisys confirms the validity of the application results.

Abstrakt

Tato práce představuje nový přístup k analýze výkonu v silových sportech pomocí modelu strojového učení pro odhad lidské pózy. Implementované řešení využívá model RTMPose k odhadu klíčových bodů lidské pózy, následně odvozuje polohu činky ze souřadnic zápěstí a vypočítává výkonnostní metriky. Konkrétní úhly kamery nebo viditelné kotouče nejsou vyžadovány. Navrhovaná metoda vylepšuje tradiční silový trénink tím, že poskytuje zpětnou vazbu a výkonnostní metriky, jako je průměrná rychlost, a ukazuje se jako účinná ve veřejných posilovnách i v domácím prostředí, a to i pro cvičení s vlastní vahou nebo pomůckami jako je např. odporová guma. Rozsáhlé experimenty potvrzují širokou použitelnost navrhnutého řešení. Srovnání s profesionálním systémem Qualisys prokazuje dostatečnou kvalitu aplikace.

Keywords

powerlifting, velocity-based training, computer vision, mobile application, human pose estimation, RTMPose, client-server architecture, monocular space calibration

Klíčová slova

silový trojboj, trénink založený na rychlosti, počítačové vidění, mobilní aplikace, odhad lidské pózy, RTMPose, architektura klient-server, monokulární kalibrace prostoru

Reference

KLEM, Richard. *App Supporting Strength Sports Training.* Brno, 2024. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor prof. Ing. Adam Herout, Ph.D.

Rozšířený abstrakt

Sportovci potřebují vědět, jaký mají ve své sportu výkon. To platí o to více, jedná-li se o sportovce na závodní úrovni. K zjištění sportovních výkonů lze využít několika různých způsobů měření. Vytrvalostní sportovci mohou například využít senzory srdečního tepu, aby věděli, jestli se nachází ve správném pásmu, jestli mají správné tempo apod. Dále pro ně jsou metrikami výkonu hodnoty jako čas, za který uběhnou či uplavou určitý počet metrů či délek.

Silové sporty mají ale zcela odlišný záměr. Siloví sportovci tedy sledují například únavu nebo obtížnost se kterou se potýkají během provádění cviku. Také sledují dynamiku svého pohybu, respektive rychlost a obtížnost provedení daného cviku. Například u vzpěračů nebo trojbojařů je důležité, jak rychle pohybují s činkou, kterou drží v dlaních. U strongmanů se pak může jednat ne jenom o činku, ale třeba o tzv. *log lift* nebo Atlasův kámen.

Hodnocení sportovního výkonu bývá zpravidla subjektivní, zejména na amatérské nebo středně profesionální úrovni. Zkušenější siloví sportovci používají k ohodnocení obtížnosti dané série či opakování škálu RIR nebo RPE. RIR vyznačuje počet opakování v rezervě. RPE pak na stupnici 0-10 obtížnost, kdy 10 je maximální výkon, zpravidla soutěžní úrovně a běžná hodnota v tréninzích se pohybuje okolo čísla 6. Posouzení obtížnosti nebo únavy závisí tedy na pocitech, vjemech a úsudku sportovce. V lepším případě pak více objektivně nahlížející osoby, a to trenéra či kouče.

Nabízí se tedy otázka, jestli existuje nějaký zcela objektivní způsob, jak změřit únavu, výkon nebo aktuální stav silového sportovce. K tomu lze využít analýzy rychlosti pohybu s použitím fyzických zařízení. Ta se připevní například k čince a pomocí gyroskopů a akcelerometrů dokáže spočítat rychlosti pohybu. Nejlevnější variantou jsou pak tzv. lineární převodníky. Ty pomocí namotaného lanka měří pohyb činky. Existují také komerční kamerové systémy, které však nejsou přenosné a jsou poměrně drahé (např. 3499 $).

Nabídka mobilních aplikací na obchodu Google Play nabízí pouze velmi malé množství aplikací, které by fungovaly samy o sobě. Dostupné mobilní aplikace pro Android spoléhají zpravidla na počítačové vidění, které detekuje kotouče z boku. Dále nesmí být sportovec moc blízko kamery, ale také ne moc daleko. To omezuje použití takovýchto aplikací, a to zejména ve společných prostorech, jako jsou fitness centra a veřejné posilovny, kde nelze obtěžovat ostatní sestavováním vhodných kamerových kompozic. Všechny zkoumané aplikace (např. WL Analysis, RepSpeed, Qwik VBT) vyžadují vesměs striktně boční pohled zaměřený na kotouče. WL Analysis dovoluje 30°odklon.

Jako řešení tohoto problému je představena nová metoda pro výpočet rychlosti pohybu. Řešení využívá strojové učení, specificky odhad klíčových bodů lidské kostry (pózy). Pro získání klíčových bodů kostry sportovce je použit model RTMPose z dílny čínské skupiny MMPose. Myšlenka vychází z faktu, že pro velké množství cviků se činka drží v rukou. Ze souřadnic zápěstí se tedy vypočítává pohyb činky, a nakonec dopočítá i rychlost.

Byla navržena a implementována aplikace s architekturou klient-server s velmi lehkou Android aplikací. Toho je docíleno zpracováním na straně serveru, která má za úkol provádět výpočetně náročné úkoly. To je nejenom samotná inference RTMPose modelu, ale i následné zpracování surových dat, jejich vícenásobné vyhlazení a počítání rychlostí a dalších derivovaných metrik.

Navrhované řešení nevyžaduje po uživateli žádný specifický úhel nebo polohu kamery. Kotouče mohou být zcela mimo zorné pole anebo nemusí být vůbec použity. Aplikace poskytuje výkonnostní metriky, jako je průměrná rychlost nebo průměrná propulsivní rychlost.

Bylo ukázáno, že pomocí tréninku založeného na sledování rychlosti (VBT) lze dosáhnout lepších výsledků za stejný čas anebo stejných výsledků s menší únavou nebo v kratším čase, než jak je tomu u tradičního tréninku založeného na běžném odporovém cvičení.

Kalibrace a převod z pixelů na metrické jednotky je zásadním problémem a prvkem provedeného řešení. Aplikace integruje několik heuristik, aby bylo dosaženo co nejvíce přesných výsledků. Nicméně i tak je zapotřebí dalších úprav a vylepšené kalibrace pro dosažení robustnějšího řešení.

Aplikace byla porovnána s profesionálním systémem Qualisys a ukázala, že za vhodných podmínek je schopna dodat velmi uspokojivé výsledky. Aplikace byla průběžně konzultována s uživateli, kteří přispěli k jejímu zdokonalení. Je navrženo několik možností rozšíření jako např. integrovaný virtuální trenér pro kontrolu správné techniky provedení cviku.

App Supporting Strength Sports Training

Declaration

I hereby declare that this Master's thesis was prepared as an original work by the author under the supervision of Mr. prof. Ing. Adam Herout, Ph.D. The supplementary information was provided by Ing. Jan Šťastný, PhD. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

. . . .
Richard Klem
May 17, 2024

Acknowledgements

I would like to thank my supervisor prof. Adam Herout, PhD. for his valuable advice and practical management, technical consultant Ing. Jan Šťastný, PhD. for his arrangement of Qualisys measurement session and perceptive comments from a technical point of view, Bc. Jakub Šigut for practical perspective consultation, and Petr Kříž for his helpful role as a reflexive marker actor. Finally, my thanks also go to the users who provided me with valuable comments during evaluation sessions.

Contents

Chapter 1

Introduction

Athletes need to know how they perform in their sport. Endurance athletes can, for example, use heart rate monitors. Strength sports athletes must observe their movement and lifted object dynamics. For example, how fast they move a barbell for weightlifters and powerlifters or a log for strongmen.

Sports performance evaluation tends to be subjective, particularly at the amateur or intermediate professional levels. Assessment of difficulty or fatigue depends on the athlete's (or preferably the coach's) feelings, sensations, and judgment. Analyzing the velocity of a movement usually involves using hardware devices that need to be attached to barbells. Accurate commercial camera systems are not portable and are expensive (e.g. $3499).

On the side of standalone mobile applications, there are very few options on Google Play. The mobile applications available on the Android market depend on computer vision to detect weight plates from the side, which restricts their use, especially in communal areas such as fitness centers. All of the examined applications (e.g. WL Analysis, RepSpeed, Qwik VBT) require a strict side view pointed to the weight plates.

To address this issue, a novel method for calculating movement velocity is introduced. RTMPose human pose estimation model is used to obtain the athlete's skeleton keypoints. From the wrist coordinates, the movement and finally the velocity are computed. The server-client architecture with the lightweight Android application and server-side processing that handles computationally intensive tasks.

The proposed solution does not require any specific camera angle or position; the weight plates could be completely out of view or not used at all. The application provides performance metrics such as mean velocity. The conversion from pixels/s to m/s is crucial and several heuristics are integrated in the solution to achieve accurate results. Android application can be installed from Google Play[1] after joining Google Group[2].

Chapter 2 explores the particular elements of strength sports. Chapter 3 discusses the sport analysis, camera systems and human pose estimation models. Chapter 4 discusses the wide topic of mobile development, including computation on edge and model compression methods. The design and implementation are described in Chapter 5 and 6. The results of the experiments, limitations, and future development can be explored in Chapter 7.

[1]https://play.google.com/apps/testing/cz.vut.fit.xklemr00.vbt.app
[2]https://groups.google.com/g/dyno-coach-testing

Chapter 2

Challenges in Strength Sports Training

This chapter describes the problematic of strength sports. It should introduce the reader to this topic and outline the context.

Strength sports, such as weightlifting, bodybuilding, and powerlifting are a subset of strength sports that emphasize speed and acceleration as key performance indicators. Specific aspects of strength sports are discussed as well as strength athlete's daily routine, including diet, recovery, mental strategies, and finally the technology use in training sessions. Fitness apps, wearable devices, and training tools that enhance performance, track progress, and prevent injuries are examined. As the result, implicit requirements and usage behaviour are gathered as a prerequisite for creation of mobile application for strength sport athletes.

2.1 Unique Aspects of Strength Sports

Strength sports, a category of resistance training, have unique aspects compared to sports such as tennis, football, and athletics. Within strength sports, differences exist, such as strongman's endurance disciplines. For explaining VBT, weightlifting will be discussed in detail. Powerlifting is also a focus because the majority of data captured during experiments is focused on powerlifting disciplines.

Signore [57] compares different sports using power as a comparison tool. The power definition is $P = F * v$. Powerlifting emphasizes force, American football balances force and velocity, and baseball focuses more on velocity. The SAID (Specific Adaptation of Imposed Demands) principle states that specific activities, loads, and demands are necessary for optimal performance in each sport. This section discusses weightlifting and powerlifting in detail to highlight their specific demands and characteristics.

As official Olympic sources [66] describe, Olympic weightlifting, or simply weightlifting, is a well-established Olympic sport since 1896 and the International Weightlifting Federation (IWF) was founded in 1905. But until Sydney's 2000 Olympics, this sport was men only.

Weightlifting includes the following disciplines: *snatch* and *clean and jerk*, both starting with the bar on the ground and ending overhead. The snatch lifts the bar in one motion, while the clean and jerk involves lifting to the chest, then overhead. Competitors get three attempts per lift, with one minute per attempt. Failed attempts must use the same or higher weight, while successful attempts can increase the weight.

Powerlifting, on the other hand, is not part of the Olympic games despite long-term efforts to achieve this goal, as Gaston, the president of the International Powerlifting Federation (IPF), says [49]. But a new highest-level competition has recently been created, named Sheffield. To be nominated for this competition, an athlete must have won the last world championship or intentionally hold the world record[1].

Powerlifting consists of three disciplines: squat, bench press, and deadlift, as defined by the IPF[2]. In the squat, the lifter descends with a barbell on their shoulders and returns to standing. In the bench press, the lifter lowers a barbell to their chest and presses it upward. The deadlift involves lifting a barbell from the floor to an upright stance and then lowering it back down. The number of attempts and time requirements are the same as in weightlifting. See Figures 2.1, 2.2, and 2.3 for illustrations.

The 60-second time limit to prepare, wait for the command, and start the lift shows that competition-level attempts are short-time movement. During warm-ups, athletes perform various sets and repetitions similar to regular training, a common practice in other strength sports like strongman and bodybuilding.

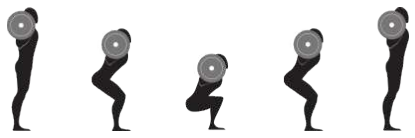


Figure 2.1: Movement of the squat. Drawing edited from Radenković [51].

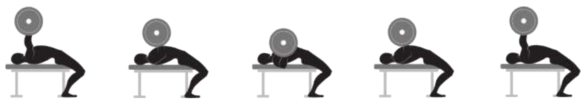


Figure 2.2: Movement of the bench press. Drawing edited from Radenković [51].

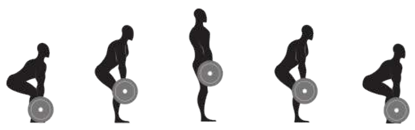
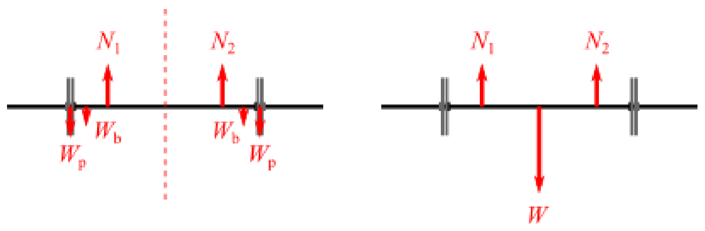


Figure 2.3: Movement of the deadlift. Drawing edited from Radenković [51].

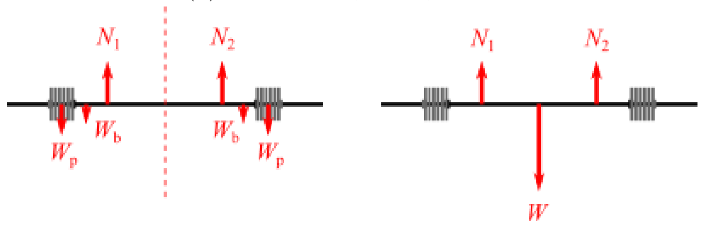
Radenković discusses the importance of the center of mass during lifts. Figures 2.4a and 2.4b show different distances from the center of mass to the holding points for the same weight on a bar. If an athlete's force isn't perfectly balanced, a tilt in the bench press increases the moment of inertia (case *b*), causing a shift in the force vector. This can lead to a failed lift due to arm leverage, bench size, and moment of inertia.

[1]Wildcard possibility applies.

[2]https://www.powerlifting.sport/about-ipf/disciplines



(a) Bar *a* using 15 kg plates.



(b) Bar *b* using 5 kg plates.

Figure 2.4: Force applying at center of mass for two bars with the same total weight but different plates usage. Drawings from Radenković [51].

Competition-level calibrated thin plates in powerlifting ensure that the barbell can be loaded heavily while maintaining stability and safety for the athlete. We want to emphasize the physics background of sports. Analyzing movement in detail is not only better for performance, but also a way to prevent injuries in the long term.

2.2 Strength Sport Training Programming

Strength sport programming involves the creation of training schemes and the preparation for competitions. Parry [50] outlines six pillars: specificity (focus on main exercises – squat, bench press, deadlift), overload (sufficient load for muscle growth), recovery (avoid overtraining), variation (change exercises, sets, and reps to maintain progress), individualization (tailor to individual differences), and reversibility (acknowledge performance decline from not training and the effort needed to regain it).

For specific exercises, do singles at 90% 1RM, doubles and triples at 80% 1RM, and sets of 5 reps at 70-85%. Accessory exercises such as glute bridge or triceps extension focus on hypertrophy, not high 1RM percentages. Avoid consecutive heavy squats. Training cycles include macrocycles (up to a year for major competitions), mesocycles (8-12 weeks, for major changes), and microcycles (weekly, for minor adjustments).

At the end of macrocycles, training plans must be adjusted for recovery and performance, known as tapering. Travis et al. [64] highlight the lack of evidence-based tapering protocols for strength sports, but note that tapering is established for endurance sports. Their review suggests a 30-50% volume decrease for 1-2 weeks and 2-7 days of training cessation to improve competition performance in strength sports.

Evaluating technique, form or RPE is better done with a coach than by self-assessment. Traditionally, this involves in-person sessions, which are time-consuming and costly. Online coaching options like Team Rohr[3], TSA[4], and The Strength Guys[5] offer remote coaching, often using video reports from training sessions for weekly evaluations and adjustments. These programs typically use spreadsheets with customized formulas and formatting, and some have their own mobile apps. Athletes are used to record their training sessions on a daily basis.

2.3 Velocity-Based Training Methodology

Velocity-Based Training (VBT), is a methodological pivot in strength training that emphasizes velocity and acceleration as key performance indicators. The primary focus of VBT is on dynamic strength sports such as weightlifting; however, this methodology is transferable to other strength sports (powerlifting, strongman, . . .)

Signore [57] investigates the importance of velocity in strength sports. In Table 2.1 it can be seen the approximate conversions between strength zones, velocity, and 1RM percentage. The ranges are made with average athlete in mind.

	Speed-Strength	Strength Speed	Accel. Strength	Absolute Strength
v[m/s]	1.0-1.3	0.75-1.0	0.5-0.75	<0.5
1RM[%]	20-40	40-60	60-80	85-100

Table 2.1: Conversions between strength zones, velocity and 1RM percentage according to Signore [57].

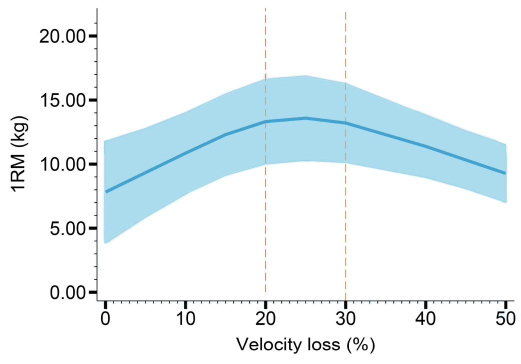
Zhang et al. [74] debate optimal velocity loss for strength development. The study examines velocity loss during velocity-based training (VBT) and its impact on maximum strength and training efficiency. They reviewed nine controlled trials with 336 trained men, using 1RM gain and 1RM gain per repetition as indicators.

Previous discussions suggest a lower velocity loss threshold for elite athletes. Zhang et al. support this statement with a non-linear (reverse U-shaped) relationship between velocity loss and 1RM gain (Figure2.5a), with an optimal range of 20-30%. A negative linear relationship between velocity loss and 1RM gain per repetition (Figure 2.5b) indicates that lower velocity losses improve training efficiency.

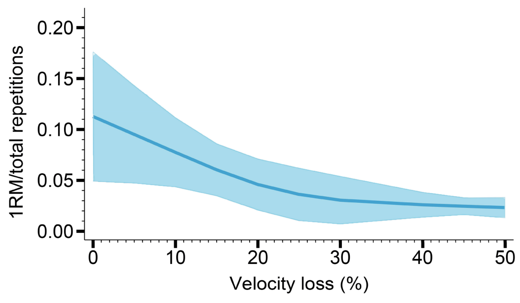
From a physiological perspective, the differential effects of velocity loss on strength development are linked to training-induced muscular adaptations. Excessive velocity loss might enhance endurance over maximal strength by altering muscle fiber activation, enlarging slow-twitch fibers at the expense of fast-twitch fibers. Lower velocity losses in VBT can maintain maximal strength with less training volume, conserving energy and time for other modalities.

However, limitations include the exclusive focus on male participants, a narrow range of velocity loss metrics, and a lack of differentiation between lower and upper body exercises. González-Badillo et al. [14] emphasize the importance of accurately estimating effort in resistance training. Their study shows that monitoring velocity combats the limitations

[3] `https://www.teamrohr.com/powerlifting-training/`
[4] `https://www.thestrengthathlete.com/`
[5] `https://thestrengthguys.com/`



(a) Reverse U-shaped correlation between velocity and 1RM.



(b) Negative correlation between velocity loss and 1RM gain per repetition.

Figure 2.5: Findings about velocity loss and 1RM relationships. Retrieved from [74].

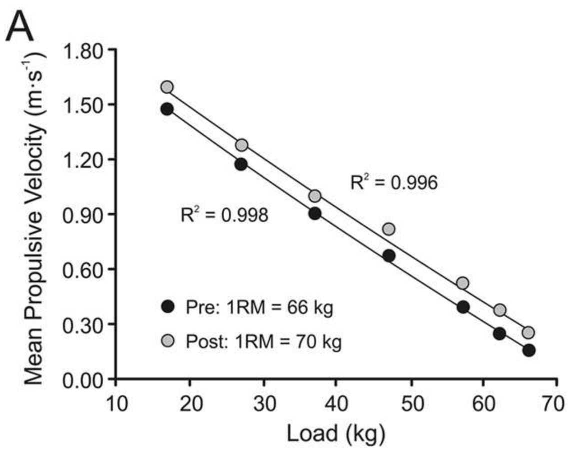


Figure 2.6: Load-velocity relationship. Pre- and Post-performance lines are drawn showing a 6.1% increase in strength. Drawing taken from González [14].

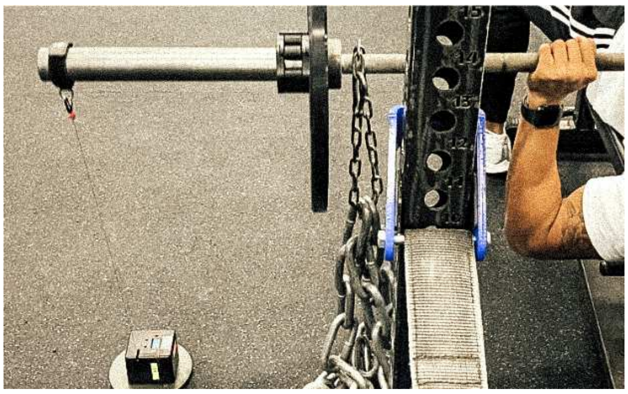
of subjective effort evaluation, demonstrating a 6.1% improvement in 1RM and increases in all other loads as can be seen in Figure 2.6. This was consistent across various VBT interventions with minor differences.

González et al. [14] highlight that velocity-based training (VBT) uses real-time data to guide training, measuring lift velocity for immediate feedback. This allows precise, individualized training loads, enhancing strength gains by accounting for daily fluctuations. Unlike subjective RPE, VBT provides objective data. Velocity thresholds optimize power and minimize overtraining, making VBT effective for power-oriented sports such as weightlifting or powerlifting.

VBT is very closely linked with measuring devices. The most popular is linear encoder that can be seen in Figure 2.7a. Picture is cropped from Dynamic Sports Training[6].

The other popular option is an accelerometer sensor. VBT relies heavily on measuring devices, such as linear encoders that can be seen in Figure 2.7a and accelerometer sensors,

[6]https://www.dynamicsportstraining.com/wp-content/uploads/2020/02/1-2.png


(a) Linear encoder.

(b) Bar sensor kit from Eleiko.

Figure 2.7: Examples of velocity measuring equipment.

such as those of Eleiko showed in Figure 2.7b. Image taken from manufacturer[7]. González et al. focus on these hardware devices. Camera systems are discussed in Section 3.3.

While velocity monitoring aids performance, it does not ensure optimal results or proper training. Effective VBT requires careful load and intensity management, enhancing already well-designed training plans by fine-tuning loads and individualizing programs.

González et al. critique the RT field's terminology, noting confusion due to inconsistent use of terms like "periodization", "maximal strength", and "power". This inconsistency complicates training design and evaluation. The term "periodization" for instance, lacks a uniform definition, leading to varied practices and reduced effectiveness. This highlights the need for validated results from sport studies.

2.4 Technology Usage and Habits in Strength Sports

This section explores key aspects of a strength athlete's daily routine, including diet, recovery, mental strategies, and technology use in training. It examines fitness apps, wearable devices, and training tools that enhance performance, track progress, and prevent injuries, highlighting how these can be integrated into mobile applications. The aim is to illustrate a typical day for a strength athlete, combining technology with traditional methods, to identify needs for fitness apps, particularly those supporting training.

Nutrition is vital for strength athletes. King et al. [36] surveyed 305 powerlifters, finding common practices like consuming carbohydrates before/during training and increased protein intake afterward. Smith [59] discusses the essential habits of powerlifters for long-term success, focusing on recovery over training. Key factors include adequate sleep, nutrition, injury prevention, and training at 75-85% intensity to enhance systems and technique. Long-term dedication, beyond short-term gains, is crucial. Elite athletes cannot miss sessions for minor reasons, and self-belief is vital, especially when training for new personal or world records without frequent 1RM testing.

[7]https://eleiko.com/en-cz/equipment/bars/3085286-eleiko-bar-sensor-kit

Even in strength sports, monitoring of heart rate (HR) is important. Malone [43] suggests a range of 70-85% of maximum HR for demanding exercise and 50-70% for medium intensity. Powerlifters monitor HR during training to avoid heart-related problems. Gotcher and Baraki [24] note that high intra-abdominal pressure in strength sports strains the heart.

Fitness or smart watches can monitor HR, but chest HR trackers are the most accurate. Upper arm strap devices[8] are less restrictive for strength sports, as they avoid interference with wrist wraps and chest contact with barbells.

For diet tracking, apps such as Strava[9] and KalorickéTabulky[10] are useful. Sleep, crucial for recovery, can be optimized with sleep coaching from Samsung[11] or Apple[12]. Velocity trackers for VBT are discussed in Section 3.2.

From the information provided in this chapter about various aspects of strength sports, it can be said, that strength sport athletes deal with many optimization during their daily life, training sessions, and preparation for competitions. In all aspects, mobile applications offer helpful functionality.

This creates a fertile ground for developing computer vision-based applications because athletes are already using smartphones as video recording devices on a daily basis as can be seen in Section 2.2.

[8]https://www.wired.com/gallery/best-heart-rate-monitors/
[9]https://www.strava.com/
[10]https://www.kaloricketabulky.cz/
[11]https://insights.samsung.com/2023/10/06/7-sleep-and-health-benefits-of-galaxy-watch6/
[12]https://support.apple.com/en-us/HT211685

Chapter 3

Sport Activity Analysis

Technology today offers various approaches for analyzing sports activities, primarily focusing on performance measurements like power in watts or time per distance. These are quantitative metrics. Qualitative metrics, such as technique quality, are harder to measure and often rely on subjective judgments by coaches or judges. Tenenbaum and Driscoll [63] delve into quantitative and qualitative metrics in sport. Objective performance metrics relevant to VBT, such as mean velocity (MV), mean propulsive velocity (MPV), and Impulse, are discussed in Section 3.1.

Modern devices enable more objective measurements by tracking keypoints of the human skeleton to define optimal movement patterns. These can then be compared with an athlete's performance to evaluate qualitative metrics. The following sections discuss devices, methods, and approaches for measuring both quantitative and qualitative metrics, with a focus on technological solutions. This includes camera systems, accelerometers, and especially monocular camera systems that use video recordings without additional hardware.

3.1 Metrics in Strength Sports

The mean propulsive velocity (MPV) [15] focuses on the propulsive phase of the movement to estimate the load. The propulsive phase was defined as the portion of the concentric phase during which the measured acceleration α is greater than the acceleration due to gravity ($\alpha \geq -9.81 m \cdot s^{-2}$).

According to Carl Valle [65], impulse is the most important metric. The impulse is defined as the product of force and time. The importance of force is supported by Sarabia et al. [54]. They conducted a study comparing power-based training with traditional resistance training. The outcome is that the former group produced the same performance as the latter group but with less neuromuscular fatigue. Especially good results were found in a short-term time horizon which is usable for the pre-competition training phase.

The conversion from the theoretical definition of impulse to usable and effective computation follows. For a continuous time, the impulse is defined as

$$I = \int_{t_1}^{t_2} F(t) dt, \tag{3.1}$$

where $F(t)$ is the force (a function of time), and t_1 and t_2 are the start and end times of the period over which was measured.

In discrete time, which is also the case for sampled video sequences, the equation can be rewritten accordingly.

$$I = \sum_{i=1}^{N} I_i = \sum_{i=1}^{N} (F_i \times \Delta t)$$

There is an issue of how to estimate the force without special force plates or other force measurement-capable hardware. In the context of a standalone mobile application, the captured video is the only available input that can be measured. To address this issue, we can apply the 2nd Newton's law $F = m\frac{dv}{dt} = ma$. This equation is usable with the user's input of overall weight on the bar. The mass is obtained from the user, and the acceleration can be computed from the velocity.

However, this would introduce unwanted error propagation. This can be avoided by rewriting the equation to our advantage. Since $a = v\frac{d}{dt}$ and $v = s\frac{d}{dt}$, we can rewrite the formula in discrete time:

$$a = \frac{\Delta v}{\Delta t}.$$

Now we can adjust the preceding equation from $I = F\Delta t$ to $I = ma\Delta t = m\frac{\Delta v}{\Delta t}\Delta t = m\Delta v$. This is indeed the impulse-momentum theorem.

3.2 Trackers

For measuring movement speed (velocity) and calculating their derivatives, physical trackers are a usual option. The wide spread of this kind of device is supported by usage in trustworthy sports studies such as one from González et al. [14] where they used a 1000 Hz linear velocity transducer.

Brooks et al. [4] mention three main types of devices available for VBT purposes. Two of them are physical trackers, the third one is a camera-based system. Let us now discuss the linear position transducer (LPT) and inertial measurement unit (IMU/accelerometers). Camera systems are addressed in Section 3.3.

3.2.1 Linear Position Transducer

Linear Position Transducers (LPTs) measure barbell velocity with high accuracy by quantifying the speed of a pulled tether during the concentric phase of exercises. The spindle's rotation, triggered by the tether, provides direct velocity measurements. Design optimizations minimize friction and safety risks, with the device placed on the floor and the strap connected to the bar via a magnet or lightweight collar. Recent LPT advancements include range-of-motion assessments and three-dimensional tracking, crucial for correcting technique inconsistencies.

LPTs focus on concentric movements, sometimes capturing irrelevant data, but users can filter this out. Despite higher costs, LPTs are preferred for their accuracy and reliability in strength training analysis, highlighted by Brooks et al. [4]. Devices like GymAware and RepOne enhance LPT functionality for performance optimization.

3.2.2 Accelerometers/Inertial Measurement Units

Accelerometers and Inertial Measurement Units (IMUs) are more affordable than LPTs and offer adaptable interfaces, initially used in wearable devices and later adapted for barbells. According to Brooks et al. [4], despite rapid advancements, IMUs lack extensive research on their validity compared to LPTs.

IMUs calculate velocity and other metrics using algorithms, leading to potential inaccuracies versus LPTs. The placement of barbells affects the accelerometer readings, and they are less reliable for dynamic movements. However, their portability allows them to be used in various activities beyond weightlifting, such as sprinting or jumping. Both LPTs and IMUs rely on supportive applications for data computation and output. Companies typically provide end-user applications, sometimes with dedicated devices for fitness centers. Examples of LPT and IMU units has been shown in Figures 2.7a and 2.7b.

3.3 Camera Systems

Per Hasan and Alani [28], computer vision is essential in human movement analysis, balancing, and postural control. Advancements include multi-camera networks for sports performance and low-cost cameras for tennis tracking. Innovations extend to automatic sports video event detection using techniques like probabilistic Latent Semantic Analysis and Conditional Random Field Models. Multi-target tracking, crucial in surveillance and sports analysis, has progressed with deep learning and support vector machines.

Monocular vision (single camera) limits depth perception and balance, while binocular vision (two cameras) provides overlapping visual fields. Advances in robotics and sports science use monocular vision for tasks like tracking racket trajectories and spinning balls in table tennis, requiring sophisticated image processing. Recognizing human motion in videos, crucial in computer vision and machine learning, faces challenges in complex backgrounds and with camera motion. Applications include human-computer interfaces, biometrics, and surveillance. Traditional movement analysis uses CNNs. Other methods like Multilayer or Self-Organizing Background Subtraction were used in the past. They are often highly influenced by factors such as frame rate, lighting, and background movement.

Brooks et al. [4] discuss camera systems like Perch and EliteForm, which use multi-camera arrays to analyze exercise movements, similar to Microsoft Xbox Kinect. These systems measure the velocity of the barbell and the athlete and the three-dimensional movements, capturing multiple points for detailed movement analysis. Despite providing advanced data, their indirect measurement can be unreliable for bar speed and may result in data loss if the athlete moves out of frame. The high costs and installation requirements limit accessibility, though they are used in elite sports. These products offer extensive data and real-time feedback without the need to be attached to equipment, but cost and computational demands restrict widespread use.

The selection and pricing of VBT systems are shown in Table 3.1. Camera systems are the most expensive, often exceeding \$2000. Accelerometers and IMUs are more affordable, and thus suitable for personal or sport club use. High-tech professional systems such as Vicon or Qualisys for laboratories can cost more than \$100,000. Qualisys, used in the experiments is detailed in Section 3.3.1.

Name	Company	Type	Cost [$]
GymAware	Kinetic Performance Technology	Camera	2750
RepOne	Squats and Science Labs	LPT	399
Tendo	Tendo Sport	LPT	1329
Bar Sensei	Assess2Perform	Acc./IMU	375
Beast Sensor	Beast Technologies	Acc./IMU	289
FLEX	Kinetic Performance Technology	Acc./IMU	495
PUSH Band	PUSH, Inc.	Acc./IMU	449
VMaxPro VBT Tracker	SimpliFaster	Acc./IMU	329
Perch	CataLyft Labs, Inc.	Camera	1999

Table 3.1: Selected commercially available VBT systems covering all three main types. Data taken from Velocity-Based Training: Current Concepts and Future Directions [4].

3.3.1 Qualisys – Motion Capture Camera System

Londo et al. [42] discuss motion capture systems. Motion capture systems can track and record movements in three-dimensional space with high precision. They are used in biomechanics, sports science, medical rehabilitation, animation, and industrial research. Optical systems use high-speed cameras and markers – reflective (passive) or infrared-emitting (active) – to calculate positions via triangulation. Qualisys provides such systems, with high-speed cameras and software for real-time visualization, marker labeling, and data analysis.

In sports science[1], Qualisys captures detailed movement data to evaluate athletic techniques, identify improvement areas, and design training programs, reducing the risk of injury. In medicine[2], it evaluates gait patterns and monitors the progress of therapy for personalized rehabilitation. In entertainment[3], it records actor movements for realistic animations in films and games, integrating with Unity and Unreal Engine. Qualisys also supports industrial uses, including human-robot interaction, UAVs, and automotive solutions.

The Qualisys system was used as a reference for evaluating a human pose estimation model during *in the wild* experiments, detailed in Section 7.6.

The kit for sports studies and experiments usually consists of a number of Arqus[4] cameras and a few reference Miqus[5] cameras. The setup uses passive reflexive markers. At a distance of 10 meters, the 3D resolution can be up to 0.03 mm. 3D resolution is the smallest detectable motion that can be measured at a specific distance[6]. In Figure 3.1 the setup available at FEKT BUT laboratory can be observed.

[1] https://www.qualisys.com/life-sciences/human-biomechanics-and-sports-research/

[2] https://www.qualisys.com/analysis/gait/

[3] https://www.qualisys.com/entertainment/animation/

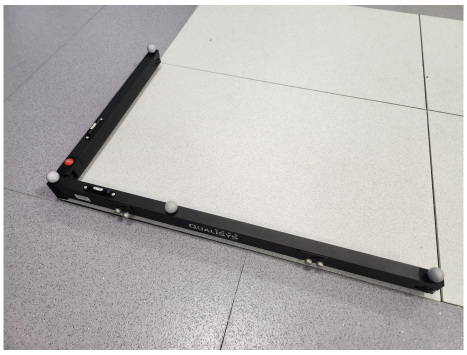
[4] Arqus is the main camera capturing the movement from reflexive markers.

[5] Miqus is camera with no captuering capability, it is used to record the movement as casual video recording to have a visual reference what was happening in the monitored area.

[6] https://www.qualisys.com/cameras/arqus/#tech-specs

(a) Sports laboratory at FEKT BUT with Qualisys motion capture system.

(b) Arqus camera (on the left), connected to next ones in a chain using RJ-45 connector.


(c) Calibration construction used before each session to calibrate the Qualisys system.

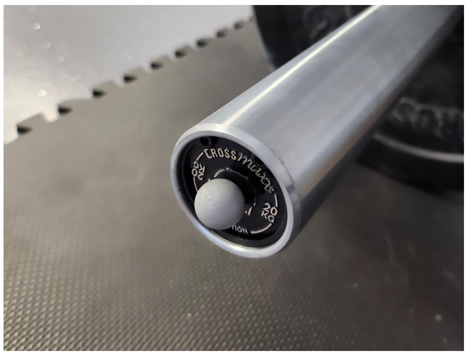

(d) Passive reflexive marker placed on the very end of a barbell.

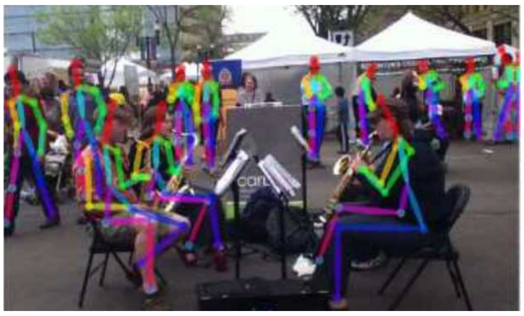
Figure 3.1: Qualisys motion capture system at FEKT BUT, managed by CESA BUT.

3.4 Human Pose Datasets

There are multiple pose datasets covering the whole human body, human hand, or face, but also animals or even anything as valid for Pose for Everything [71]. More conventional datasets focused on the human body are discussed first. Then detail-centric datasets follow, and other object datasets and specialties like Pose for Everything close the section.

Traditional accuracy and precision metrics used for evaluating the model's performance can be used in object detection task. Usually, the average precision (AP) and the average recall (AR) are used with the popular variant of the mean average precision (mAP) as stated by the authors of the COCO dataset [41].

For object detection, the tightness of the bounding box is used as a metric, represented by the Intersection over Union (IoU) value. IoU $= \frac{S_{overlap}}{S_{union}}$ is a ratio of overlapping area over merged area of the predicted and ground-truth bounding boxes. For keypoints localization, the authors of the COCO dataset prescribe an Object Keypoint Similarity (OKS) metric that mimics the IoU and is described in more detail in Section 3.4.2.


(a) Multiple people scene. Taken from [62].

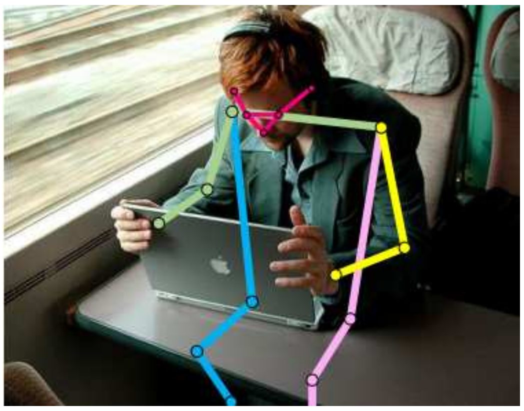

(b) Occluded scenery. Taken from [12].

Figure 3.2: Examples of dataset samples.

3.4.1 Keypoints

In computer vision and human pose estimation, keypoints are specific points on an object or human body, such as joints and anatomical landmarks, defining shape and pose. Detailed pose estimation may include facial expressions or small bones in the hands and feet. Keypoints are used in tasks such as pose estimation, object detection, and tracking.

Keypoints are detected and localized using deep learning models, particularly convolutional neural networks (CNNs), trained on annotated datasets. These models are discussed in Section 3.5.2. Understanding the semantic relationship between keypoints is crucial, especially for multi-person pose estimation where the bounding boxes overlap, as shown in Figure 3.2a. Keypoint detection faces challenges such as occlusions, varying lighting, and complex backgrounds as shown in Figure 3.2b.

3.4.2 COCO

Published by Microsoft in 2014, the COCO dataset [40] introduced a novel approach to object segmentation. The novelty is in annotating each instance, not just each class. The COCO data set includes more than 200,000 labeled images, covering 80 object categories and 500,000 instances, including the *person* class. For human keypoints localization, 250,000 person instances are labeled, generating 1.7 million keypoints. The example can be seen in Figure 3.3.

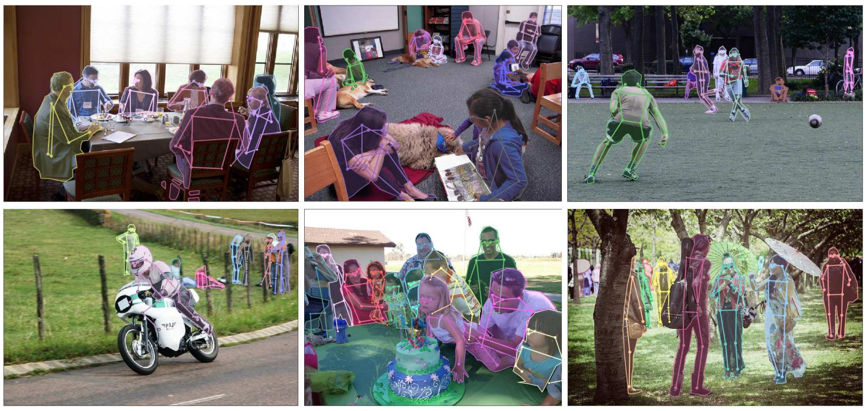


Figure 3.3: Example of keypoint localization labeled images from COCO dataset. Retrieved from [41].

The keypoint task in the COCO dataset is defined as a joint task of object detection and keypoint localization. The authors introduce a novel metric for this task while taking inspiration from the traditional object detection metric. OKS is defined as follows:

$$OKS = \frac{\sum_i e^{\left(\frac{-d_i^2}{2s^2k_i^2}\right)^{\delta(v_i>0)}}}{\sum_i \delta\left(v_i > 0\right)}, \tag{3.2}$$

where d_i is the i-th Euclidean keypoint distance between the predicted and ground truth, k_i is the i-th keypoint specific weight, s is a scale of an object, and $\delta(v_i)$ is the i-th keypoint visibility flag.

In 2020, the DensePose task was added with a tight cooperation with Facebook Research based on their DensePose model [26]. DensePose model is discussed in more detail in Section 3.5.5.

3.4.3 MPII

MPII [2] is a dataset from Max Planck Institutes for Informatics and Intelligent Systems and Stanford University introduced in 2014. The data source was 3, 913 YouTube videos after filtering and manual query adjustments. The dataset consists of more than 40,000 images of people in various situations. The authors state that MPII with 491 unique actions is complementary to J-HMDB [33] which has 21 distinct actions (and almost 32,000 images).

In Figure 3.4 can be seen the upper body pose skeleton called *armlets*. In the middle is the variant from Articulated Pose Estimation Using Discriminative Armlet Classifiers [13] and on the right is the MPII proposed dataset armlets pose distribution.

The beneficial contribution of the MPII dataset can be seen from the more dispersed distribution of arms. In contrast, the original Armlets dataset shows an observable pattern of arms concentrated down along the body as Andriluka et al. [2] point out. For that reason, the MPII dataset is considered as a valuable contribution to pose estimation tasks.

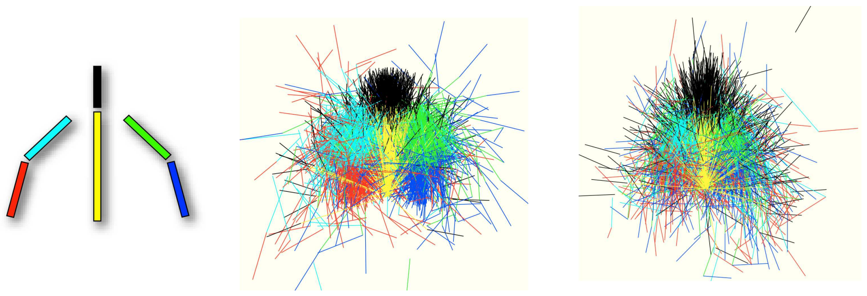


Figure 3.4: Upper body pose used in Armlets dataset [13] in the middle and with MPII variant on the right. Retrieved from [2].

3.4.4 Human3.6M

Human3.6M [32] is a comprehensive benchmark dataset with more than 3.6 million captured poses of actors performing various activities such as walking, sitting and sports. This dataset is used for training and testing algorithms in human pose estimation and activity recognition such MotionBERT 3.5.8, enabling the determination of 3D human poses from single or multiple camera views.

The dataset includes extensive annotations with 3D and 2D coordinates for human joints, recorded simultaneously from multiple camera angles, and enriched with high resolution 3D marker data from a motion capture system. The system used is Vitus Smart LC3 which is similar to Qualisys discussed in Section 3.3.1. This combination of synchronized video and motion capture data supports multi-modal learning approaches, integrating visual cues with precise 3D joint data.

3.4.5 AP-10K

AP-10K [73], released in 2021, is an animal pose dataset with 10,015 images that cover 23 animal families and 53 species, following COCO's 17 keypoints format. Unlike species-specific datasets, AP-10K adds the needed variance. The accuracy of the keypoint annotations was manually checked. The authors benchmarked pose models on supervised learning, cross-domain transfer from human to animal pose estimation, and domain generalization for unseen animals.

The distribution reflects natural family occurrence, with 913 images for Felidae and 200 for Procyonidae. Cross-domain transfer learning was less effective, probably due to differences like clothing vs. fur, but interfamily generalization showed better results. The dataset covers only a fraction of all families and species. See an example AP-10K labeled sample in Figure 3.5.

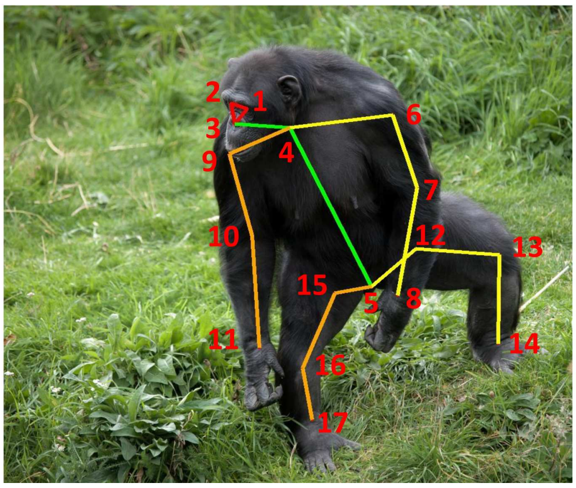


Figure 3.5: COCO style keypoints annotation on a labeled image of chimpanzee from AP-10K dataset. Taken from [73].

3.5 Human Pose Models

According to LeCun, Bengio and Hinton [38], recent computer vision machine learning models use deep learning and are usually based on convolutional neural networks. Researchers from Princeton University made a breakthrough with their idea of ImageNet [6]. Later in 2012, AlexNet won the ImageNet competition [37] and finally, in 2015, Microsoft dethroned AlexNet with their ResNet [29]. State-of-the-art models for human pose estimation often use ResNet as a backbone architecture and are discussed in Section 3.5.2.

3.5.1 Neural Networks in Computer Vision

Fundamentals and preliminaries for deep learning, machine learning oriented computer vision, convolutional neural networks, and other relevant topics are discussed in various sources such as the famous Deep Learning by Goodfellow et al. [16]. Only convolutional neural networks are discussed, as they are used very often in at least backbone architectures for human pose estimation models. More advanced aspects are mentioned by the examined model later.

Convolutional Neural Networks (from now on CNNs) are a class of deep neural networks specifically designed to process structured grid data, such as images. Originating from the field of computer vision, CNNs have become the core of most image and video recognition tasks due to their ability to learn spatial hierarchies of features from input images. Other positives of CNNs are their robustness to translation and distortion of input images and their scalability to large datasets. The well-known CNN architectures are AlexNet, VGG, ResNet, or Inception.

CNNs are composed of multiple layers that perform operations on the input data to extract relevant features. These layers typically include convolutional layers, pooling layers, and fully connected layers. The convolutional layers apply a set of kernels (learnable filters) across the input image, performing element-wise convolution (multiplications and summing the results) to produce feature maps. These feature maps capture various aspects of the input image, such as edges, textures, and patterns, at different abstraction levels. By stacking multiple convolutional layers, CNNs learn increasingly complex and abstract representations of input data.

Pooling layers, often following convolutional layers, perform down-sampling operations to reduce the spatial dimensions of feature maps, reducing computational load and controlling overfitting. Common pooling operations include max pooling, which selects the maximum value within a pooling window, and average pooling, which computes the average value within the window. There also exist unpooling and upsampling methods such as dense upsampling convolution presented by Wang et al. [68].

Fully connected layers, typically towards the end of the CNN architecture, act as classifiers. They take the high-level features extracted by the convolutional and pooling layers and use them to produce class scores, which predict the class labels of the input images.

CNNs are trained using backpropagation, where model parameters (filters and weights) are updated to minimize a loss function, typically using gradient descent-based optimization algorithms. During training, CNNs adjust their filters to detect important features indicative of target classes, effectively learning to recognize patterns in the data.

3.5.2 Human Pose Models

In researching existing solutions, multiple state-of-the-art models and datasets were examined, including Ultralytics' Yolov8, OpenMMLab's MMPose, and Facebook's Detectron2. These models cover not only human full-body poses but also detailed face, hand, animal poses, and large groups.

Recent models predict keypoint spatial coordinates using deep learning, often with CNNs or transformers. Earlier approaches, like Felzenszwalb and Huttenlocher's 2004 Pictorial Structures for Object Recognition [11], used statistical methods to model poses as skeletons from rectangles, predicting object configurations through probabilistic models. Before discussing specific models, the next section covers metrics for evaluating pose estimation models.

3.5.3 Model Evaluation Metrics

Percentage of correct keypoints (PCK) is an accuracy metric to measure the deviation of predicted keypoint and the ground truth. It is described by OECD.AI[7]. Usually, threshold variables are used, such as PCKh@0.5 (threshold = 50% of the head bone link[8]) or PCK@0.2 (distance d between the predicted and the true joint coordinates $d < 0.2 \times$ torso diameter). Sometimes, it can be fixed 150 mm distance.

Average Precision or Mean Average Precision represents the area under curve (AUC) of precision and recall. Furthermore, **Mean Average Precision** (mAP) is a mean of APs of each class. For human pose estimation, average precision is measured over OKS metric defines in Equation 3.2.

[7]https://oecd.ai/en/catalogue/metrics/percentage-of-correct-keypoints-%28pck%29

[8]Depending on the dataset, the head bone link is distance from top of the head to the neck or some similar distance.

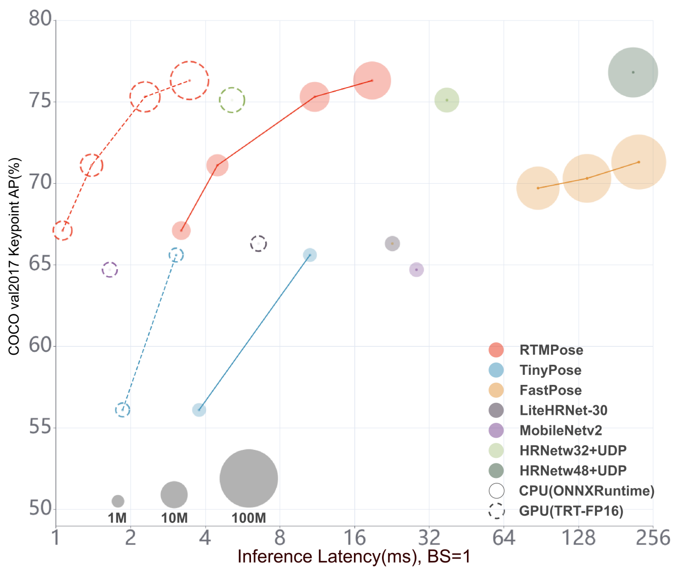


Figure 3.6: Inference latency to average precision comparison of RTMPose and a numerous of related works. Full circles represent performance using ONNX runtime on the CPU, void circles represent performance using TensorRT on the GPU. It can be observed that RTMPose should reach higher precision with lower inference latency for both CPU and GPU cases in comparison to almost any other example. Please notice the log_2 inference latency axis.

3.5.4 RTMPose

RTMPose [35] is a recent work from 2023 by Tao et al. from the Shanghai AI Laboratory research institute. RTMPose is built on the MMPose framework[9] and solves the task of multi-person pose estimation. RTMPose has been designed to balance performance and computational efficiency. It was tested on mobile chipset and it promises production-ready real-time performance even on edge devices such as smartphones. The inference latency to the average precision graph can be seen in Figure 3.6.

RTMPose is built on the MMPose framework and incorporates several unique aspects to enhance its performance. The input to RTMPose consists of images, a directory with images, or video frames. The detection model takes square input. This rectangular detection with a vertical 4:3 aspect ratio emphasizes the dimensions of natural human pose. The model requires standard preprocessing steps, including normalization and resizing, to prepare the data for inference. The output of RTMPose includes the coordinates of keypoints for each detected person for each image or frame.

RTMPose has been trained on several prominent datasets, including COCO and COCO-WholeBody. Data augmentation techniques using Cutout [7] are used during training to improve the model's ability to generalize to new data. Post-processing steps include keypoint refinement to enhance the final output.

In Figure 3.7 is shown RTMPose architecture emphasizing changes to previous architectures. RTMPose employs CSPNeXt (CNN-based neural network, lighter than ResNet50)

[9]https://github.com/open-mmlab/mmpose/tree/main/projects/rtmpose

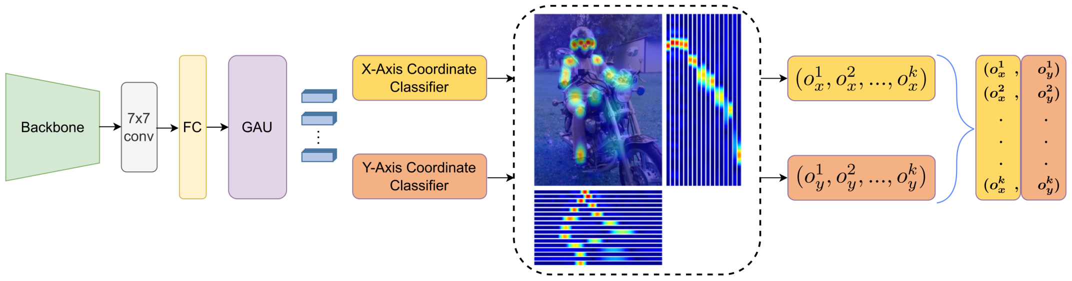

Backbone
7x7 conv
FC
GAU
X-Axis Coordinate Classifier
Y-Axis Coordinate Classifier
$(o_x^1, o_x^2, ..., o_x^k)$
$(o_y^1, o_y^2, ..., o_y^k)$
(o_x^1, o_y^1)
(o_x^2, o_y^2)
(o_x^k, o_y^k)

Figure 3.7: High-level architecture pf RTMPose model. The 2D heatmap observable in the middle of the architecture pipeline represents probability distributions predicted over each keypoint to achieve label smoothing. Retrieved from [35], edited by redrawing right side of scheme in higher quality.

in various sizes as its backbone. The detection model to find bounding boxes is Faster-RCNN, YOLOv3, PicoDet, or RTMDet.

A significant improvement in RTMPose is the integration of SimCC [39] into the keypoint head component. SimCC directly predicts the x and y coordinates of keypoints. The image space is divided into discrete bins and the model predicts the probability that each bin contains the keypoint. Although pixels are put in discrete bins, there are multiple bins per pixel and thus sub-pixel precision is reached. This approach improves the accuracy and efficiency of the model by providing a more straightforward method for keypoint localization.

On the side of RTMPose model integration in MMPose framework, the specification is described in MMPose documentation[10]. Experiments with the RTMPose model in various scenarios in the context of strength sports are described in Section 5.3.3.

3.5.5 DensePose

DensePose, introduced by Güler et al. [26] from Facebook AI Research in 2018, is built on Detectron and Mask-RCNN. Later it was integrated into the Detectron2 framework [70].

DensePose predicts a surface-based representation of the human pose using textures rather than keypoints. It handles real-world scenarios including street, nature, and sports activities, for both single and multi-person settings. DensePose was introduced along with a novel specific COCO dataset DensePose task discussed in Section 3.4.2. This model specializes on predicting surface-based representation of the human pose. Applications include augmented reality and semantic 3D object representation.

The DensePose architecture is based on CNNs, originating from DenseReg and Mask-RCNN. The baseline solution, DensePose-RCNN which can be seen in Figure 3.8 has similar architecture as Mask-RCNN.

DensePose uses distillation-based ground-truth interpolation, learning full density prediction from sparse annotations (100-150 points). This allows the loss function to cover every pixel in dense pose estimation.

[10] https://mmpose.readthedocs.io/en/latest/model_zoo/wholebody_2d_keypoint.html

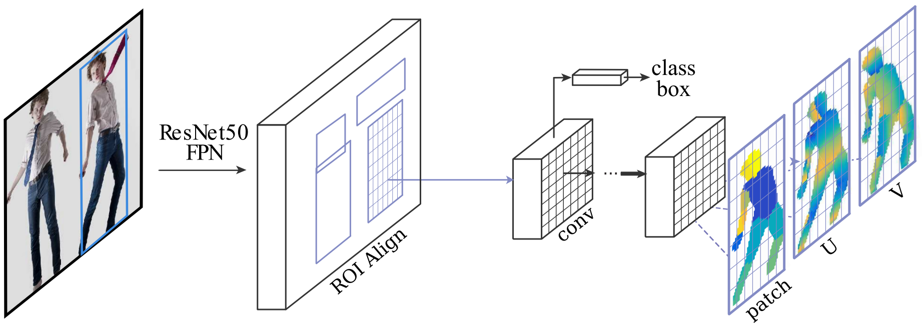


Figure 3.8: Base architecture of DensePose-RCNN without cross-cascading and teacher modules. The patch is used in the transfer learning for inpainting the predicted dense pose into the original image. The U and V coordinates are x and y axis values respectively for the fragment surface position. MaskRCNN has output as class labels, bounding box offsets, and object masks. Scheme edited from [26].

No other models are benchmarked with DensePose on Papers with Code. Güler et al. [26] showed DensePose outperforming SMPLify. Parsing R-CNN [72], surpassed DensePose-RCNN in the COCO 2018 DensePose Challenge (AP 64 vs. 56, AP_{75} 75 vs. 64)[11]. Despite this, DensePose won in AP_{75} and API metrics in the ECCV 2018 PoseTrack DensePose Challenge[12].

3.5.6 AlphaPose

AlphaPose [8], was developed by Fang et al. at Shanghai Jiao Tong University in 2022. The paper is an extension and enhancement of the preliminary work originally published in 2016, with the latest version [9] from 2018. The model is designed for whole-body multi-person pose estimation and tracking in real-time.

Along with the model, authors published also a new wholebody dataset named Halpe. Halpe data samples consist of 136 keypoints and cover the basic pose with detailed keypoints for the hand and facial. The keypoint labels can be seen in Figure 3.9. Halpe is used also for specialized face or hand-focused models such as RTMPose variant for hand.

[11] https://competitions.codalab.org/competitions/19636#results
[12] https://competitions.codalab.org/competitions/19650#results

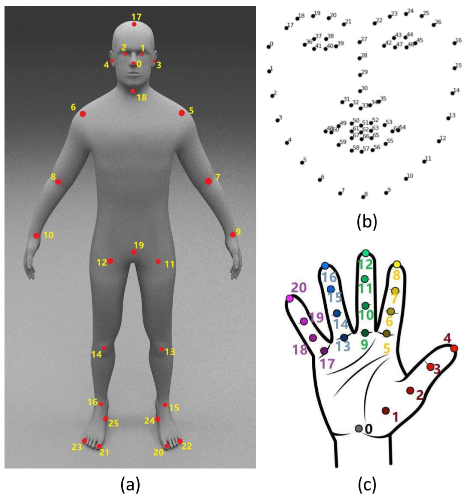


Figure 3.9: Format of annotations in Halpe dataset. On the left are 26 keypoints for body, on the top right are 68 facial keypoints and on the bottom right is annotated hand with 42 keypoints.

AlphaPose uses a top-down approach, detecting human bounding boxes and then estimating poses within each box. It integrates Symmetric Integral Keypoint Regression (SIKR) for fast keypoint localization, Parametric Pose Non-Maximum-Suppression (P-NMS) to remove redundant detections and Pose Aware Identity Embedding for tracking.

AlphaPose employs advanced methods such as Part-Guided Proposal Generator (PGPG) and multi-domain knowledge distillation to improve training accuracy. Its pipeline modules as shown in Figure 3.10 communicate via FIFO queues, enabling parallel computation for real-time performance.

AlphaPose ensures high detection recall by lowering detection confidence thresholds and applying Parametric Pose Non-Maximum-Suppression (P-NMS). The model's performance depends on the accuracy of the initial detection stage, as errors here affect subsequent pose estimation. It is available as an open-source model and dataset[13]. Experiments with AlphaPose in strength sports scenarios are described in Appendix F.

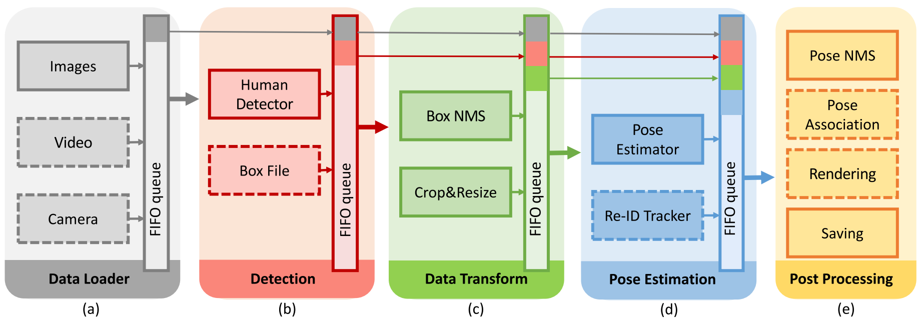


Figure 3.10: AlphaPose pipeline shown via multiple modules. Modules communicate with each other via FIFO queues, allowing efficient parallel computation and real-time application. Dashed components are optional. Retrieved from [8].

[13]https://github.com/MVIG-SJTU/AlphaPose/

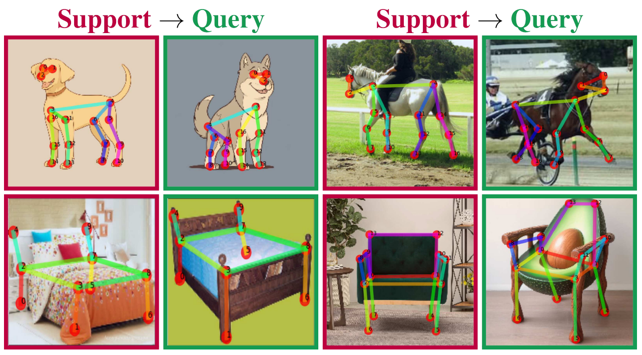


Figure 3.11: Pose Anything model pairs of support and query images. Support image consists of the image itself and the skeleton. The query image has already a predicted skeleton overlayed.

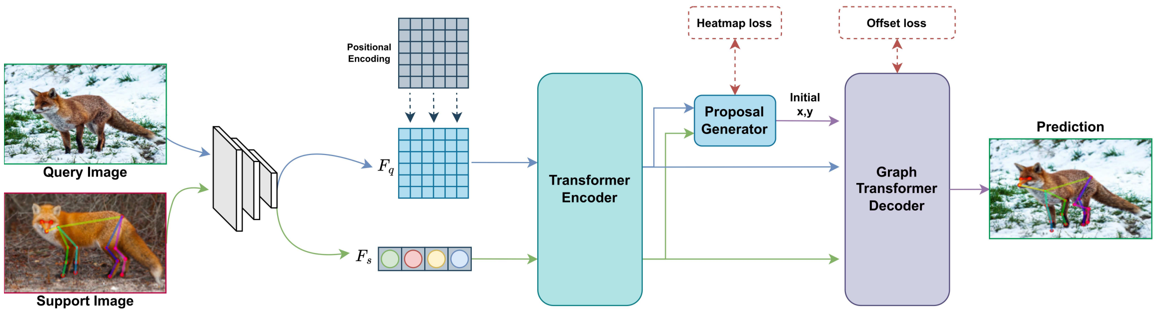


Figure 3.12: The architecture origins from CapeFormer model, replacing Resnet-50 to SwinV2-T and removing positional encoding. Furthermore, the authors replaced the original transformer decoder with their own novel Graph Transformer Decoder. Scheme retrieved from [31].

3.5.7 Pose Anything

Pose Anything: A Graph-Based Approach for Category-Agnostic Pose Estimation [31] from 2023 is a recent research introducing Graph Transformer Decoder as a novel method to use keypoints as graph data to predict the custom keypoint structure for any possible object. The example of inputs and outputs is shown in Figure 3.11. The high-level architecture of the Pose Anything model can be seen in Figure 3.12.

3.5.8 MotionBERT

Besides traditional 2D pose estimation, there is also 3D pose estimation, which can be done on monocular input or multiple angles input. For purposes of usage on a single mobile device, the monocular 3D pose estimation is more relevant. One of the best performing monocular 3D pose estimation models is MotionBERT [75]. According to the ranking[14] MotionBERT is the best performing model. Based on a multiview ranking[15] monocular MotionBERT is the second best and has transposed several multiview models[16].

[14]https://paperswithcode.com/sota/monocular-3d-human-pose-estimation-on-human3
[15]https://paperswithcode.com/sota/3d-human-pose-estimation-on-human36m
[16]Leaderbord was sorted by Mean Per Joint Position Error (MPJPE) metric.

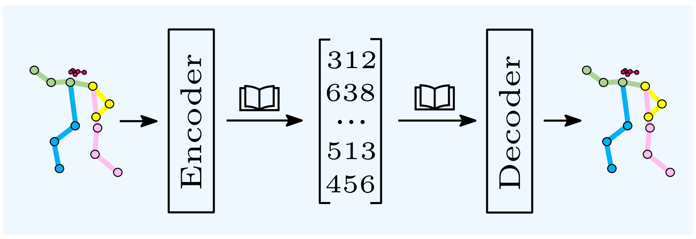


Figure 3.13: The process of learning the token representation of the pose. The codebook magnitude (between 256 and 2048) is large enough to cover sufficiently every possible pose. Retrieved from [12].

MotionBERT employs a dual-stage approach: unified pre-training followed by task-specific fine-tuning. Initially, a motion encoder is trained for the 2D-to-3D lifting task using DSTformer, which integrates spatial and temporal blocks with Multi-Head Self-Attention (MHSA). Spatial MHSA (S-MHSA) focuses on relationships among joints within the same time step, while Temporal MHSA (T-MHSA) models relationships across time steps. After pre-training, the encoder undergoes fine-tuning for tasks like 3D pose estimation, using 2D skeleton sequences. Experiments with MotionBERT using the MMPose framework are in Section 7.2.1.

3.5.9 Human Pose as Compositional Tokens

Geng et al. introduced a novel method to address the issue of occluded scenes with their publication Human Pose as Compositional Tokens (PCT) [12].

Human pose models using coordinate vectors or heatmap embeddings struggle with occlusions, reducing precision for non-visible joints. The PCT model improves accuracy in occluded environments by representing poses as compositional tokens, addressing the issue of independently treated body joints.

PCT uses a compositional encoder to map poses into discrete tokens, each representing interdependent joints, reducing reconstruction errors and improving efficiency. The pose is described as M discrete tokens, each indexing a codebook entry for various joint configurations. These tokens are then decoded back into the skeletal representation without additional post-processing as briefly outlined in Figure 3.13.

The PCT method, developed using COCO and MPII datasets, was also evaluated on occluded image datasets such as CrowdPose, OCHuman, and SyncOCC. Training involves joint learning of the encoder, codebook, and decoder to minimize reconstruction errors and optimize pose data representation. PCT casts pose estimation as a classification problem, predicting poses through token classification, supported by a learning algorithm that minimizes reconstruction loss and quantization error.

PCT excels in predicting human poses under various conditions, especially occlusions, demonstrating a high generalization across datasets. It outperformed models such as HRNet, UDP, and ViTPose in occluded tasks with the AP metric and in MPII using PCKh@0.5. In the COCO dataset, PCT matched accuracy with faster inference speed for the largest model (Swin-Huge backbone) and surpassed others with the smallest variant (Swin-Base backbone).

Chapter 4

Mobile Development and Server-Client Architecture

This chapter focuses on the fundamentals of mobile development and server-client architecture, as numerous mobile apps interact with servers through APIs, which plays a crucial role in the entire process. The general approaches and methodologies used in mobile development are discussed. Important aspects of a mobile development cover user interface, storage and databases on mobile platforms, networking and connectivity, and others like performance or accessibility. In the second part of this chapter, computer vision on edge devices, such as a smartphone is examined. The model compression and optimization methods are presented at the end of the chapter.

4.1 Fundamentals of Mobile Application Development

Mobile Applications [52] by Randhawa is a recent book published in 2022 discussing various mobile development aspects. All mobile applications address certain usage needs and thus must solve similar issues like granting permissions, storing data, calling API calls to the Internet, responsiveness, etc. Relevant topics are discussed in this section, more complex or various areas have a dedicated subsection for better clarity.

General aspects of mobile development, common with other software, are discussed in the first chapter [52]. Broad problem statements, often epics, need clarification. These epics or user stories, too large for a sprint, define the product from user roles' perspectives. Each user story has acceptance criteria in Gherkin's Given-When-Then (GWT) syntax for test automation. In GUI-driven applications, user stories and criteria can be supplemented with wireframes, screen mockups, or storyboards. Wireframes follow user stories, but UX design is completed before development. An MVP (Minimal Viable Product) comes from user stories, acceptance criteria, and wireframes.

On the requirements side, it can be distinguished between two basic categories: functional and nonfunctional requirements. Functional requirements can be defined, for example, as user stories and UML Use Case Diagrams. Non-functional requirements examples are various software quality attributes, speed, or other performance metrics, etc. Their impact on software architecture and design is significant and must be studied in the very beginning of the project.

The goal of software development is customer satisfaction. Acceptance Test Driven Development (ATDD) is effective, involving initially failing tests and developing code until they pass. Behavior-Driven Development (BDD), ideal for GUI apps, ensures that tests define features from the user's view. Tools like Gherkin in BDD help create business language tests linked to technical implementation, leading to focused refactoring.

Integration challenges occur when later combined modules are developed independently. Frequent code submissions, builds, and regression testing can mitigate these issues. However, resource-intensive automation streamlines the process, allowing developers to focus on regular code check-ins and simplifying integration and error detection. These practices, aimed at maintainability, integration, and delivery, are also useful for single-person projects.

4.1.1 User Interface and Experience

As Randhawa [52] noted, mobile apps generally include a GUI, data storage, and network connectivity. This section combines UI and UX concepts, addresses limitations like small screens and touch capabilities, and introduces mobile-specific UI controls.

Smartphone GUI frameworks offer a rich array of UI objects for easy integration. In Android, UI elements come from the `View` class in the `android.view` package and are displayed on a `Activity` from the `android.app` package. The `android.widget` package provides a wide range of GUI objects, enhanced by updates. Developers arrange these elements using layouts such as constraint, relative, linear, and grid, all descendants of `ViewGroup`, which aggregates multiple `View` objects. Complex interfaces can be created with nested `ViewGroups`. User interactions trigger events handled by listeners in the `View`. Navigation between *Activities* is managed by `Intent` objects.

Android UI can be designed with two different approaches: XML layouts or Jetpack Compose. According to the documentation [20], Jetpack Compose is now the recommended way to implement the Android user interface. However, since XML layouts have a huge legacy foundation, and it is always important to understand the history and context of frameworks, the outline of XML layouts follows.

Android XML layout documentation [21] describes XML layouts as essential for UI creation, defining objects, and organizing `Views`. The XML layout files in `res/layout` describe the GUI structure. `ViewGroup` subclasses like `LinearLayout`, `RelativeLayout`, and `ConstraintLayout` offer flexible positioning. `findViewById()` references `Views` using unique IDs (`@+id/name`) or Android resource IDs (`@android:id/name`). The layout system supports various screen sizes and densities, ensuring compatibility. Tools like `AutoCompleteTextView`, `DatePicker`, and `TimePicker` enhance user interaction, ensuring consistent UX. An example of XML layout is shown in Listing 1, with the final UI in Figure 4.1.

Jetpack Compose, announced in 2019, is now recommended for the development of Android user interface. Like Swift and React Native, it is fully declarative, meaning the UI is written in Kotlin and components are structured in modules, composed into larger components. An example is shown in Listing 2, with the final UI viewable in the Android Studio Compose Preview tool, and the result in Figure 4.2.

Listing 1 Example of XML layout definition. Retrieved from [21].

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:orientation="vertical" >
    <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello, I am a Button" />
</LinearLayout>
```

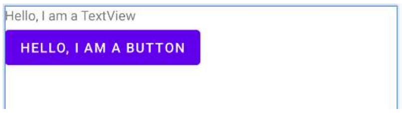


Figure 4.1: Rendered UI from XML layout Listing 1

It is important to understand that the XML layout in Android development serves solely as a declaration of the UI widget hierarchy (tree). The XML file defines the structure and relationships between UI components, but does not handle instantiation or event handling. For instance, to create instances of these UI components, assign `onClick` or other `on<Event>` handlers, and manage additional behaviors, you must use a custom `View` class. This separation ensures that the XML layout remains a pure definition of the UI structure, while the `View` class contains the logic and behavior associated with those UI elements.

In contrast, Jetpack Compose offers a different approach by allowing the integration of UI declarations and event handlers within the same file, or even inline using anonymous functions. This means that both the UI structure and its behavior can be defined in one place. However, this approach shifts the responsibility of maintaining proper code separation to the developer. It is up to the developer to ensure that the code remains organized and manageable, adhering to the best practices for the separation of concerns.

4.1.2 Storing Data

This section explores the various methodologies and technologies for data storage in mobile applications and the specific options on Android devices. Different storage options such as local storage and cache, cloud storage, and database systems such as the SQLite database are covered. The focus is on their applicability for diverse data types (data blobs, images) and application needs.

As Randhawa states [52], smartphones, despite having less disk capacity, offer various storage options. Data can be stored remotely, though it requires network connectivity. Local storage is possible via the file system using familiar APIs. Android uses `SharedPreferences` for key-value pairs, similar to Windows' registry. It also includes SQLite for structured data, with support for third-party SQL or NoSQL databases after installation.

Listing 2 Example of Jetpack Compose UI declaration. Retrieved from [20], edited to show the Android Studio Compose Preview tool.

```kotlin
@Preview
@Composable
fun NamePicker(
    header: String="Colleagues",
    names: List<String> = listOf("Petr", "Martin"),
    onNameClicked: (String) -> Unit = {}
) {
    Column {
        Text(header, style = MaterialTheme.typography.bodyLarge)
        Divider()

        LazyColumn {
            items(names) { name ->
                NamePickerItem(name, onNameClicked)
            }
        }
    }
}

@Composable
private fun NamePickerItem(name: String, onClicked: (String) -> Unit) {
    Text(name, Modifier.clickable(onClick = { onClicked(name) }))
}
```

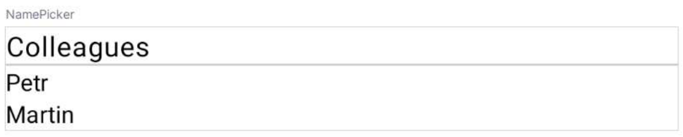


Figure 4.2: Rendered UI from Jetpack Compose Listing 2

Android's `SharedPreferences` stores small collections of key-value pairs. Access it via `getPreferences()` for single files or `getSharedPreferences()` for multiple files. Use methods like `putString()` and save with `apply()` or `commit()`. SharedPreferences files can be private or shared, and changes are in-memory with `apply()` or saved to disk with `commit()`. It supports basic data types and complex structures such as strings, requiring serialization for non-basic types. `SharedPreferencesChangeListener` detects modifications. An example of saving a URL is in Listing 3.

External storage is accessed via the `getExternalFilesDir()` function, and shared storage then via the `getExternalStoragePublicDirectory()` function. Internal storage is private, but limited. User permissions for external storage are required and can be granted statically in the Manifest or dynamically at runtime. The `Context` for creating objects is obtained through various methods. Use Android's FileProvider for secure file sharing. Examples are shown in Listings 4 and 5.

Database systems improve data management over file systems with efficient structures and finer data access. Types include RDBMS (Oracle, SQLServer), ODBMS[1] (ObjectStore,

[1]ODBMS = Object database management system

Listing 3 Example of process of saving preferences. Retrieved from [52]. The common practice of using `et` as an abbreviation for *edit text* can be seen.

```
SharedPreferences settings = getSharedPreferences("ProgramSettings", 0);
SharedPreferences.Editor editor = settings.edit();
String url = ((EditText) fndViewById(R.id.etURL)).getText().toString();
editor.putString("URL", url);
editor.commit();
```

Listing 4 Example of definition of content provider in the Android Manifest file. Retrieved from [52], updated `android:name` to the most recent package.

```
<provider
    android:name="androidx.core.content.FileProvider"
    android:authorities="${applicationId}.fileprovider"
    android:exported="false"
    android:grantUriPermissions="true">
    <meta-data
        android:name="android.support.FILE_PROVIDER_PATHS"
        android:resource="@xml/file_paths" />
</provider>
```

Gemstone), and ERDBMS[2] (UniSQL). RDBMS uses SQL for tables, ODBMS stores objects in collections, and ERDBMS combines both for 3D object storage. SQLite, common on smartphones, supports relational models with SQL, handling data in tables. The `SQLiteDatabase` class in Android provides database operations, supporting types like null, integer, real, text, and blob. `SQLiteDatabase` class shows database and table creation, ACID compliance, and role/permission management.

The Data Access Object (DAO) pattern is utilized to abstract and encapsulate all interactions with the database, providing a clear and consistent API for data access. This pattern is implemented effectively using, for example, the Room persistence library, which serves as an abstraction layer over SQLite. Room simplifies database access in Android applications by providing a set of annotations to define SQL queries and database operations. The DAO pattern is central to Room's architecture, ensuring a separation between the application's data layer and the business logic.

In Room, a DAO is an interface or abstract class annotated with `@Dao`, which includes methods that interact with the database. These methods are annotated with special annotations such as `@Insert`, `@Update`, `@Delete`, and `@Query` to map them to SQL operations. This approach allows Room to generate the necessary code for these operations at compile-time, ensuring type safety and reducing boilerplate code.

The use of DAOs with Room offers several features, such as compiling-time verification of SQL queries, the guarantee of type safety by matching the parameter of the method and the return types with the database schema, or the integration with the lifecycle components of Android, such as ViewModel and LiveData.

[2]ERDBMS = Extended relational database management system

Listing 5 Example of usage of content provider in Java language. Retrieved from [52].

```
Uri photoURI = FileProvider.getUriForFile(
    this,
    "com.example.photogallery.fleprovider",
    photoFile
);
```

Listing 6 Example of GET request with OkHttp library. Retrieved from [60].

```
OkHttpClient client = new OkHttpClient();

String run(String url) throws IOException {
  Request request = new Request.Builder()
      .url(url)
      .build();

  try (Response response = client.newCall(request).execute()) {
    return response.body().string();
  }
}
```

4.1.3 Network Communication and API Integration

The essentials of network communication in mobile applications, include the use of APIs for data exchange between the app and remote servers. Topics such as RESTful services, data parsing, and handling asynchronous network calls are explored. Security aspects such as encryption and secure data transmission are addressed in Subsection 4.1.4.

Per Mobile Applications [52], smartphone networking includes cellular, WiFi, Bluetooth, and NFC interfaces. Communication is similar to other devices, starting with a DNS query via UDP, followed by a TCP three-way handshake to establish a connection. HTTP requests and responses are exchanged, ending with a FIN packet exchange. UDP is for small, best-effort data exchanges like DNS, while TCP ensures reliable, ordered delivery for HTTP traffic. HTTP supports methods like GET, POST, PUT, and DELETE, with responses including status codes, headers, and a message body.

Android platform provides APIs such as `java.net.InetAddress` for domain name resolution and `java.net.HttpUrlConnection` for web server communication, supporting GET and POST requests. `HttpUrlConnection` handles tasks like uploading images or retrieving data, while `MultipartEntity` transfers larger files. Web servers use cookies to maintain state and identify returning users. Android also provides abstraction libraries for network communication.

Although `HttpUrlConnection` is sufficient for communication and data transfer between servers and mobile clients, developers are required to write a lot of code. Thus, various libraries with higher level of abstraction were developed, such as OkHttp [60], Volley [18], or Retrofit [61]. These advanced libraries provide more advanced features such as secure connection protocols (HTTPS, TLS) with certificate handling, data dispatching, event listeners, scheduling requests, or even caching options. Example GET request using libraries OkHttp, Volley, and Retrofit can be seen in Listings 6, 7, and 8 respectively.

Listing 7 Example of GET request with Volley library. Retrieved from [18].

```
RequestQueue queue = Volley.newRequestQueue(this);
String url = "https://www.google.com";

StringRequest stringRequest = new StringRequest(Request.Method.GET, url,
            new Response.Listener<String>() {
    @Override
    public void onResponse(String response) {
        textView.setText("Response is: " + response.substring(0,500));
    }
}, new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
        textView.setText("That didn't work!");
    }
});

queue.add(stringRequest);
```

Listing 8 Example of GET request with Retrofit library. Retrieved from [61].

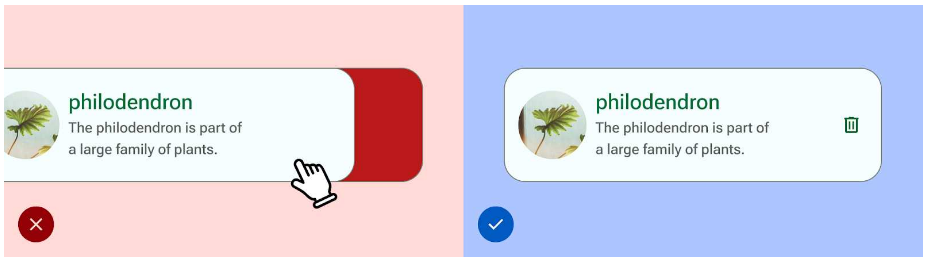
```
    public interface GitHubService {
      @GET("users/{user}/repos")
      Call<List<Repo>> listRepos(@Path("user") String user);
    }

    Retrofit retrofit = new Retrofit.Builder()
        .baseUrl("https://api.github.com/")
        .build();

    GitHubService service = retrofit.create(GitHubService.class);

    Call<List<Repo>> repos = service.listRepos("octocat");
```

4.1.4 Additional subjects: Security, Accessibility, and Performance

This section examines security and privacy in the development of mobile applications, focusing on data encryption, authentication, and compliance with privacy laws. Accessibility for disabled users and the performance of the application are also discussed, with more information on performance in Section 4.2.

The security of Android devices, well studied since the rise of smartphones, shows that data protection is user-centered but often inconvenient.

Zaidi et al. [10] in 2016 identified threats such as insufficient API management leading to security breaches. They created a comprehensive hierarchical structure of possible threats shown in *Fig 1* in Android Security publication [10]. In 2012, defenses against physical threats were considered inadequate, with suggestions for improving lock screen systems to increase security and user confidence. High-privilege APIs are targets for attacks, and wireless networks are vulnerable to data interception. Encrypting communications can secure Wi-Fi, Bluetooth, cellular, and GPS data. User awareness is key, as malicious apps can disguise themselves. Backdoors exploit system bugs and insufficient authentication,


philodendron
The philodendron is part of a large family of plants.
philodendron
The philodendron is part of a large family of plants.
Figure 4.3: Example of non-compliant and compliant solution for accessibility actions. It is advised to not implement action available solely via gestures or a single type of action. The right side shows the design where the button serves as the accessibility action option.

while viruses and worms spread through untrusted sources. Malware, posing as legitimate applications, threatens data security.

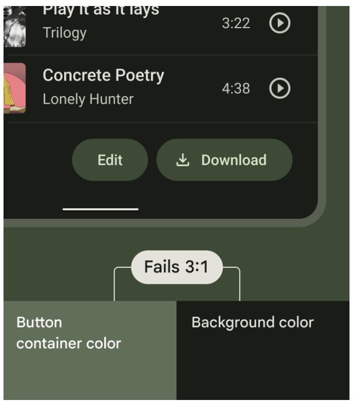
Some threats are prevented by Android security patches, but new hazards regularly emerge. Mayrhofer et al. [44] examine the Android security model in the AOSP. Android requires explicit user consent for security or privacy actions and includes a global user base and diverse devices from many OEMs. Google-certified devices undergo firmware checks; AOSP-based devices do not. The Compatibility Test Suite (CTS) ensures app compatibility, and apps can be developed in any language but must use Java APIs, necessitating runtime security checks.

Mobile devices face unique threats due to portability, frequent connections to untrusted networks, and access to privacy-sensitive data. A layered threat model categorizes risks by adversary capability. One major threat category is untrusted code, which Android allows with user consent. This leads to potential attacks like abusing OS or app APIs, executing untrusted web code without consent, mimicking system interfaces to trick users, reading content from other apps or the system, injecting input events into other apps, and exploiting OS bugs. Hardware-level attacks are becoming more popular, necessitating additional defenses, though often with performance trade-offs. Mayrhofer et al. warn that inexperienced developers can unintentionally introduce security issues, emphasizing the need for security awareness.

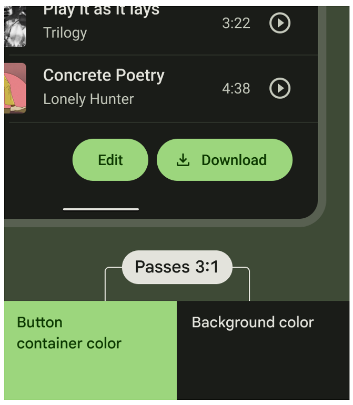
Inclusive design incorporates features for various user needs, including those with disabilities, and should be considered from the start. The WHO reports that 16% of the global population is disabled [69], and only 20% of them are sport-active [1], amounting to 3.2% of the world's population. Comparatively, 22% of non-disabled people are sport-active [25], leading to 18.5% of the population. Thus, 15% of sport-active people are disabled, highlighting the importance of accessibility in mobile sports apps.

The Android platform supports users with various disabilities with several recommendations [19]: use scalable pixels (sp) for adjustable fonts, avoid small fonts (below 12 sp), and maintain a text-background contrast ratio of at least 1:4.5. Android provides guides for accessible design and implementation of color, sound and audio, with the Material3 design system emphasizing an accessible-compliant color system.

Figure 4.3 shows an interface with gesture actions that lack alternative actions for accessibility and a suitable solution. Figure 4.4a shows low contrast that does not meet the 1:3 ratio for large texts, while Figure 4.4b meets the ratio, which makes button colors appropriate.



(a) Low contrast ratio.



(b) Proper contrast ratio.

Figure 4.4: Examples of sufficient and not sufficient contrast ratios. Retrieved from [17].

Ajayi et al. [47] emphasize that the performance of mobile apps is crucial to user satisfaction. Though average use cases do not require deep optimization, performance metrics like app load time, memory, CPU, battery, and network usage are vital, especially for non-native frameworks, which can be inherently non-optimal. Web-based apps, for example, can have start times up to 20 times slower.

Android documentation [22] highlights key performance issues:

- Scroll Jank: Visual stutters during scrolling due to slow content rendering. Aim for at least 90Hz refresh rates.
- Startup Latency: Delay from app launch to content display. Cold starts should be under 500ms, with analysis on the 95th and 99th percentiles. Warm starts should be faster. Avoid lengthy IPCs and unnecessary I/O to improve startup time.
- Power Inefficiency: Battery drain due to excessive memory allocations. Modern ART optimizes this, but excessive allocations, especially in loops, can hinder performance. Balance maintainability and resource efficiency.

Android Studio's profiling tools measure CPU and memory usage, with more complex metrics available through the Microbenchmark and Macrobenchmark libraries. Figures 4.5 and 4.6 show profiling session examples and frame-tracing benchmark analysis. Macrobenchmark tests larger use cases and demanding UI tasks, while Microbenchmark is for lighter tasks or smaller code blocks.

4.2 Computer Vision on Edge Devices

This section explains the topic of computer vision models in resource-constrained environments for optimal performance. Edge computing shifts processing from centralized servers to local devices, such as smartphones, offering reduced network reliance, lower latency, and improved privacy by processing data locally.

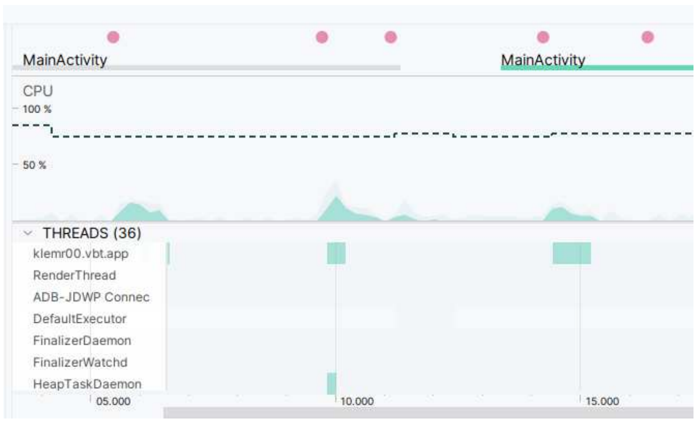
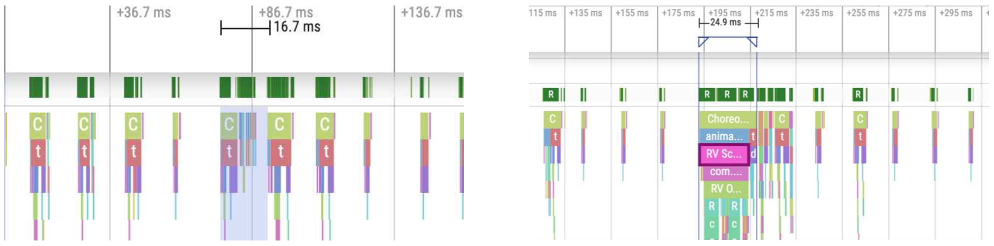


Figure 4.5: Profiling session in Android Studio. Red dots are touch interactions of the user. The rise in CPU usage can be seen in the events of user interaction.



(a) Highlighted frame rendering took a bit longer than for other frames, but still fitted in the 16.7 ms maximum window, so it is not considered as an issue.

(b) Rendering took too long and exceeded the maximum of 16.7 ms, this is an example of a janky frame that occur for example in the janky scrolling issue.

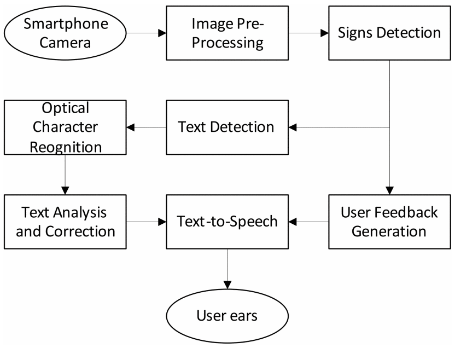
Figure 4.6: Visualizations of frame tracing from Macrobenchmark profiling. The 16.7 ms interval mentioned as the maximum rendering window equals the very minimal 60 Hz refresh rate. To achieve the recommended refresh rate of 90 Hz, it must be 11.1 ms. Both images were retrieved from [22], cropped.

Key considerations include hardware and software requirements, power consumption, latency, maintenance, and the need for continuous learning. It also focuses on fine-tuning and optimizing models for edge deployment, ensuring that they are lightweight and fast. Techniques such as model pruning, quantization, and knowledge distillation to reduce model size and complexity are explored in Section 4.2.1.

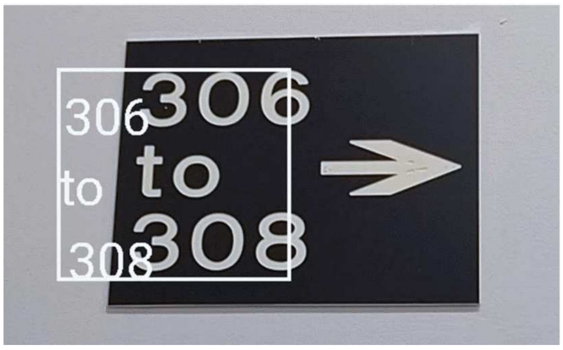
Rohith, Ajeet Sunil, and Mohana [53] predict 50% of industrial data will be processed by edge computing by 2022, with over 75.4 billion IoT devices by 2025. Edge computing reduces latency and ensures privacy, with 4.5 billion edge devices already in use for real-time processing.

Sinha's 2023 prognosis [58] forecasts 22.2 billion IoT devices by 2025 and 29.7 billion by 2027, showing continued exponential growth. Edge computing processes data close to the source, enhancing real-time performance and reducing bandwidth and storage needs, optimizing mobile computing for local data.

Key features of edge computing include ultra-low latency, beneficial for immediate data processing in applications like industrial IoT and self-driving cars. Processing data at the



(a) Scheme of the system designed by Jiang et al. [34].



(b) Example output of the system designed by Jiang et al. [34].

Figure 4.7: Computer Vision and Text Recognition for Assisting Visually Impaired People using Android Smartphone by Jiang et al. [34].

edge minimizes delays, enhancing response times. Edge computing also offers larger bandwidth by managing data locally, reducing the need for continuous cloud transfers, which is crucial for data-heavy IoT technologies. Additionally, it improves privacy, security, and cost-effectiveness by decentralizing data processing, adhering to data sovereignty standards, and reducing data traffic costs. Edge devices in IoT include:

- IoT gates: Handle device security, data processing, and transmit crucial information to the cloud.
- Edge sensors: Send signals when necessary, reducing unnecessary cloud data traffic.
- Routers: Transfer data packets and may have computing capabilities for edge applications.

In their research, Gupta et al. [27] highlight the utility of smartphones as edge devices, exemplified by their use in agricultural machinery health monitoring. Their study demonstrates how smartphones can detect noise around an oil filter to assess its condition as depicted in Figure 3 [27].

A natural use of a smartphone as an edge device in computer vision is basic camera use. Jiang et al. [34] discuss complex computer vision applications, such as helping visually impaired people navigate their environment using enhanced photo capture, OCR, and TTS. Their system uses the smartphone's camera to find signs or tables, detects text, and uses OCR to convert it to speech with TTS. The result is shown in Figure 4.7b.

Running OCR on edge devices requires significant hardware resources. For example, the TensorFlow Lite OCR model based on the Keras CRNN architecture combines RNN and DCNN layers. The CRNN, presented in 2015 [56], optimized the RNNs for a smaller model size of 8.3 million parameters, using approximately 33 MB of RAM.

4.2.1 Model Compression for Android devices

In deploying machine learning models on Android devices, model compression is necessary along with the transformation to the appropriate back-end. Although not used in Chapter 5 and 6, it is relevant for future development 7.9 and experiments with human pose estimation on mobile devices. Such experiments are described in Section 7.4.

Model compression techniques such as pruning, distillation, and knowledge transfer enhance the performance and feasibility of complex models on Android and edge devices. Pruning, as described in [45], reduces the size of the model by removing less significant neurons or weights, making it more lightweight without a major loss of precision. This is vital for the efficiency of Android devices.

Pruning neural networks dates back to the early 1990s, using second-order Taylor expansion to select parameters for deletion, improving generalization. However, it can be more memory and computationally demanding than standard fine-tuning because of the partial or complete solution of the Hessian matrix.

A recent method for CNNs is structured pruning by Anwar et al. [3], which uses a particle filtering approach. Naive pruning, which zeros near-zero values, reduces the size of the model but is unsuitable for parallel computing and hardware-optimized algorithms. Sparse representation schemes can address this, but introduce computational overhead. Anwar et al. propose structured pruning to solve this issue; vivid illustration can be found in their paper.

Molchanov et al. present a Taylor expansion-based criterion for pruning, optimizing by finding parameters with near-flat gradients, indicating low significance in convolutional layers. Their iterative method allows fine-tuning and re-evaluation, achieving a 200% speedup with only a 2% accuracy loss.

Molchanov et al. [45] discuss other methods such as combining parameters with correlated weights, reducing precision, and tensor decomposition. These require separate training or fine-tuning but can offer additional speedups when combined with their proposed method. Knowledge Transfer, or Transfer Learning, as discovered by Pan and Yang [48], involves transferring information from one model to another. This broad category of optimizations includes many specialized methods, such as knowledge distillation.

Hinton et al. [30] introduce knowledge distillation as the process of transferring knowledge from a larger, more complex model (teacher) to a smaller, more efficient model (student). It allows the distilled model to perform similarly to the larger model, but with less computational demand. They considered softmax the last output layer with temperature T:

$$q_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)} \tag{4.1}$$

The higher the T, the softer the output and vice versa. The initial value also used for inference is $T = 1$. By softening the output or softening the probability, the model is meant to be less certain about the output.

Knowledge distillation trains a smaller model using outputs from a larger model at a high softmax temperature. The distilled model learns from both the soft target distribution and the correct labels, using a weighted average of cross-entropy with soft targets and correct labels at $T = 1$, typically giving lower weight to the latter. Scale the soft target gradients by $\frac{1}{T^2}$ and multiply by T^2 to balance the contributions. In Automatic Speech Recognition, a distilled model can match the accuracy of an ensemble of 10 models.

Modern model quantization approaches are discussed by Nagel et al. [46]. Model quantization reduces the computational complexity and memory footprint of machine learning models, making them more efficient for resource-constrained devices like smartphones, embedded systems, and edge devices. It converts model parameters from high-precision (e.g., 32-bit floating-point) to lower-precision (e.g., 16-bit floating-point, 8-bit integer) representations.

There are two main approaches to quantization: post-training quantization (PTQ) and quantization-aware training (QAT). Post-training quantization converts a pre-trained model without additional training data, using methods like full integer quantization (8-bit integers) and float16 quantization (16-bit floating-point). In contrast, QAT trains the model with simulated lower-precision arithmetic, allowing it to adapt to the quantized representation, which typically results in better accuracy for complex models.

The quantization process involves calibration, quantization, and evaluation. Calibration uses a representative (calibration) dataset to determine the range of values for activations and weights, helping compute the scale and zero-point for converting floating-point values to integers. The quantization step applies these values to transform the model's weights and activations to lower-precision formats. Finally, the quantized model is evaluated to ensure acceptable performance.

Despite its advantages, quantization can lead to accuracy loss, especially in complex models or tasks This can be mitigated through QAT. For example BERT-base models loss a lot of accurac using PTQ, but fit under 1% loss of GLUE sscore using QAT. The effectiveness of quantization also depends on hardware support for lower-precision operations. Models with a large dynamic range of activations and weights may experience more significant accuracy loss when quantized.

In the Android ecosystem, libraries such as TensorFlow Lite[3] and PyTorch Mobile[4] provide options to optimize models for mobile devices or edge devices, in general. TensorFlow Lite offers tools for model optimization as well as support for pruning and knowledge distillation. PyTorch Mobile is specific for smartphones and also supports model optimization. Its ability to integrate with Android Studio and the Android Neural Networks API (NNAPI) makes it a practical choice for deploying compressed models on Android platforms.

Backends such as PyTorch Mobile[5], ncnn[6], and ONNX[7] are essential for deploying and executing machine learning models on mobile devices. PyTorch, developed by Facebook's AI Research lab, is an open-source library offering a flexible framework for building and training neural networks. It supports dynamic computation graphs and is optimized for performance on both CPUs and GPUs, suitable for research and production.

ONNX (Open Neural Network Exchange) is an open format for representing machine learning models, enabling transfer between frameworks like PyTorch and TensorFlow. ONNX Runtime is a high-performance inference engine designed to execute ONNX models efficiently on various hardware platforms. ncnn, developed by Tencent, is a high-performance neural network inference framework optimized for mobile platforms. ncnn's integration with Vulkan API further boosts performance by leveraging GPU compute capabilities for accelerated neural network inference on smartphones and embedded devices.

[3] https://www.tensorflow.org/lite
[4] https://pytorch.org/mobile/home/
[5] `https://pytorch.org/mobile/home/`
[6] `https://github.com/Tencent/ncnn`
[7] `https://onnx.ai/`

The Vulkan[8] API, developed by the Khronos Group, is a low-overhead, cross-platform 3D graphics and compute API. It provides high-efficiency, high-performance access to modern GPUs across PCs, consoles, mobile phones, and embedded platforms. Vulkan's design reduces CPU usage and distributes work among multiple CPU cores, ideal for performance-critical applications like gaming, real-time graphics, or machine-learning.

In machine learning, Vulkan's compute capabilities can accelerate neural network inference. Compatibility with inference frameworks like ONNX or ncnn is achieved through Vulkan's compute shaders, enabling parallel computation on the GPU. ncnn has integrated Vulkan to optimize neural network inference on mobile devices, leveraging its efficient GPU compute capabilities for enhanced performance. This integration allows ONNX models to be executed efficiently on Vulkan-supported hardware, enabling high-performance deployment of complex AI models across various devices. Vulkan and ncnn were used in experiments with human pose estimation models on mobile devices described in Section 7.4.

[8] https://www.vulkan.org/

Chapter 5

Design of Client-Server Application for Strength Sports Training

The approach to creating the solution was based on an iterative development correspondingly as defined by the assignment. For that reason, it is difficult to describe the design as one all-inclusive chapter. We decided to present the final design. When certain feature or decision is discussed, it is going to be mentioned that it was the result of iterative development if applicable.

In the iterative development, multiple user feedback sessions were conducted, and the notes, comments, and remarks from users were reviewed. Indication of whether user's remark is more prone to be a *must-have* or rather to be a *nice-to-have* was made for each user note. Methodology of user evaluation is described in Appendix C. Records and transcriptions of user evaluation and testing sessions are available as text files in the `App/user_testing/live_sessions/` folder.

Regarding the chapter's structure, first the high-level design is outlined, followed by the design of the server's API. Then preliminary experiments are presented to support the decisions about the selected human pose estimation model. The integration of human pose estimation, post-processing, and the computation of metrics are discussed next. Then, the Android application client is presented. In the end, challenges, interesting improvements, and other changes done during iterative development are discussed.

5.1 High-Level Architecture

First, the architecture of the system is described at the high level, explaining fundamental aspects of the whole application. To simplify the naming, by *application*, the whole system is meant. The *client* is Android mobile application and *server* is Python server which exposes the API and performs the model inference.

Since majority of models with which the preliminary experiments were performed do not provide the pretrained model converted to the mobile backend (e.g. ncnn) nor the frame-

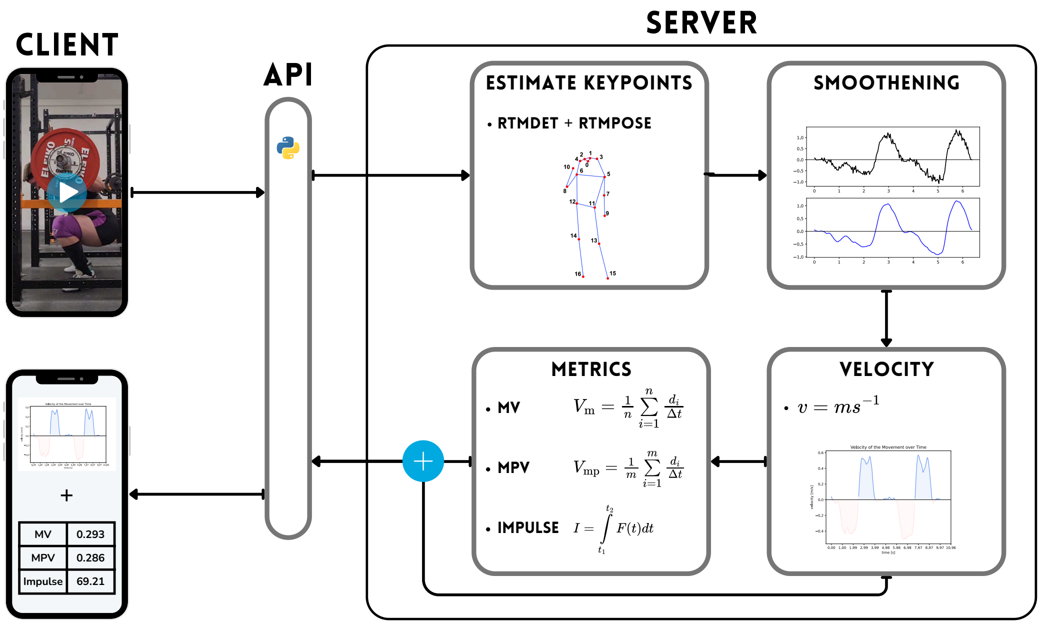

CLIENT
API
SERVER
ESTIMATE KEYPOINTS
• RTMDET + RTMPOSE
SMOOTHENING
METRICS
• MV $V_m = \frac{1}{n}\sum_{i=1}^{n}\frac{d_i}{\Delta t}$
• MPV $V_{mp} = \frac{1}{m}\sum_{i=1}^{m}\frac{d_i}{\Delta t}$
• IMPULSE $I = \int_{t_1}^{t_2} F(t)dt$
VELOCITY
• $v = ms^{-1}$
+
MV 0.293
MPV 0.286
Impulse 69.21

Figure 5.1: Whole application architecture. Client sends the video via API to server. Inference of RTMPose is performed. Post-processing computes the velocities, smooths the data and calculates exercise performance metrics. This result is sent back to the client.

work for mobile integration, models were tested solely on desktop GPU or CPU. From this initial configuration of the development setup, client-server architecture organically arose. The final design of application's architecture can be seen in Figure 5.1.

5.2 Server API

REST API in Python language can be implemented in various frameworks, from heavier and more powerful Django, through all-around capable Flask up to lightweight FastAPI. Since the requirements on API of the developed application are rather simple than complex, lightweight FastAPI was chosen.

We want to emphasize that server is used only for processing and it is not storing any persistent data. All the data are stored only and only on client using local storage options. For that reason, there is no need for a complex solution for the API.

First, we propose supplementary endpoint `/healthcheck` for testing purposes. This endpoint can be used for monitoring and observability purposes.

Endpoints for HPE inference should accept video with possibility of accepting another parameters or data needed for required processing and optional features.

The core endpoint is `/graph`. In that case, server perform the HPE inference using the RTMPose model in MMPose framework. All following calculations are also computed on the server and the complete result is returned back to the client.

Additional usage can be offloading the postprocessing on the client. For that purpose we propose a second keypoint `/keypoints`. In this case server only do the HPE inference and returns predicted keypoints for each frame in a form of a JSON object. The postprocessing, velocity, and metrics calculations is then left on the client. This can be beneficial for scenarios where the client has implemented native interactive graph visualization or similar features. We concluded that to satisfy the minimal requirement, only one endpoint – `/graph` is necessary.

5.3 Preliminary Experiments with Human Pose Estimation Models

An experiment to verify the human pose estimation model works consistently across different mobile devices was performed. The selected devices are Samsung Galaxy Z Flip 4 and Realme 10. Their relevant specification can be seen in Appendix D. In Figure 5.2 can be seen a comparison of two repetitions of squat captured on the devices. Data in attached medium related to the preliminary experiments are denoted with `preliminary` prefix.

In following sections, results from preliminary experimsnts with HPE models are presetned. Those models were examined before the dicision which HPE models will be used. After the decision, research of state-of-the-art models continued. Selected HPE models were described in Section 3.5 and experiments with them are presented in Appendix F.

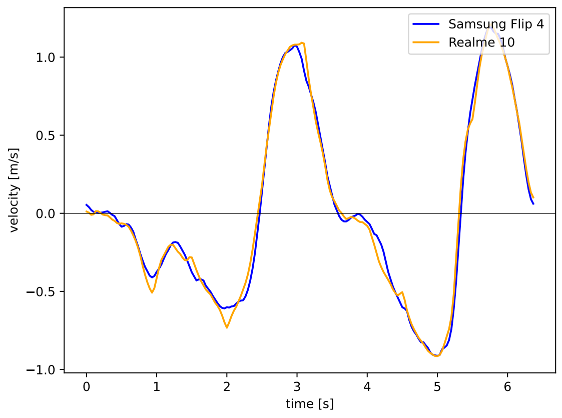
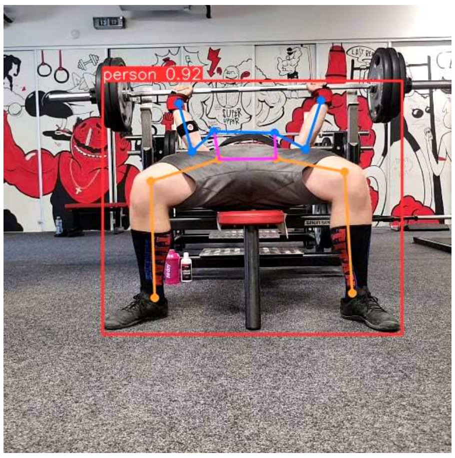


Figure 5.2: Comparison of velocities computed in preliminary experiment. The mean absolute error is 0.04 m/s. The used model was RTMPose-m.

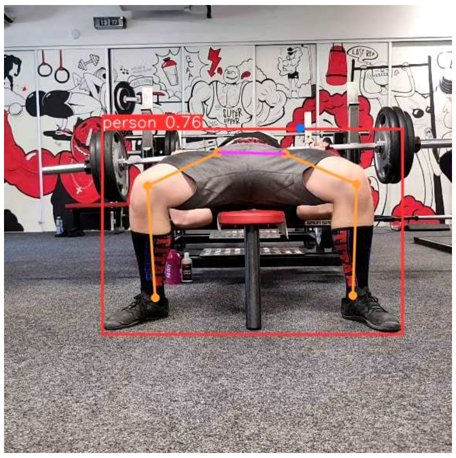
5.3.1 YOLOv8

The very first examined model was YOLOv8. Inference was performed with code cloned from official Ultralytics repository[1]. Keypoint prediction for front views (such as shown in Figure 5.3c and 5.3d) are satisfactory. Complex benchpress scene is recognized at a level acceptable for exploratory experiments only in a top position where more body parts (including tracked wrists) are visible. Bottom position is estimated only partially and also it can be seen the detection model is not as sure with the bounding box prediction as in the top position.

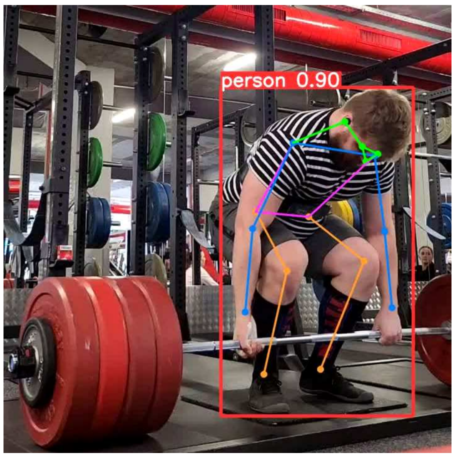
[1]https://github.com/ultralytics/ultralytics



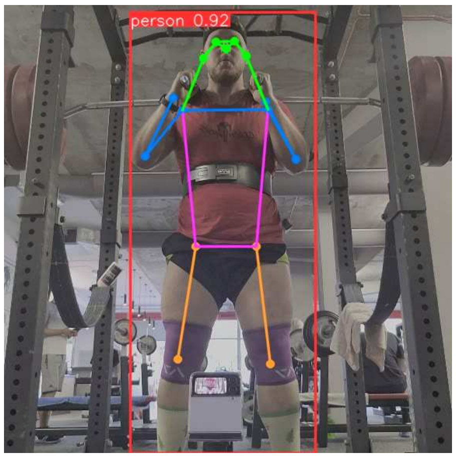
(a) Top position of benchpress. The upper body including wrist is recognized and annotated.



(b) Bottom position of benchpress. The vanishing whole upper body keypoints can be observed.

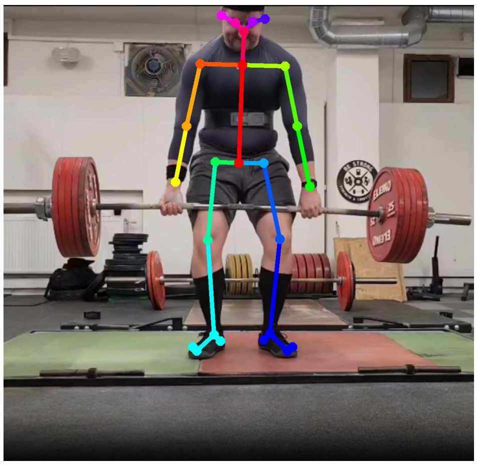


(c) Result for deadlift from lower medium distance camera position.

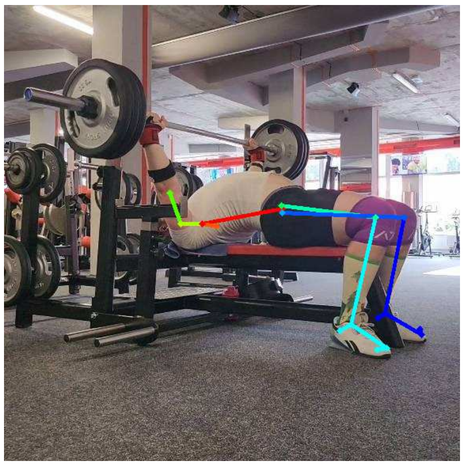


(d) Result for SSB from the lower closer camera position.

Figure 5.3: Examples of experiments with YOLOv8 inference. Benchpress was recorded from parallel medium distance camera position.

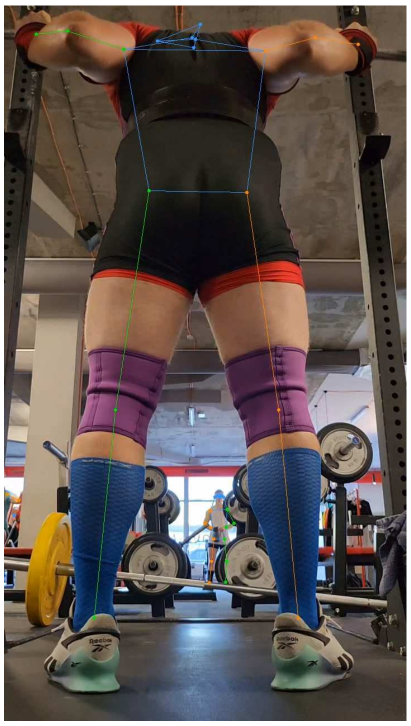


(a) Result for deadlift from parallel medium distance camera position.

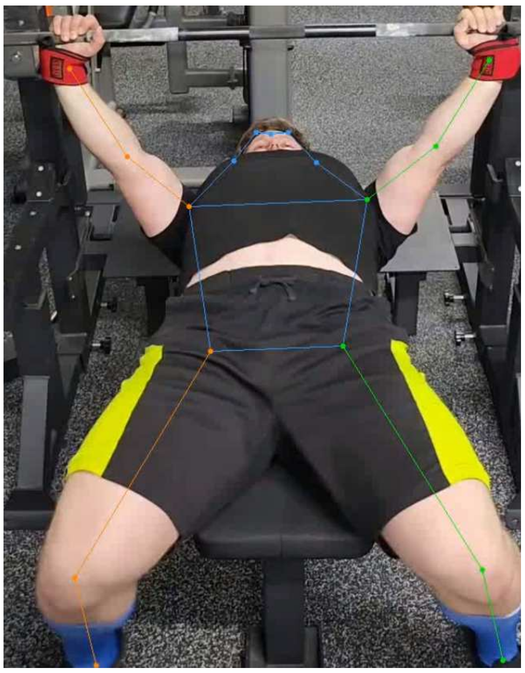


(b) Result for benchpress from parallel height, medium distance, 45° horizontal angle.

Figure 5.4: Examples of experiments with OpenPose inference.



(a) Result for squat from low close distance camera position.



(b) Result for benchpress from a higher height, close distance, −45° vertical angle.

Figure 5.5: Examples of experiments with RTMPose inference.

5.3.2 OpenPose

The second model we investigated was OpenPose [5]. Again, the official repository[2] was used to run the inference. The results can be seen in Figure 5.4a and 5.4b. As can be observed, the front view shows sufficient prediction accuracy, but the benchpress body position is not recognized properly.

5.3.3 RTMPose

RTMPose was inspected as a third choice. Based on the paper [35] it promises top-leading accuracy. The MMPose framework[3] provides functional scripts that run inference on the NVIDIA GPU.

5.4 Server-side Human Pose Estimation

Based on COCO instructions [41] and other papers [35, 5, 8] we can define the human pose estimation metrics notation as:

- AP or AP^{50-95}: AP at OKS $\in \langle 0.5, 0.95\rangle$ with 0.05 step (primary challenge metric)
- AP^{50} or $\text{AP}^{0.5}$: AP at OKS $= 0.50$ (loose metric)
- AP^{75} or $\text{AP}^{0.75}$: AP at OKS $= 0.75$ (strict metric)

For more details about OKS see Section 3.5.3 and COCO evaluation instructions [41].

Using this unified terminology, we can now compare and evaluate the models examined in preliminary experiments. Since RTMPose [35] uses only AP^{50-95}, we compared the models using that metric. YOLOv8 has the performance evaluation on their website[4] and RTMPose and OpenPose included evaluation in their papers. See the Table 5.1 bellow. Based on the performance comparison, which promised high accuracy and MMPose's convenient workflow, RTMPose was chosen (after consultation with the supervisor) as the model for the server-client architecture.

There is a potential for moving RTMPose model on Android with usage of MMDeploy. More about this is discussed in Section 7.4 where RTMPose and MoveNet deployments on Android devices are discussed in more detail.

5.4.1 Calibration

The keypoints returned from the RTMPose HPE inference are in a pixel space as $[x, y]$ coordinates from top left corner of the image. Calibration from the pixel space to the metric unit space is undoubtedly necessary.

[2] https://github.com/CMU-Perceptual-Computing-Lab/openpose
[3] https://github.com/open-mmlab/mmpose
[4] $\protect\protect\unhbox\voidb@x\hbox{AP}^{50-95}$

Table 5.1: Comparison of AP and AP^{50} (if applicable) metric for models participating in preliminary experiments.

	AP	AP^{50}
YOLOv8m-pose	65.0	88.8
YOLOv8x-pose	69.2	**90.2**
OpenPose	64.2	86.2
RTMPose-m	75.3	-
RTMPose-l	**76.3**	-

The reasonable user input could be user's height. Asking user for measurement of their ankle height is possible but asking for measurement of parallel lines touching the top of the head and crossing their eyes is practically impossible. The idea of calibration is using the highest and lowest keypoints to approximate the person's height. The COCO annotation style labels 17 keypoints. The most top ones are the eyes. The lowest ones are the ankles.

There is an issue with the eyes being the top most keypoints when the distance from the eyes to the top of the head is unknown. It applies analogously to the ankles. To address this issue we used an anthropometric study conducted on U.S. Army Personnel [23]. The data were measured on 11,357 people introducing 94 directly measured dimensions and 39 derived dimensions.

The objective is to find such scale factor which can transform the pixel dimensions to metric unit dimensions. We use the *lateral malleolus*[5] height and the difference between stature (regular human height) and eye height[6] to find the values which need to be subtracted from the person's height to match the erected position of athlete. Other option of top calibration keypoints could be ears and the equivalent anthropometric dimension *tragion*[7]. Additional details can be found on the OPEN website[8].

5.4.2 Post-processing of Raw Data

Data obtained from HPE models, such as RTMPose, have noise due to the flickering of keypoints even if the athlete is in a very still position. This is a characteristic for all thinkable solutions, including professional motion capture systems such as Qualisys 3.3.1.

The velocity calculation introduces more errors, which is given by the equation

$$v(t) = \frac{dx(t)}{dt}.$$

The conclusion is that the data should be smoothed, and we outline two levels of that procedure. First, a smoothing of the raw position data (coordinates per each frame) can be done. Second, the velocity of the movement can be smoothed after the calculation.

[5] *lateral malleolus* = height from ground to the outer side of the ankle
[6] Eye height is derived dimension, see Section D16 in [23]
[7] *tragion* = height from top of the ear entry to top of the head
[8] https://www.openlab.psu.edu/ansur2/

We decided that the smoothing of the final velocity graph is necessary to reduce visual clutter for the user. This requirement is backed by user evaluation sessions which can be found in files `user_testing/**/s_initial.txt|s_1.txt`.

Naive approach of smoothing using the moving average can have a negative impact on the peaks in the data which are the most important 3.1. Sensitive approach must be taken into account in the smoothing process.

5.4.3 Metrics

The velocity graph observation provides derived metrics such as fatigue drop or increased time under tension. Other metrics such as mean velocity (MV) or impulse provide a more comprehensive understanding. Since MV and other similar metrics are numeric values, it is faster for a user to compare two numbers than to analyze the entire velocity graph.

The main numeric metric will be MV. However, the mean propulsive velocity (MPV) is computed in a very similar way to MV. For this reason, we decided to include the MPV metric as well.

The impulse is considered by Valle [65] as the best overall performance metric. However, the impulse metric is very sensitive to the accuracy and consistency of the measurement, even after an effective computation design. Within a stable environment, for advanced athletes or coaches, it can be a very valuable insight into their training. For that reason, we expect the impulse to play a more experimental role.

You can see the proposed computation of the metrics in Equations 5.1, 5.2, and 5.3

$$V_{\mathrm{m}} = \frac{1}{n}\sum_{i=1}^{n} d_{c_i}, \tag{5.1}$$

where d_{c_i} is i-th concentric phase velocity.

$$V_{\mathrm{mp}} = \frac{1}{m}\sum_{i=1}^{m} d_{p_i}, \tag{5.2}$$

where d_{p_i} is i-th propulsive phase position difference.

$$I = m\sum_{i=1}^{m} \left(d_{p_i} - d_{p_{i-1}}\right), \tag{5.3}$$

where d_{p_i} is i-th propulsive phase velocity and m is used weight.

The metrics will be computed per repetition and per whole set. The presentation of computed values to the user will be done using numerical values in a visual container.

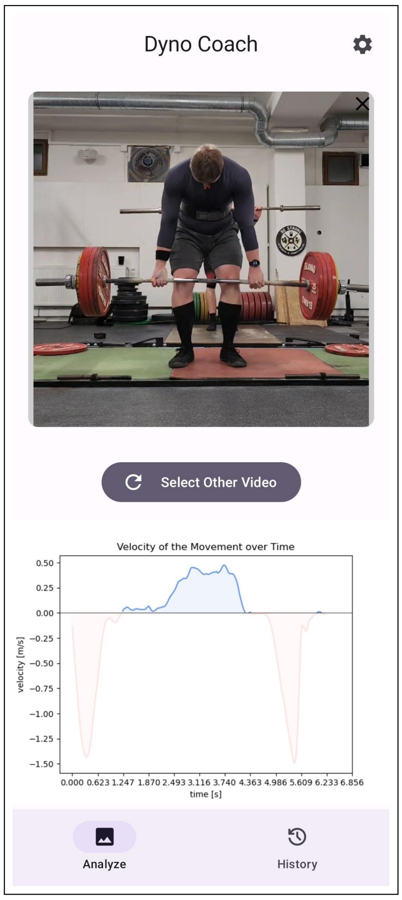
5.5 Client – Android Application

This section discusses the design presented in the early stages of development (between the first phase and intermediate phase as described in Section 5.6). This design was presented to users during user evaluation sessions. The final result is then shown in Section 6.6

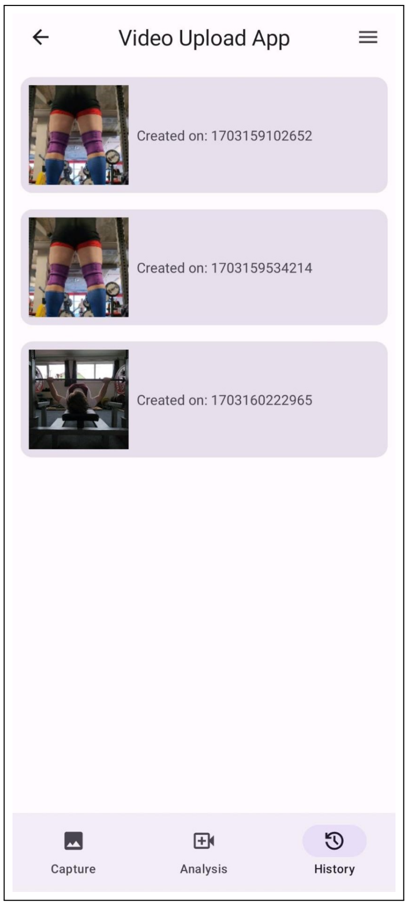
5.5.1 Application Layout

Please, see the proposal design for *Analysis* screen and *History* screen in Figure 5.6. The UCD diagram is shown in Appendix B.

The user interface is simply designed, so athletes in gyms in dusty and humid environment can easily use the main functionality. The *History* UI elements will also be easy to operate. Smaller elements are not intended to be used in mid-exercise series.



(a) Proposal of *Analysis* screen in client – Android application.



Created on: 1703159102652

Created on: 1703159534214

Created on: 1703160222965

(b) Proposal of *History* screen in client – Android application.

Figure 5.6: Semi-mocked design of the Android application.

5.5.2 Local Storage

Android provides native support for SQLite and Room library can be used for object access (ORM[9]). Since accessing database from main thread is forbidden on Android, a Data Access Object (DAO) approach is necessary. Android platform offers a usage of ViewModel in conjunction with `LiveData` to observe changes in the database and provide live UI updates. This approach should be used when presenting data from the database to the user.

[9]ORM = Object Relational Mapping

Data about video and exercise along with data from the server-side analysis will be stored in SQLite database. Any binary data will not be saved in the database. To provide later access to the video, its path will be stored in the database only. The velocity graph will be saved on the device's disk and its path in the database.

At first, the video URI was considered to be used as the primary key. However, the automatic replacement of the same video is not intuitive for users. Instead of video URI, an autogenerated ID is used as the primary key. The proposed database scheme can be seen in Figure 5.7. The scheme is simple, which corresponds to the idea of keeping the application simple and clean for ease of use in harsh conditions of gyms and fitness centers.

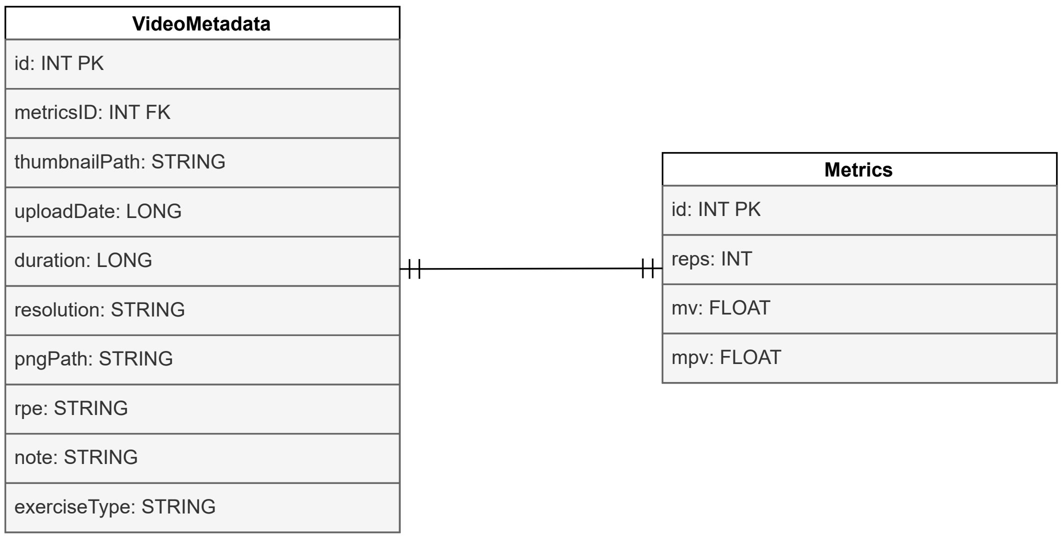


Figure 5.7: Entity Relationship Diagram for analysis record. `VideoMetadata` represents data about video and user inputs. `Metrics` table represents the metrics values obtained from the JSON response from the server.

5.6 Iterative Development and User Evaluations

The development methodology used was based on iterative development with incremental additions and evaluations. Even in a minimalist form, we adhered to the pure Scrum methodology as defined in The Scrum Guide [55].

In the first phase, the most viable product (MVP) was created and presented to the first users. From the initial feedback collected, the next steps were refined and enriched with new ideas. During the second phase, incremental additions and improvements were made and periodically evaluated with alpha testing users. In the final phase, the application was presented as "finished" and the user's impressions and satisfaction with the made changes were gathered.

The whole process was tracked on a weekly basis in Google Docs with the supervisor. For personal project management and task tracking, Plane.so[10] was used.

[10] https://plane.so/

In the context of consultation with users about application UI, functionality and performance, the term evaluation is used rather than testing. Especially in the early phases users were commenting on the proposed UI with limited functionality and not really testing a production-ready application.

Multiple user testing sessions were conducted during the development of the application in an iterative way. The first type of user testing session was more exploratory. It was more challenging for users because they had to work with UI that – in some cases – was not fully responsive, contained missing titles, or there were occasional crashes of the application, and so on.

However, the first exploratory sessions were extremely insightful in regards to addressing the important features and UI and UX aspects from the users' point of view. The users were proposing their opinions on what they expected and what they experienced if the actions resulted in the expected results. If all the titles and names are explanatory enough, etc.

The following sessions with the users had evaluation and testing scope. For users who already had the exploratory session, their particular points were re-evaluated after the second session. Among the improvement proposals, there were crucial points which were solved accordingly.

There were also points that imply creating a completely new module, such as automatic depth-of-motion verification. These ideas were not considered to be solved in the scope of this work for their low value/time spent ratio. Such demanding ideas were evaluated as possible extensions of the application and can be addressed in the following development; more details are described in Section 7.9.

Chapter 6

Implementation of Server-Client Application for Strength Sports Training

This chapter discussed the details of the implementation of the application. Practical documentation for programmers and developers who would like to continue with this work (including ourselves) is created separately. Android application and server code (placed in `App/` and `Server/` directory respectively) have each own README.md files. These README.md files describe the project more comprehensively and from different point of view.

The top-level README.md is used as an introduction and a signpost. It is advised to start there when studying the source codes. In the following text, when referring to folders, it is always in the context of mentioned root directories.

6.1 Server API

The server part of the client-server architecture is implemented in Python with the FastAPI library. A brief description of the stack of Python libraries for API is presented next.

ASGI (Asynchronous Server Gateway Interface) is a standard interface between asynchronous Python web servers, frameworks, and applications. It is designed to replace WSGI (Web Server Gateway Interface) to provide a standard for asynchronous web apps to run on asynchronous servers, enhancing the capability to handle more concurrent connections.

Uvicorn is a fast ASGI server implementation. Built on `uvloop` and `httptools`, Uvicorn provides a simple and efficient way to deploy and manage asynchronous Python web applications, such as those built with frameworks like Starlette or FastAPI. It supports automatic reload, server hooks, and WebSocket handling, serving the needs of modern Python web development.

Starlette is a lightweight ASGI framework/toolkit for building asynchronous web applications in Python. Starlette supports requests directly from any ASGI server, offering a robust foundation for WebSockets, GraphQL, in-process background tasks, and more. Starlette middleware is used for custom functionality as can be seen in Listing 9.

FastAPI is a modern, fast web framework for building APIs with Python based on standard Python type hints. FastAPI is a fast web framework with high performance as can be seen in the TechEmpower benchmark[1]. FastAPI[2] is built around Starlette[3] which is built around Uvicorn[4].

This combination allows FastAPI to offer automatic interactive API documentation (with Swagger UI and ReDoc) and serialization that is deeply integrated with Pydantic[5].

6.1.1 Endpoints

Three endpoints are implemented, the primary and most crucial being `/graph`. The brief explanation of endpoints is as follows:

- `/healthcheck`
  - Request: GET method.
  - Response: test if the connection to the server is live.
- `/keypoints`
  - Request: POST method with video bytes as `multipart/form-data`.
  - Response: JSON with raw processed keypoints. Not smooth or filtered.
- `/graph`
  - Request: POST method video bytes as `multipart/form-data`.
  - Response: JSON object with base64 encoded velocity graph and metrics values.

To see the webpage UI for API documentation using Swagger UI or Redocly[6] respectively, run the server and visit `/docs` or `/redoc` respectively.

To limit automated exploiting requests from foreign countries and keep the setup simple (e.g. not creating the firewall), IP blocking middleware was created. Starlette provides a `BaseHTTPMidleware` class that was extended to limit requests only to Czech IP addresses.

This setup is not optimal for wide scaling on the international market, but in the state the product is at the moment of writing, it is a sufficient solution. You can see the IP blocking middleware in Listing 9. The example of blocked requests can be seen in Listing 10.

To have a bit more customized documentation webpages with Swagger UI and Redocly on paths `/docs` and `/redoc` respectively, `BaseHTTPMiddleware` base class was again utilized to

[1] https://www.techempower.com/benchmarks/#section=test&runid=7464e520-0dc2-473d-bd34-dbdfd7e85911&hw=ph&test=query&l=zik0zj-6bi
[2] https://fastapi.tiangolo.com/
[3] https://www.starlette.io/
[4] https://www.uvicorn.org/
[5] https://docs.pydantic.dev/
[6] https://redocly.com

create `CustomSwaggerUIMiddleware` and `CustomRedoclyMiddleware` classes. These middlewares effectively change the favicon and title of API documentation pages. At the same time, with usage of middleware, the FastAPI's HTML template is kept, and a new HTML template does not have to be created.

Listing 9 Starlette middle-ware for blocking non-Czech IP addresses on Python FastAPI server. The `bogon` refers to local network IP addresses and is there to enable usage on the same network that the server is running on. The response does not contain detailed information on purpose reflecting famous *security by obscurity*.

```python
class GeoIPMiddleware(BaseHTTPMiddleware):
    async def dispatch(self, request: Request, call_next):
        client_ip = request.client.host

        async with httpx.AsyncClient() as client:
            try:
                response = await client.get(
                    f"https://ipinfo.io/{client_ip}/json?token={API_TOKEN}"
                )
                response.raise_for_status()
                data = response.json()
                if not data.get("bogon", False) and data.get("country", "") != "CZ":
                    return Response(status_code=403)
            except httpx.HTTPStatusError:
                return Response(status_code=503)
            except KeyError:
                return Response(status_code=500)

        return await call_next(request)
```

Listing 10 Example of successfully blocked request from Honk Kong, and Oregon and Virginia, US.

```
INFO:     152.32.213.68:33536 - "GET / HTTP/1.1" 403 Forbidden
INFO:     152.32.213.68:33780 - "GET /favicon.ico HTTP/1.1" 403 Forbidden
INFO:     152.32.213.68:34026 - "GET /sitemap.xml HTTP/1.1" 403 Forbidden
INFO:     152.32.213.68:34028 - "GET /robots.txt HTTP/1.1" 403 Forbidden
INFO:     15.204.207.48:45322 - "GET / HTTP/1.1" 403 Forbidden
INFO:     15.204.31.10:53524 - "GET / HTTP/1.1" 403 Forbidden
```

6.2 Human Pose Estimation

MMPose framework offers a quick setup for inference of various models. The use of provided demo scripts such as `inferencer_demo.py` with broad customization arguments can bee seen in Listing 11.

Listing 11 Usage options for `inferencer_demo.py` script. There is a set of predefined model aliases. Alternatively, users can provide their own configs and model's weight, both for detection and pose estimation models. Mention the only one positional argument `inputs` at the very end. There can be provided single image, directory with multiple images or a video.

```
inferencer_demo.py [-h] [--pose2d POSE2D] [--pose2d-weights POSE2D_WEIGHTS]
[--pose3d POSE3D] [--pose3d-weights POSE3D_WEIGHTS] [--det-model DET_MODEL]
[--det-weights DET_WEIGHTS] [--det-cat-ids DET_CAT_IDS [DET_CAT_IDS ...]]
[--scope SCOPE] [--device DEVICE] [--show-progress] [inputs]
```

Based on the findings presented in Chapter 7, it has been determined that among the existing models for human pose estimation, there exists a model that meets the necessary criteria for accuracy, robustness, and size that are suitable for a client-server application with various alternatives. Following discussions with the supervisor, the models RTMDet and RMTPose were selected as adequate, requiring no additional fine-tuning for our specific needs.

The MMPose framework provides straightforward methods for conducting both training and inference sessions. Additionally, it maintains a website featuring a variety of pre-trained models with different sizes and backends (ONNX, ncnn). Therefore, it is also advantageous for ongoing development and experimentation, particularly for transitioning the model exclusively to a mobile client, to evolve our solution utilizing the MMPose framework.

Since the model was not trained any further, with regard to the model inference topic, only the integration and usage are discussed in this section. We take the leverage of demo scripts and use them to run the inference on the incoming request from the client. The most seamless usage was using a subprocess. You can see how the command is run from Python script in Listing 12. The code of the MMPose framework is not edited in any way.

Listing 12 RTMPose inference run as a subprocess on the server in incoming request for video processing from client. The `-pose2d` argument is specifying the pose estimation alias. Other arguments are sufficiently self-explanatory. For experiments with detection model sizes extra parameters `-det-model` and `-det-weights` were used.

```
subprocess.run(
    [
        "python",
        "../inferencer_demo.py",
        video_filename.__str__(),
        "--pose2d",
        "human",
        "--vis-out-dir",
        "vis_results",
        "--pred-out-dir",
        "./",
        "--draw-bbox",
    ]
)
```

To improve inference time, the supplementary scripts for framerate and resolution reduction (functions `resize_video()` and `reduce_frame_rate()`) and resolution were created. The current preset values for resolution reduction are *lower*, *medium*, and *high/original*, with corresponding 256, 512 and original pixels. The API parameter is called *quality.* The value defines the length of smaller video side (e.g. 256x756 or 1234x512).

The reduction of FPS provide 5 options – from 10 to 30 FPS. The user has options available in the settings. This is described in Section 6.6.6.

For debugging purposes, the script `get_frames.py` can be used to get (undersampled) frames from the video. It can be used on any video, so by using it on visual video output from the RTMPose inference, the developer can investigate visually if any anomalies are happening such as missing or hopping[7] bounding box or keypoint hopping[8].

6.3 Calibration of Pixel Measurements to Metric Units

The unit of result of the inference of the RTMPose model is pixel. The conversion from pixels to some metric unit such as centimeter or meter must be done to correspond to the movement velocity measurement standards. Based on analyzed studies, meters per second were chosen as a unit of movement velocity. Therefore, the values that describe the position of keypoints in pixels are converted to meters.

The conversion algorithm is crucial for the accuracy and consistency of the values produced between different videos. Multiple assumptions, approximations, and robustness enhancing steps were made.

First, the presumption that the largest person appearing in the video is the main character is made. We expect that athletes either try to film themselves or at least try to get reasonable quality via filming from adequate distance and angle. For that reason, we pick the largest bounding box and use it as a reference.

Then for each frame, the best matching bounding box is found on the basis of the reference bounding box.

The similarity of the bounding box S is calculated as

$$S = |x_r - x_i| + |y_r - y_i| + |w_r - w_i| + |h_r - h_i|,$$

where x and y are coordinates of the top left corner and w and h are width and height of the bounding box. r and i indices respectively denote the reference and i-th bounding box.

The next step is to calculate the scale ratio to convert the movement from pixels to meters. There are two calculations; first the reference bounding box is taken, and since it crops the person tightly, the height can be considered as the person's height. The second value is calculated on the basis of the anthropometric points on the human body and the estimated pose skeleton. The COCO style labels include the ankles and nose. From statistical anthropology studies, we can take the mean distance from the top of the head to the nose and the height of the ankles from the ground and subtract these values from the person's defined height. We can then measure the distance of the keypoints of the nose and ankle and get

[7]By bounding box *hopping* we mean the jump of bounding box from one person to other without detecting the other person simultaneously. This can happen for example if second person is walking in front of camera.

[8]By keypoint *hopping* we mean jump of keypoint label/annotation for example from left knee to right knee having both keypoints at the same knee. This can happen in occluded or in low-light scenes.

the second scale ratio. The first ratio can be near 50% larger, because the RTMDet model adds a 1.25 times area for context in certain cases. By comparing the two, we can decide whether we need to adjust the first ratio or not. In the end, the two ratios are averaged to get a more robust approximation of the ground-truth conversion ratio.

The anthropometric points are gender specific. The needed values are obtained from the ANSUR II study [23] as described in Section 5.4.1. The server uses predefined getters to obtain the correct values based on the parameters obtained from the request from the client.

6.4 Velocity Computation

Movement velocity is now implemented in a way that supports all possible exercises where the athlete is holding the bar in hands. The velocity is calculated as $y_i - y_{i-1}$ position differences in the time of the left and right wrist. The y positions of wrists are `np.array` objects so the average can be easily computed with these vectors as

$$\vec{y}_{\text{avg}} = \frac{\vec{y}_{\text{left}} + \vec{y}_{\text{right}}}{2}.$$

This is an approximation of the center of the bar. Please, note two important remarks. First, if recorded from an angle, the farther wrist is showing a lower velocity on camera, when in reality it would be the same. Second, movement can be unequal. Typically for deadlift and mixed grip or in weightlifting, body rotation can happen. Averaging the left and right positions provides some level of data smoothing. For occluded scenes, it also helps to reduce the noise generated by the model uncertainty at the occluded body parts.

For data smoothing, the Kalman filter was used. Smoothing is done first on the wrist coordinates to smooth the raw position data. Then velocities are computed and smoothed again by the Kalman filter. This combiner approach produces the most visually pleasant results. Certain experiments with data smoothing are described in Section 7.4.

6.5 Strength Sports Performance Metrics

Sports performance metrics can give a meaningful insight for the athlete just in the training session or can be useful for micro-cycle planning for coach and the athlete as well. The theoretical background including discussion of supportive sport studies discussion is examined in Section 3.1. In the following sections, the implementation steps of those metrics and the pre-processing are discussed.

In a personal consultation Bc. Jakub Šigut[9] said that from his point of view, MV is the most valuable metric.

There was not an increased interest in the impulse metric from users or from powerlifting coaches. Impulse is a sensitive metric and requires another user's input, the weight used. For that reason, impulse metric is not implemented.

[9]Bc. Jakub Šigut is a physiotherapist and personal powerlifting coach in the Revolution Powerlifting organization (`https://linktr.ee/revolution_powerlifting`).

6.5.1 Counting Repetitions

The detection of repetitions again requires some heuristics and approximations emerging from assumptions based on knowledge of the topic. Having a squat as an example, the athletes place the bar on their shoulders and upper back.

Then some of them do a few little pushes that look like 10% squat, this could be taken as repetitions by accident and the algorithm needs to prevent this. The walkout as well as racking back can be shaky. During the set, the single repetition can again end with a slight bounce caused by the biomechanics of the athlete's body.

The described issues are solved in a complex way. First, the smoothing of the data is already removing the low-magnitude noise, mainly coming from the pose estimation itself but also covering very small movements. However, it does not solve more prominent movements like shaky walkout or inter-reps bounce.

To address this problem, a dynamic threshold is set at 10% of the maximum amplitude. Still, the athlete can bounce so much that it would look like some little range of motion repetition. For that purpose, half of the average duration of detected repetitions is taken as the threshold for repetition duration. But then the repetition is defined from zero crossing points, not from thresholds. You can see the whole algorithm of repetition detection in Appendix A in Listing 15.

6.5.2 Mean Velocity and Mean Propulsive Velocity

Calculation of mean velocity (MV) and the mean propulsive velocity (MPV) is calculated for one repetition. We need to first get repetitions as described in Section 6.5.1 and then we can calculate the metrics. The implementation follows the design proposed in Section 5.4.3.

With the data split into repetitions already, the next step is to get the concentric phase. This would be a trivial task if the movement would not oscillate around the zero in between the repetitions.

With that said, the largest amplitude concentric phase is considered as the main concentric phase. See Listing 13 for the algorithm to find the concentric phase for each repetition.

There could be an issue with paused exercises, where the bar can have a very subtle eccentric dip and generate two similar peaks. In such case, the concentric phase would be the larger of these two and the algorithm would produce in fact an incorrect result.

However, this is a very specific aspect, and the solution would be very complex involving detection of absolute position in the space, athlete's ranges of motion in various exercises, etc.

The mean velocity is then computed as a simple mean of the concentric phase `c` as `mv = np.mean(c)`. The mean propulsive velocity can be computed easily again thanks to already prepared concentric phases. The propulsive phase is computed as `p = c[: np.argmax(c) + 1]`, then mean propulsive velocity is computed as `mpv = np.mean(p)`.

Listing 13 Algorithm which finds the concentric phase for one repetition. The amplitudes of every possible concentric phase are computed and the largest movement is considered as the main movement.

```
positive_mask = rep >= 0
padded_mask = np.pad(
    positive_mask, (1, 1), mode="constant", constant_values=False
)
diff = np.diff(padded_mask.astype(int))
starts = np.where(diff == 1)[0]
ends = np.where(diff == -1)[0]

amplitudes = [rep[start:end].ptp() for start, end in zip(starts, ends)]

if not amplitudes:
    return np.array([])
max_amp_index = np.argmax(amplitudes)
concentrics.append(rep[starts[max_amp_index] : ends[max_amp_index]])
```

6.6 User Interface

In this section, possible user interactions available in the app are discussed. It has strong implicit requirements basement in the habits of athletes discussed in more detail in Section 2.4. In this section, by *application* is meant the Android application, not the whole system.

Android application serves the purpose of the client in the designed server-client application. There are multiple modules that are structured into folder structure based on the functionality they have. The latest (release candidate) Kotlin version 2.0.0-RC1(RC2) and the latest compatible libraries are used.

The application is a single activity application with Jetpack Compose UI components. The implementation uses declarative Compose code[10] instead of XML layouts. The top level UI consists of `Scafold` layout with top and bottom bars and a side drawer on the right. Transition between different screen is made through the `NavHost` component which sets the correct content.

The top bar is `CenterAlignedTopAppBar` component providing access to *Settings* in the right drawer. The drawer us implemented using `ModalNavigationDrawer` component and visibility is controlled by `drawerState`. The bottom bar is `NavigationBar` component providing *Analysis* and *History* tabs via `NavHostController` class.

All UI components are in `ui/` folder, where `ui/common/` contains common components such as `RowPick`, `InputChip`, or `Dropdown`. Main *Analysis* screen, *History*, and simpler *Settings* and *Help* are all at top level `/ui` or in own subdirectory e.g. `/ui/History/`.

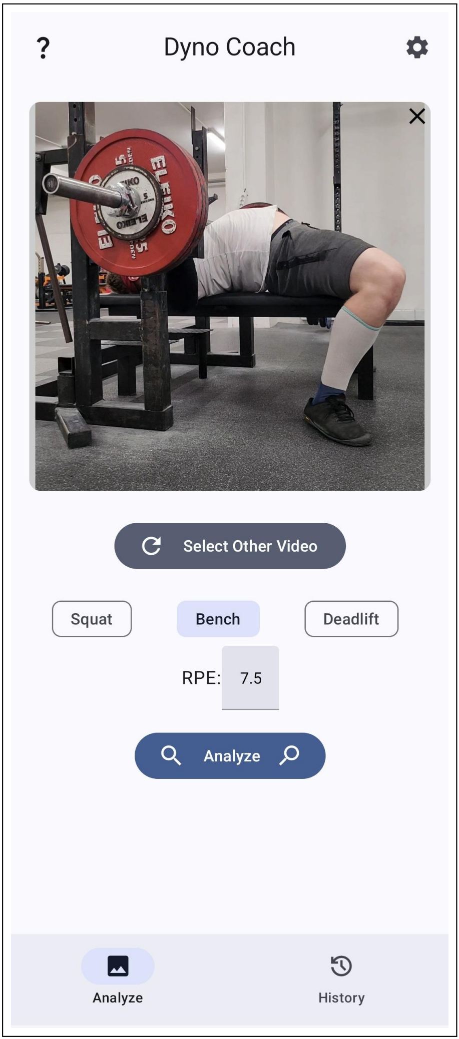
An important part of the UI is the ability to play videos. Users must be able to see what they have selected and are going to upload to the server, even though it is stateless processing, and video recordings are not stored. The video player is implemented with the `ExoPlayer` interface. It is used in *Analysis* screen and in recording detail in *History* screen.

[10]Compose code is a domain-specific language different from pure Kotlin.

6.6.1 Analysis Screen

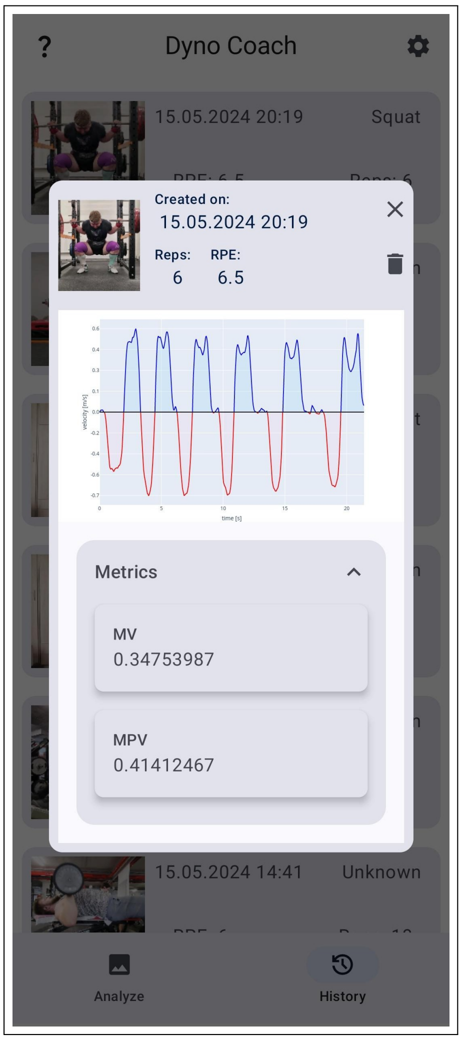
Analysis screen enables user to pick a video from the gallery. If the user did not provide the permissions with the first application run after installation, the application asks again. The latest permission handling for Android 14 (API level 34+) is implemented. This means that the user can give only partial permissions for certain files (videos).

After the video is selected, the user can adjust the RPE value and select from the three predefined exercises. Currently, only the three powerlifting exercises are supported.



RPE: 7.5

(a) Analysis screen with already selected video from gallery. Optional values RPE, and exercise type are chosen. Now, the user can send the video to the server for analysis.



15.05.2024 20:19

Created on:
15.05.2024 20:19

Reps: 6
RPE: 6.5

MV
0.34753987

MPV
0.41412467

15.05.2024 14:41

(b) Detailed information about analysis in *History* screen. All data available after analysis at the *Analysis* screen are also presented here.

Figure 6.1: Primary UI screens.

6.6.2 History Screen

A LazyColumn was utilized to list each video processing record, with each item displayed in a card format containing the video thumbnail, creation date, and video duration. The video duration is important so that the user can easily distinguish between sets recorded from the same spot. This is a specific requirement for this application that comes from the unique usage and habits of strength sport athletes. More about athlete technology habits is discussed in Section 2.4.

History screen shows the previously analyzed videos in `LazyColumn`. The component `LazyColumn` renders items on demand, so for a large number of videos, it does not render all of them at the time. It corresponds to `RecycleView` in XML layout.

Each item in *History* is a `HistoryCard` showing basic information and thumbnail of video. `HistoryCard` items can be enlarged into the `HistoryCardDetail` observable in Figure 6.1b.

`HistoryCardDetail` shows detail information about the analysis. The `VideoPlayer` component is embedded into video thumbnail picture User can replay the video from the past. The video is not stored redundantly in the database but is accessed through its path, which is saved in the database. For that reason a permanent access to such files has to be requested with `FLAG_GRANT_READ_URI_PERMISSION` flag.

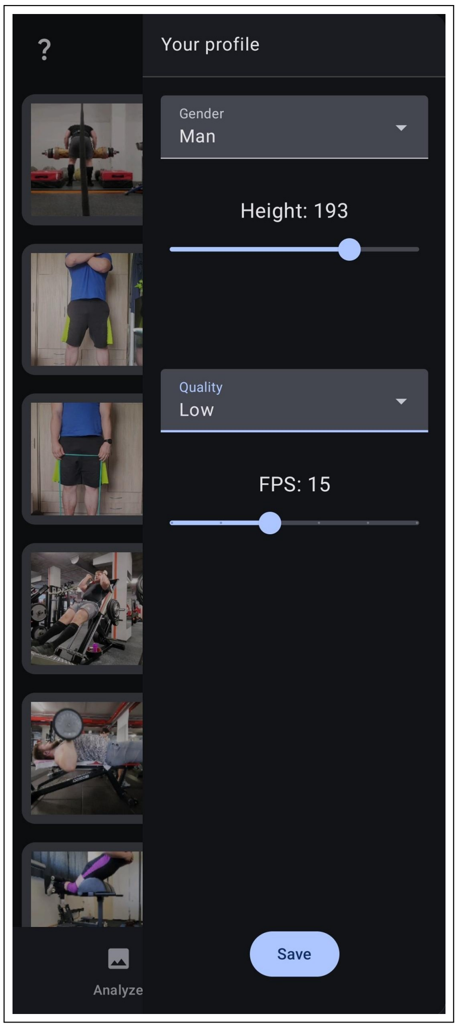
The metrics visible also in exercise detail are implemented as a fixed layout and length component with regular `Column` and `Card` base component with one nested level. The user can delete the analysis only from the `HistoryCardDetail` component. Delete confirmation dialog prevents accidental deletions.

6.6.3 Secondary Screens

Settings screen provides input fields for personal information and other settings. It is implemented as a right-hand drawer that supports swipes from right to left. The access is also provided through the settings icon.

Help can be shown using the question mark in the top left corner. It gives instructions on how to use the application and emphasizes the importance of setting up personal information for proper calibration.

In Figure 6.2a dark themed *Settings* screen for user's height and other parameters adjustments is shown. In Figure 6.2b can be seen the *Help* screen with instructions on how to use the application.



Your profile

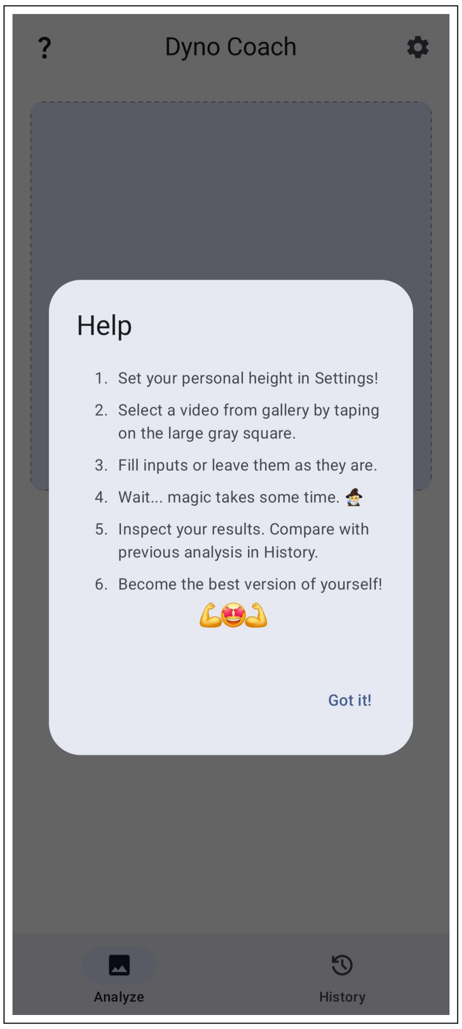
Gender
Man

Height: 193

Quality
Low

FPS: 15

(a) *Settings* screen.



Help

1. Set your personal height in Settings!
2. Select a video from gallery by taping on the large gray square.
3. Fill inputs or leave them as they are.
4. Wait... magic takes some time. 🧙
5. Inspect your results. Compare with previous analysis in History.
6. Become the best version of yourself! 💪🤩💪

(b) *Help* screen with instructions.

Figure 6.2: Secondary UI screens.

6.6.4 API Requests and Response Decoding

For application API connectivity the API service singleton is created featuring a logging Interceptor for logging purposes. The API-related code is implemented with usage of OkHttp3 and Retrofit2 libraries.

There most important endpoint is `/graph` as reasoned in Section 5.2. The request body contains video footage and thus a `MultipartBody` type of parameter is used for that. The response from server is more complex, containing the image with the plotted velocities and a JSON with metrics data. Custom Response type is defines as `Response<ResponseData>`. `ResponseData` data class contains serializable `Metrics` data class, which wraps the actual values of metrics.

6.6.5 Storing Analyzed Data on Local Device

Storage possibilities on the Android platform were discussed in Section 4.1.2. The primary objective was to efficiently store video metadata, which consists of thumbnail, upload date, video duration, and file paths of an associated result graph. ERD diagram can be seen in Figure 5.7.

For faster database operation, the thumbnail and result graph are stored only in the form of their respective paths. This approach enables fast data retrieval and display, which, with the usage of lazy rendering, leads to a user-friendly experience.

The chosen storage solution involved the use of SQLite, a lightweight relational database described more in the Section 4.1.2. For the manipulation with the database in the application, Room[11] library, a higher-level abstraction provided by Android for database management, was used. Room allowed for compile-time verification of SQL queries, reducing errors and simplifying database interactions through annotated objects and methods.

Handling the `java.time.Duration` type, which is not natively supported by SQLite. The duration is stored as a long integer representing seconds, which is then converted back to Duration upon retrieval.

For fetching and displaying these data, Android's recommended architecture patterns[12] are used. This ensures a maintainable and responsive application. The combination of asynchronous database operations and lazy rendering results in an efficient, and more importantly scalable, approach to handling video analysis history in the application.

6.6.6 Settings and Information About User

All settings user can tweak are displayed in a *Settings* screen which is containerized in a right drawer. Shared preferences are used as the storage method for these settings.

The user enters personal information such as height and gender, which is used in the server calibration process described in Section 6.3. Another customization option is to choose from 4 levels of processing quality and from a scale of FPS allowed from [10, 15, 20, 25, 30].

The quality option are *LOW*, *MEDIUM*, *HIGH*, and *ORIGINAL*. The last two represent the same value, and the duplicity serves only UX purposes. Having 4 options, users select between 2 and 3. Having 3 options, users tend to the middle one.

FPS lower than 10 and quality lower than 256 pixels for shorter side (*LOW* settings) produce questionably reliable results even for good light, front view and erected poses. Limiting FPS to 30 which is also a initial value, is beneficial to processing time if someone has preset 60 FPS on their device.

6.7 Android Application Deployment on Google Play

The release of the application even for closed testing is conditioned by several requirements. The content related requirements are: setting a privacy policy, app access, ads, content

[11] https://developer.android.com/jetpack/androidx/releases/room
[12] https://developer.android.com/reference/androidx/lifecycle/LiveData

rating, target audience, news apps, COVID-19 contact tracing and status apps, data safety, government apps, financial features, and health. For managing the app's organization and presentation, selecting an app category and providing contact details, as well as setting up your store listing is required. All the mentioned steps were completed. The explanation of these steps is described on the Google Play Console support websites[13].

The *Set privacy policy* step requires a functional active online website with a declaration of how the application collects and handles user data. This requirement was accomplished by using personal GitHub Pages. The application's Privacy Policy website is deployed here: https://richardklem.github.io/Dyno-Coach-Privacy-Policy/.

6.8 Internal and Closed Testing on Google Play Platform

Internal and closed testing were performed. Recent changes in Google's policies[14] now require at least 20 external testers to test the application before it can be published publicly. There must be performed Google's own manual inspection of the application. The application must not hide anything behind the account creation. For Google's inspection, demo accounts must be created in advance and provided for the Google's inspectors. External testers must be opted-in in closed testing for at least 14 days. For both closed testing and public publishing, the privacy police website must be provided.

In the first round, the internal testing was performed as it fortunately does not require anything special. The application must be uploaded as an *android application bundle* (*.abb*) signed with upload key. The application then can be published as optimized *.apk* signed with Google generated automatically managed key, which is preferable way.

The necessary steps for closed testing and public publish were done, providing all the required information, statements, graphics and declarations. The review should be done in 7 days, but it can take longer. In case of our Dyno Coach application, it took 3 days. However, the application does not require almost any permission and has only one activity and two screens. For applications requiring many permissions and complex functionality, it is reasonable to expect a longer review. After providing a supplementary graphics, we could start the closed testing track. Closed testing is the testing stage where there is a requirement for 20 testers for 14 consecutive days.

An invitation was created for closed testing[15]. Relevant users were asked to opt-in as testers for closed testing. However, we were not able to pass the requirement completely yet.

[13]https://support.google.com/googleplay/android-developer/answer/9859455
[14]https://support.google.com/googleplay/android-developer/answer/14151465
[15]https://play.google.com/apps/testing/cz.vut.fit.xklemr00.vbt.app

Chapter 7

Experiments with Human Pose Estimation

During the exploration phase at the beginning and also later on, many different machine-learning models for human pose estimation were examined. Because we have results of experiments from 13 different models and one motion capture system (Qualisys), only models with the most different approach or architecture are presented in this chapter and others are attached in Appendix F.

More theoretical details on human pose estimation are described in Section 3.5. For some experiments there is a Jupyter notebook in the attached media as complementary source of results.

7.1 RTMDet models sizes

The MMPose framework provides two configurations for the RTMDet detection models, trained on human body detection on the COCO dataset. The configurations include a smaller *nano* version – `rtmdet_nano_320-8xb32_coco-person.py`, and a standard *medium* version – `rtmdet_m_640-8xb32_coco-person.py`. We fixed the remaining system parameters at high settings to achieve the clearest possible outcomes in the tests.

The parameters were set to FPS = 30, resolution = original, and RTMPose model = `rtmpose-m_simcc-body7_pt-body7_420e-256x192`. Manual qualitative assessments were conducted, visually evaluating the outcome videos with displayed bounding boxes.

Throughout all evaluations, the *nano* size detection model was consistently underperforming compared to the *medium* size model. To be more specific, the *nano* model was detecting people where they were not. The detection of the main character was on par with *medium* model.

Typically, even in well-lit scenes without any obstructions, several *artifact* individuals were identified for a short period of time. However, our main person tracking algorithm managed to maintain unaffected processing. An example of this detection can be observed in Figure 7.1.

As an example for all, the most complicated scene with low light level and a person occluded with obstacles with even mirrors in the sight reflecting lights and movements was chosen to be shown. You can see the result of detection in Figure 7.1. However, the *nano* model was less stable and the slight flickering affected the following RTMPose inference. Together with the negligible inference time difference in the entire pipeline, we kept using RTMDet *medium*.

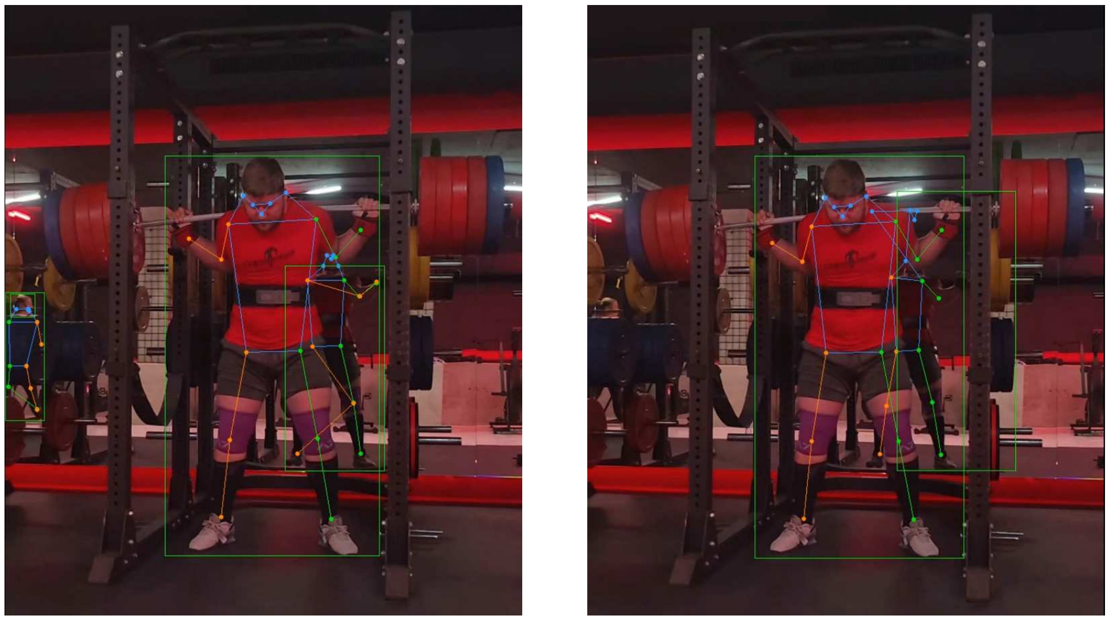

(a) RTMDet-m (b) RTMDet-n

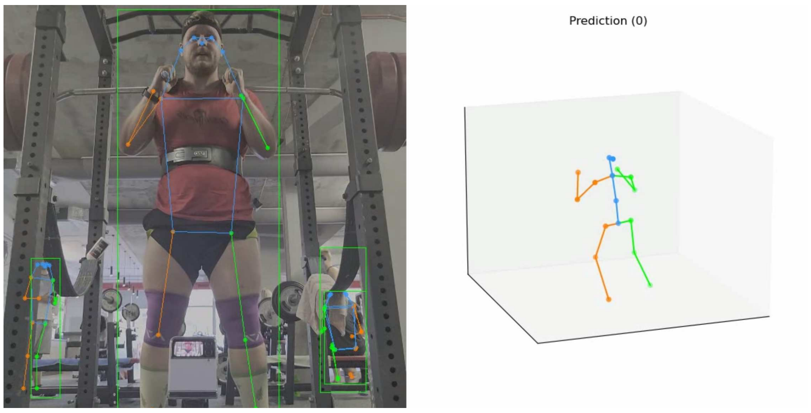
Figure 7.1: Comparison of the *medium* and the *nano* RTMDet model sizes. The detection is on par and usable for our solution.

7.2 Experiments with Models of Various Architectures

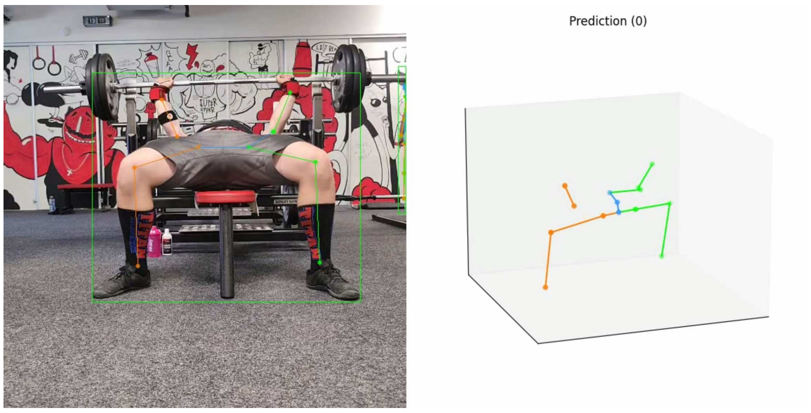
In Chapter 5 multiple experiments are shown with human pose estimation models. However, more experiments were conducted along the way. Here we show results from experiments with such models which address some special issue or aspect such as occlusion or 3D scene. The rest of the experiments with even more human pose estimation models is attached in Appendix F.

7.2.1 MotionBert

Experiments with MotionBERT produced mixed results. In some cases, the 3D reconstruction was precise. However, the results cannot be better than 2D pose HPE since it is the input for MotionBERT. The idea of using MotionBERT for self-supervised training for complex exercises such as benchpress is not promising.



(a) Screenshot from SSB exercise.



(b) Screenshot from benchpress exercise.

Figure 7.2: The result of MotionBERT is up to as good as the 2D pose estimation input. See original generated videos for the complete results.

Listing 14 Bash command for running MotionBERT inference from MMPose framework.

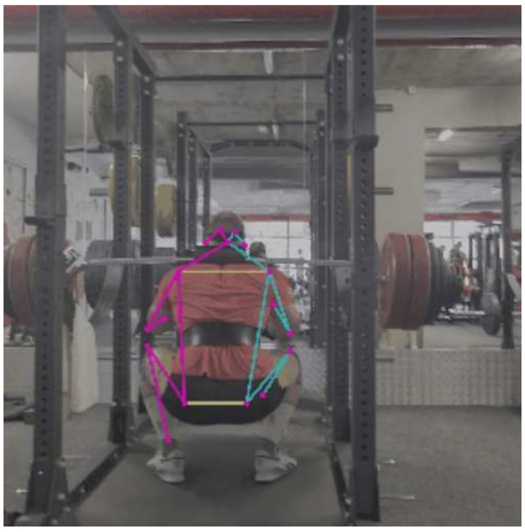
```
python demo/body3d_pose_lifter_demo.py
demo/mmdetection_cfg/rtmdet_m_640-8xb32_coco-person.py
rtmdet_m_8xb32-100e_coco-obj365-person-235e8209.pth
rtmpose-m_8xb256-420e_body8-256x192.py
rtmpose-m_simcc-body7_pt-body7_420e-256x192-e48f03d0_20230504.pth
video-pose-lift_tcn-243frm-supv-cpn-ft_8xb128-200e_h36m.py
videopose_h36m_243frames_fullconv_supervised_cpn_ft-88f5abbb_20210527.pth
--input ssb_front.mp4
--output-root
vis_results
```

7.2.2 MoveNet

MoveNet [67] presented by Google Research is human pose estimation model created for real-time applications. The backbone is based on MobileNetV2. MoveNet uses 192x192 inputs for Lightning (less accurate) and 256x256 for Thunder (more accurate), with intelligent cropping based on previous frame detections to maintain focus on the main subject.

Temporal filtering is applied to smooth keypoint estimates, reducing noise and lag during fast motions. Both variants achieve frame rates exceeding 30 FPS on modern hardware.

MoveNet is used also in experiments with HPE model inference on mobile devices which are described separately in Section 7.4.


(a) Result of `int8` quantized *lightning* light MoveNet model size for SSB from parallel medium distance camera position.

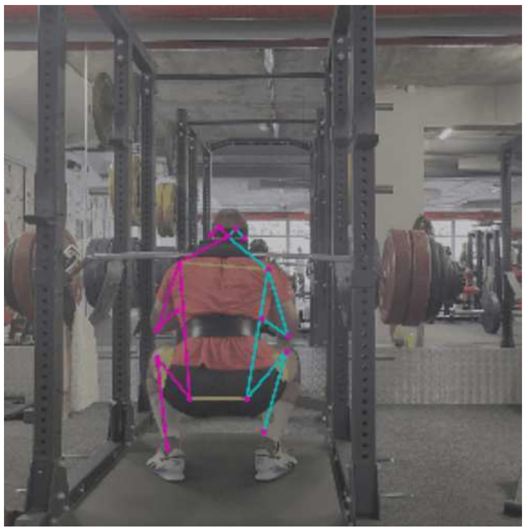

(b) Result of *thunder* heavier MoveNet model size for SSB from parallel medium distance camera position.

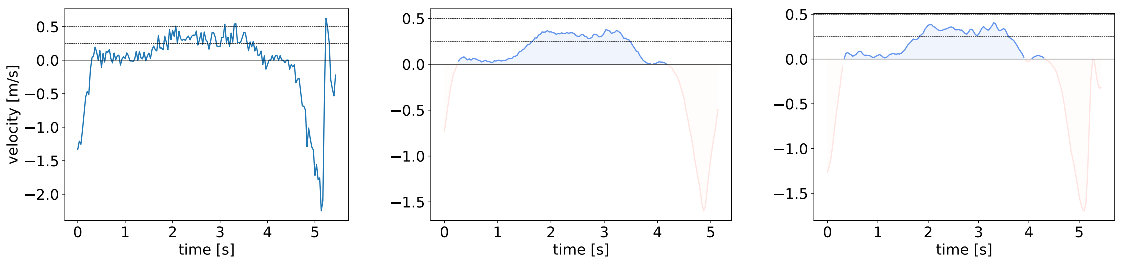
Figure 7.3: Examples of experiments with MoveNet inference.

7.3 Smoothing Methods

Since position input data looked cleaner than computed velocities, we tried to smooth the velocities first. The baseline naive approach was the moving average. We tried different parameters with different sliding windows.

Other evaluated methods include the Savitzky-Golay filter, local regression (Lowess), and the Kalman filter. Kalman filter was used at the end of the solution. Highly experimental experiments were conducted with the Fourier transformation and a combination of techniques.

In Jupyter notebook `calibration_and_smoothening.ipynb` it was evaluated how different methods and algorithm can reduce noise, smooth data, and what side effects each method brings with it. In Figure 7.4 it can seen the comparison of the selected smoothing methods. Although smoothing of velocity only is shown, the final solution uses smoothing of both movement positions and calculated velocities as mentioned in Section 6.3.



(a) Raw data. (b) Moving average, sliding window of 10 frames. (c) Kalman filter.

Figure 7.4: Comparison of (a) raw velocities, (b) moving average and (c) Kalman filter smoothing. Dashed lines note values 0.25 and 0.5 m/s.

7.4 Inference on Mobile

Original RTMPose paper [35] also refers to the performance of model on the Snapdragon 865 chipset. The documentation for deploying models on the Android platform is provided in MMDeploy repository[1]. However, the reference example is missing and in Issue #2298[2], there is a reference for a separate repository PoseTracker-Android-Prototype[3]. The actual deployment refers to different tools and their build. It is not trivial nor straightforward process. On some places, the instructions for dependent tools or utility are in conflict with instructions of MMDeploy or the example application.

Detailed instructions about compilation and whole setup can be found in Appendix E. I was able to run the application and select the video for processing from the gallery. Unfortunately, an invalid memory reference error was thrown when the model inference was about to start. Most likely it is linked to the native code compiled for OpenCV or MMDeploy SDK. Despite all efforts to debug and track the issue, seeking for help in all referenced projects and on other internet sources, I was not able to solve this error. You can see the application's main screen (`/experiments/PoseTracker/main_screen.jpg`) in supplementary materials.

7.4.1 MoveNet CPU and GPU on ncnn Backend

The second attempt of running the human pose estimation inference on mobile devices was made using MoveNet 7.4.1. In contrast to MMDeoploy, this time an unofficial implementation was used. The starting point was open-source GitHub repository[4] of user FeiGeChuanShu. It has two dependencies, first, Tencent's ncnn framework pre-build for usage on Android with Vulkan API for Arm's Adreno GPU. The second dependency is opencv library for Android from opencv-mobile repository[5] from user nihui.

[1]https://github.com/open-mmlab/mmdeploy/blob/dev-1.x/docs/en/01-how-to-build/android.md
[2]https://github.com/open-mmlab/mmdeploy/issues/2298#issuecomment-1653306717
[3]https://github.com/hanrui1sensetime/PoseTracker-Android-Prototype
[4]https://github.com/FeiGeChuanShu/ncnn_Android_MoveNet/tree/main
[5]https://github.com/nihui/opencv-mobile

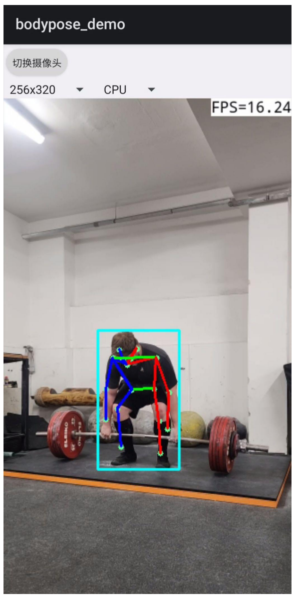


Figure 7.5: Screenshot from demo application using MoveNet HPE models.

Multiple adjustments were made mainly regarding the used versions, from CMake, through Gradle to Android Sdk and Ndk. Java files were converted to Kotlin and severe depreciations were resolved. All changes made were proposed to merge in pull request[6] in the original repository.

There is available both CPU and GPU inference runtimes for two sizes of the MoveNet model – smaller and faster lightning and bigger and slower thunder. Both front and back cameras are supported. For multiple person model, there are three resolution options with the highest resolution crashing the application on a Samsung Galaxy Z Flip 4 smartphone. Despite the intuition that GPU runtime should be faster, that is not the case for this application and CPU runtime is on par with GPU in all lightning model version scenarios. GPU is surprisingly slower by around 30% for a thunder version of MoveNet for a singlepose type of application.

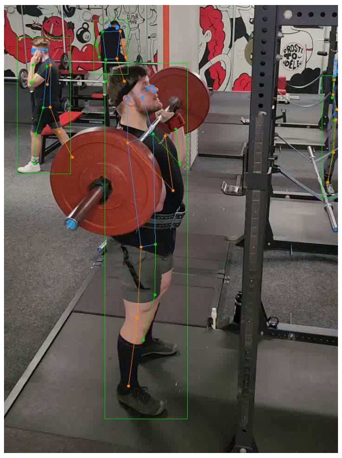
Bellow in Figure 7.5 can be seen the example of MoveNet inference in a gym environment. Specifically, this is a variant for multiperson HPE which uses MoveNet lightning model.

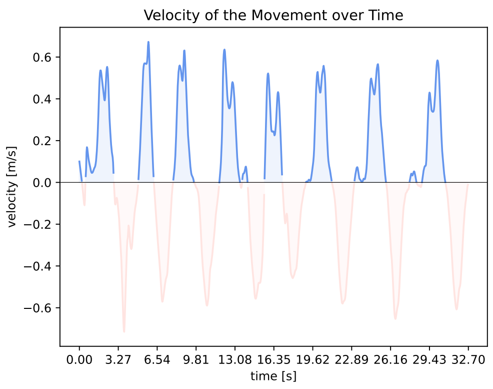
7.5 Other exercises

Experiments on other exercises than squat, deadlift and benchpress were conducted to find out if different body positions and movements are estimated good enough to produce reliable results.

In Figure 7.6 can be seen the example of overhead press (OHP) exercise jointly evaluating other exercise and not traditional position of camera. The position of the camera was height of 3 meters, negative vertical angle 30°. From the velocity graph, during the 5th repetition,

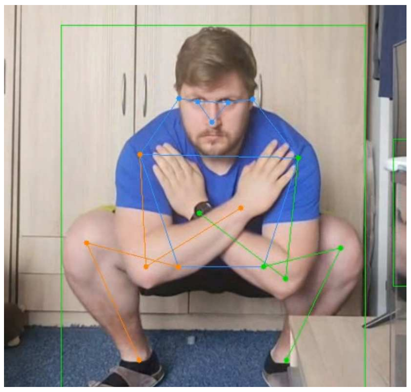
[6] https://github.com/FeiGeChuanShu/ncnn_Android_MoveNet/pull/13/files

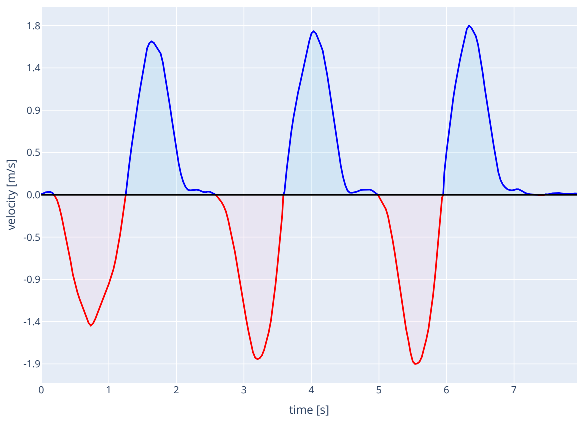

(a) Capture of OHP exercise.



(b) Velocity graph of OHP exercise.

Figure 7.6: Examples of experiments with OHP exercise and nontraditional camera placement.


(a) Snapshot from annotated video.



(b) Velocity graph with 3 repetitions.

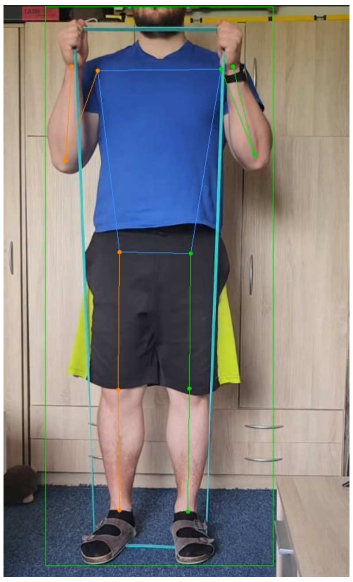
Figure 7.7: Squat with no weight and crossed arms.

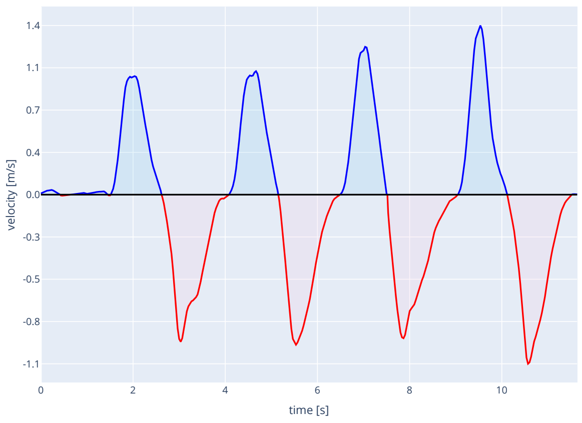
the sticking point can be seen in the middle of the concentric phase which is typical for OHP. The barbell is the heaviest at around 90°in the elbows.

To see how RTMPose estimated the athlete's pose in even more other exercises, please look in Appendix G.

In Figure 7.8 and 7.7 can be seen examples of home exercises: biceps curl with a resistance band and squat with crossed hands on the chest. As is evident from the velocity graphs, our solution is applicable even in home exercises completely without barbells and weight plates.

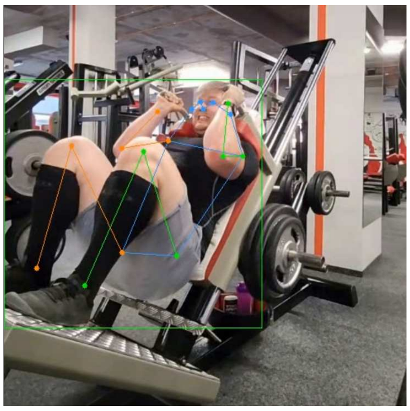
In Figure 7.9 typical bodybuilding exercise Hacken squat is subjected to experiments. The velocity is naturally not tracking the path of the machine. However, the detection and results are consistent. If the athlete keeps a similar camera setup, the velocity metrics would be informative in a long-term comparison even in the current state of the application.

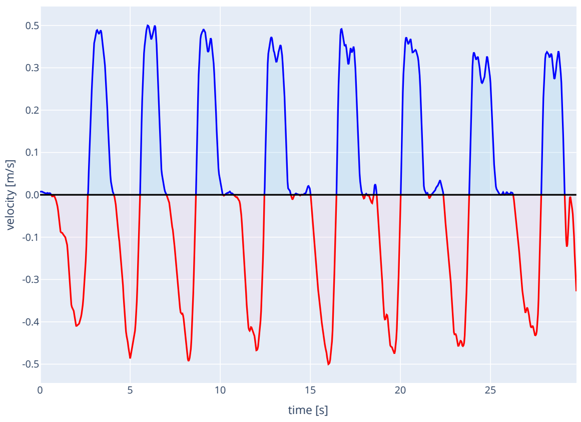

(a) Snapshot from annotated video.



(b) Velocity graph with 4 repetitions.

Figure 7.8: Biceps curl with a resistance band.


(a) Snapshot from annotated video.



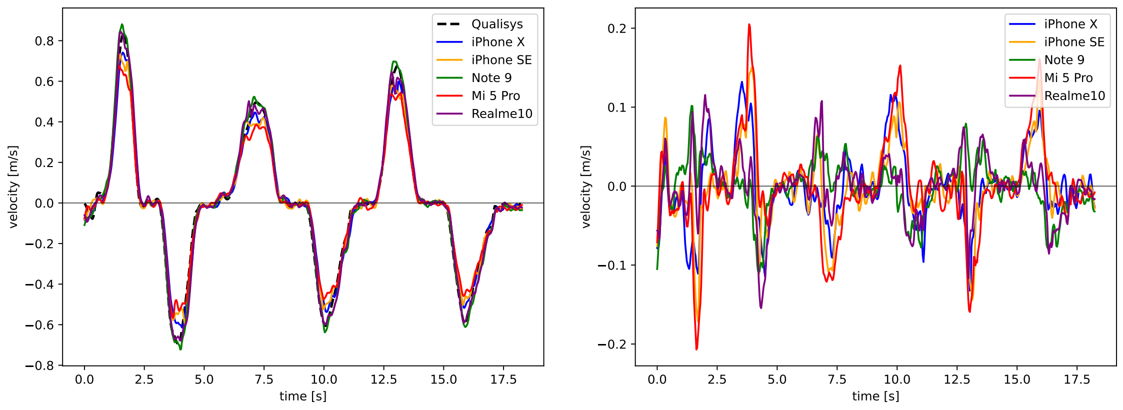
(b) Velocity graph with 8 repetitions.

Figure 7.9: Exercises on certain machines such as Hacken squat can be also analyzed successfully.

7.6 Qualisys Comparison

Experiments with the Qualisys system in a controlled environment were carried out at FEKT faculty in a sports laboratory[7]. To verify the accuracy of our developed solution. In our case, the system was built from 13 Arqus cameras and one Miqus camera for visual reference. The calibration achieved in our measurement session was 3.4mm.

[7]Facilities and equipment for Qualisys measurements had been arranged by Mgr. Jan Šťastný, Ph.D. from CESA BUT.



(a) Comparison of velocities on multiple devices. (b) Absolute deviation from Qualisys reference.

Figure 7.10: Results from comparison experiments with Qualisys motion capture system. The recordings were recorded with the main back camera in preset modes. Computed from three repetitions of deadlift exercise. The black dashed line indicates the Qualisys measurement, which is assumed to be a ground truth.

In Figure 7.10 it can be seen the results of the velocity calculated on multiple devices along with the measurement of Qualisys and the absolute deviation from the measurement of Qualisys. Details about used devices are described in Appendix D.

7.7 Discovered Limitations

During experiments with multiple devices in Section 7.10 deviation from ground truth was found. After the examination of the devices, a possible correlation was discovered between the focal length of the camera and the magnitude of the deviation. You can inspect the devices' specifications in Table D.1 in Appendix D.

After different evaluations of OHP exercises from different session, it was obvious that the camera focal length does not impact the deviation, or at least it is not the major issue. Experiments in the wild have shown completely different results. Theoretical computations in the GeoGebra tool[8] can be found in Appendix H. We found out that the main problem is probably the different angle and perspective changes caused by this factor.

A prominent limitation could be that users cannot define their own tracking points. Allowing users to do this would give them more control and could provide greater value to certain users.

There can be recognized two other eminent limitations. First, video recording right from the application. An automatic cropping mechanism should be implemented, probably using the HPE model, which will definitely be resource-demanding.

Second, HPE inference on the mobile device. Together with the previous points, the mobile device would be under a significant computational load. Other potential limitations are actually opportunities for improving the basic functionality rather than true limitations.

[8]https://www.geogebra.org/

7.8 Evaluation of Results

Multiple types of experiments were conducted to evaluate the robustness, accuracy and consistency of the human pose estimation. Different exercises, environments, camera angles and placements were examined.

Comparison with professional motion capture system Qualisys was performed. The results of experiments are encouraging and prove that human pose estimation models are capable alternative to more traditional computer vision and other methods for sport analysis usage. Solution was also evaluated across multiple mobile devices.

All experiments confirmed the usability of the designed and implemented solution. The limitations were discovered and will be addressed in further work. Significant achievement is comparison with professional system Qualisys. This experiment verified the solution under compliant conditions can produce near same result as could be seen from Figure 7.10b.

7.9 Future Development

Numerous models and approaches were evaluated during the experiments. Similarly, some limitations were discovered. These topics can be addressed in further development. A list of possible extensions and additions to this work is briefly explained next.

All the improvements mentioned were proposed to the user as possible additions, and users agreed that they would appreciate that feature or the idea was brought up directly by users. Details can be found in user evaluation sessions log files in the supplementary medium in `/user_sessions/live_sessions/` folder.

- Advanced Calibration – More accurate calibration is needed to achieve the best possible consistency and especially robustness. The main issue is different view angle, distance from the athlete, and camera position height. Using the device's gyroscope sensors to calculate the view angle can highly compensate for the error. This is the most important point.
- RPE Estimation – Strength sport athletes are used to use RPE metric to evaluate their training sessions. VBT analysis gives valuable information. However, it can be difficult to self-coached athletes to understand the relationships, context, and the whole problematic completely. Using velocity measurements and metrics to estimate the RPE from measured data can bring data-backed insights in a form of traditional and widely recognized metric such as RPE.
- Technique Guidance and Validation – From consultation during the Excel@FIT conference, there was a demand for a technique validation or guidance in the application. The technique guidance and correction can be achieved using keypoints of reference movement and actual estimated movement of the user.
- Custom Exercise – User could be able to define own keypoints to track. This would enable users to create custom exercises. The benefit for a developer is that the responsibility is left to the user.

Chapter 8

Conclusion

In this thesis, a novel method for strength sports performance analysis was introduced. Based on theoretical studies, appropriate tools and approaches were chosen and preliminary experiments verified the initial decisions.

The approach uses human pose estimation (HPE) with the RTMPose model to approximate barbell movement from wrist coordinates obtained from estimated keypoints. Metrics such as mean velocity are calculated, providing feedback without strict requirements on camera angles or visible weight plates.

The Android application offers objective insights into the performance of strength sport athletes by providing the computed data in simple yet complete native user interface. It is suitable for strength sports like weightlifting and powerlifting, and with minor adjustments, can be used by bodybuilders, calisthenics athletes, or strongmen. Currently, the setup involves a Python server for HPE inference, but steps toward edge inference have been made. Calibration and conversion from pixels to metric units are crucial, with several integrated heuristics for precision, though further calibration adjustments are needed for increased consistency.

The application was developed iteratively with user feedback, ensuring practical usability and continuous improvement. Extensive testing confirmed the robustness, accuracy, and consistency of the RMTPose model in both public gyms and home environments. The solution is effective even without barbells, supporting exercises with body weight or equipment such as resistance bands, demonstrating its potential as an alternative to traditional sports analysis methods. Android application can be installed from closed testing Google Play[1] after joining the Google Group[2].

The application was benchmarked against the professional Qualisys system, showing very good performance under appropriate conditions and satisfactory overall results. The solution was evaluated on multiple mobile devices.

Future improvements have been outlined and will be addressed soon. Overall, this work showcases a significant advancement in sports performance analysis, combining human pose estimation with modern software development to provide a powerful, flexible tool for athletes and coaches of strength sports, or for all inquisitive users.

[1] https://play.google.com/apps/testing/cz.vut.fit.xklemr00.vbt.app
[2] https://groups.google.com/g/dyno-coach-testing

Bibliography

[1] Alliance, A. and Research, I. *Annual Disability and Activity Survey* online. English Federation of Disability Sport, 2022. Available at: https://www.activityalliance.org.uk/assets/000/004/364/Annual_Survey_full_research_report_2021-22_original.pdf. [cit. 2024-01-01].

[2] Andriluka, M.; Pishchulin, L.; Gehler, P. and Schiele, B. 2D Human Pose Estimation: New Benchmark and State of the Art Analysis. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, p. 3686–3693.

[3] Anwar, S.; Hwang, K. and Sung, W. Structured Pruning of Deep Convolutional Neural Networks. *J. Emerg. Technol. Comput. Syst.* New York, NY, USA: Association for Computing Machinery, feb 2017, vol. 13, no. 3. ISSN 1550-4832. Available at: https://doi.org/10.1145/3005348.

[4] Brooks, T.; Fleming, W.; Munger, L. and Munger, K. *Velocity-Based Training: Current Concepts and Future Directions* online. 2022. Available at: https://digitalscholarship.unlv.edu/scholarship_kin/vol3/iss1/1/. [cit. 2024-04-02].

[5] Cao, Z.; Hidalgo, G.; Simon, T.; Wei, S.-E. and Sheikh, Y. OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021, vol. 43, no. 1, p. 172–186.

[6] Deng, J.; Dong, W.; Socher, R.; Li, L.; Li, K. et al. ImageNet: A large-scale hierarchical image database. In: *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*. IEEE Computer Society, 2009, p. 248–255. Available at: https://doi.org/10.1109/CVPR.2009.5206848.

[7] Devries, T. and Taylor, G. W. Improved Regularization of Convolutional Neural Networks with Cutout. *ArXiv*, 2017, abs/1708.04552. Available at: https://api.semanticscholar.org/CorpusID:23714201.

[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*, 2023, vol. 45, no. 6, p. 7157–7173.

[9] Fang, H.-S.; Xie, S.; Tai, Y.-W. and Lu, C. RMPE: Regional Multi-person Pose Estimation. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, p. 2353–2362.

[10] Farhan, S.; Zaidi, S. F.; Munam, A.; Shah, M.; Kamran, M. et al. A Survey on Security for Smartphone Device. *International Journal of Advanced Computer Science and Applications*, april 2016, vol. 7, p. 206–219.

[11] Felzenszwalb, P. F. and Huttenlocher, D. P. Pictorial Structures for Object Recognition. *International Journal of Computer Vision*, Jan 2005, vol. 61, no. 1, p. 55–79. ISSN 1573-1405. Available at: https://doi.org/10.1023/B:VISI.0000042934.15159.49.

[12] Geng, Z.; Wang, C.; Wei, Y.; Liu, Z.; Li, H. et al. Human Pose as Compositional Tokens. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023* online. IEEE, 2023, p. 660–671. Available at: https://doi.org/10.1109/CVPR52729.2023.00071. [cit. 2024-04-05].

[13] Gkioxari, G.; Arbeláez, P.; Bourdev, L. and Malik, J. Articulated Pose Estimation Using Discriminative Armlet Classifiers. In: *2013 IEEE Conference on Computer Vision and Pattern Recognition*. 2013, p. 3342–3349.

[14] González Badillo, J. J.; Sánchez Medina, L.; Ribas Serna, J. and Rodríguez Rosell, D. Toward a New Paradigm in Resistance Training by Means of Velocity Monitoring: A Critical and Challenging Narrative. *Sports Medicine - Open* online, Sep 2022, vol. 8, no. 1, p. 118. ISSN 2198-9761. Available at: https://doi.org/10.1186/s40798-022-00513-z. [cit. 2024-03-31].

[15] González Badillo, L. Movement Velocity as a Measure of Loading Intensity in Resistance Training. *Int J Sports Med.* 2010/02/23th ed., Apr 2010, vol. 31, no. 05, p. 347–352. ISSN 0172-4622. Available at: https://doi.org/10.1055/s-0030-1248333.

[16] Goodfellow, I.; Bengio, Y. and Courville, A. *Deep Learning* online. MIT Press, 2016. Available at: http://www.deeplearningbook.org. [cit. 2024-03-03].

[17] Google. *Accessibility – Material Design 3* online. Google LLC. Available at: https://m3.material.io/foundations/accessible-design/patterns. [cit. 2024-01-01].

[18] Google. *Volley overview* online. Google LLC, february 2022. Available at: https://google.github.io/volley/. [cit. 2024-01-04].

[19] Google. *Accessibility* online. Google LLC, 2023. Available at: https://developer.android.com/design/ui/mobile/guides/foundations/accessibility. [cit. 2024-01-01].

[20] Google. *JetPack Compose UI App Development Toolkit - Android Developers* online. Google LLC, march 2023. Available at: https://developer.android.com/jetpack/compose. [cit. 2024-01-02].

[21] Google. *Layouts in views* online. Google LLC, september 2023. Available at: https://developer.android.com/develop/ui/views/layout/declaring-layout. [cit. 2024-01-02].

[22] Google. *Overview of measuring app performance* online. Google LLC, 2023. Available at:

https://developer.android.com/topic/performance/measuring-performance. [cit. „2024-01-01].

[23] Gordon, C. C.; Blackwell, C. L.; Bradtmiller, B.; Parham, J. L.; Barrientos, P. et al. *2012 Anthropometric Survey of U.S. Army Personnel: Methods and Summary Statistics.* NATICK/TR-15/007. Natick Soldier Research, Development and Engineering Center, 2015.

[24] Gotcher, C. and Austin, B. *Heavy lifting and heart Health* online. December 2016. Available at: https://startingstrength.com/article/heavy-lifting-and-heart-health. [cit. 2024-03-30].

[25] Gough, C. *Americans engaged in sports and exercise per day US 2022 | Statista.* Statista, august 2023. Available at: https://www.statista.com/statistics/189562/daily-engagement-of-the-us-poppulation-in-sports-and-exercise/.

[26] Güler, R. A.; Neverova, N. and Kokkinos, I. DensePose: Dense Human Pose Estimation In The Wild. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* online, 2018, p. 7297–7306. Available at: https://doi.org/10.1109/CVPR.2018.00762. [cit. 2024-04-07].

[27] Gupta, N.; Khosravy, M.; Patel, N.; Dey, N.; Gupta, S. et al. Economic data analytic AI technique on IoT edge devices for health monitoring of agriculture machines. *Applied Intelligence*, Nov 2020, vol. 50, no. 11, p. 3990–4016. ISSN 1573-7497. Available at: https://doi.org/10.1007/s10489-020-01744-x.

[28] Hasan, M. R. M. and Alani, N. H. S. A Comparative Analysis Using Silhouette Extraction Methods for Dynamic Objects in Monocular Vision. *Cloud Computing and Data Science*, january 2022, vol. 3, no. 1, p. 1–12. Available at: https://ojs.wiserpub.com/index.php/CCDS/article/view/1201.

[29] He, K.; Zhang, X.; Ren, S. and Sun, J. Deep Residual Learning for Image Recognition. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 2016, p. 770–778. ISSN 1063-6919.

[30] Hinton, G.; Vinyals, O. and Dean, J. *Distilling the Knowledge in a Neural Network.* 2015.

[31] Hirschorn, O. and Avidan, S. Pose Anything: A Graph-Based Approach for Category-Agnostic Pose Estimation. *CoRR* online, 2023, abs/2311.17891. Available at: https://doi.org/10.48550/ARXIV.2311.17891. [cit. 2024-04-06].

[32] Ionescu, C.; Papava, D.; Olaru, V. and Sminchisescu, C. Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments. *IEEE Trans. Pattern Anal. Mach. Intell.* online, 2014, vol. 36, no. 7, p. 1325–1339. Available at: https://doi.org/10.1109/TPAMI.2013.248. [cit. 2024-05-16].

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

[34] Jiang, H.; Gonnot, T.; Yi, W.-J. and Saniie, J. Computer vision and text recognition for assisting visually impaired people using Android smartphone. In: *2017 IEEE International Conference on Electro Information Technology (EIT)*. 2017, p. 350–353. 2017 IEEE International Conference on Electro Information Technology (EIT). ISSN 2154-0373. Available at: https://doi.org/10.1109/EIT.2017.8053384.

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

[36] King, A.; Kwan, K.; Jukic, I.; Zinn, C. and Helms, E. Fueling for and recovering from resistance training: The periworkout nutrition practices of competitive powerlifters. *Nutrition* online, 2024, vol. 122. ISSN 0899-9007. Available at: https://doi.org/https://doi.org/10.1016/j.nut.2024.112389. [cit. 2024-03-22].

[37] Krizhevsky, A.; Sutskever, I. and Hinton, G. E. ImageNet Classification with Deep Convolutional Neural Networks. In: Pereira, F.; Burges, C.; Bottou, L. and Weinberger, K., ed. *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2012, vol. 25. Available at: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[38] LeCun, Y.; Bengio, Y. and Hinton, G. Deep learning. *Nature*, May 2015, vol. 521, no. 7553, p. 436–444. ISSN 1476-4687. Available at: https://doi.org/10.1038/nature14539.

[39] Li, Y.; Yang, S.; Liu, P.; Zhang, S.; Wang, Y. et al. SimCC: A Simple Coordinate Classification Perspective for Human Pose Estimation. In: *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part VI*. Berlin, Heidelberg: Springer-Verlag, 2022, p. 89–106. ISBN 978-3-031-20067-0. Available at: https://doi.org/10.1007/978-3-031-20068-7_6.

[40] 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.

[41] Lin, T.-Y.; Patterson, G.; Ronchi, M. R.; Cui, Y.; Maire, M. et al. *COCO - Common Objects in context* online. Available at: https://cocodataset.org. [cit. 2024-04-06].

[42] Longo, U. G.; De Salvatore, S.; Carnevale, A.; Tecce, S. M.; Bandini, B. et al. Optical Motion Capture Systems for 3D Kinematic Analysis in Patients with Shoulder Disorders. *Int J Environ Res Public Health*, september 2022, vol. 19, no. 19.

[43] Malone, K. *Heart rate & weight training* online. November 2019. Available at: https://www.livestrong.com/article/382612-heart-rate-weight-training/. [cit. 2024-03-30].

[44] Mayrhofer, R.; Stoep, J. V.; Brubaker, C. and Kralevich, N. The Android Platform Security Model. *ACM Trans. Priv. Secur.* New York, NY, USA: Association for Computing Machinery, apr 2021, vol. 24, no. 3. ISSN 2471-2566. Available at: https://doi.org/10.1145/3448609.

[45] Molchanov, P.; Tyree, S.; Karras, T.; Aila, T. and Kautz, J. Pruning Convolutional Neural Networks for Resource Efficient Inference. *CoRR*, 2017, abs/1611.06440, no. 5. Available at: http://arxiv.org/abs/1611.06440.

[46] Nagel, M.; Fournarakis, M.; Amjad, R. A.; Bondarenko, Y.; Baalen, M. van et al. A White Paper on Neural Network Quantization. *CoRR*, 2021, abs/2106.08295. Available at: https://arxiv.org/abs/2106.08295.

[47] Olusola Olajide, A.; Omotayo, A.; Adebola, O.; Omomule, T. and Orimoloye, S. Performance Evaluation of Native and Hybrid Android Applications. *Communications on Applied Electronics*, may 2018, vol. 7.

[48] Pan, S. J. and Yang, Q. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 2010, vol. 22, no. 10, p. 1345–1359.

[49] Parage, G. *Strategic Plan 2019 -2024* online. International Powerlifting Federation, 31. january 2020. Available at: https://www.powerlifting.sport/fileadmin/ipf/data/about-ipf/strategic-plan/IPF-StrategicPlan_2019-2024.pdf. [cit. 2024-03-22].

[50] Parry, A. *Programming for Powerlifting: Complete Guide with Practical Examples* online. July 2023. Available at: https://characterstrength.co.uk/post/programming-for-powerlifting-complete-guide-with-practical-examples#htoc-principles-of-powerlifting-programming. [cit. 2024-03-24].

[51] Radenković, L. and Nešić, L. The physics of powerlifting. *European Journal of Physics* online. IOP Publishing, mar 2018, vol. 39, no. 3, p. 034002. Available at: https://doi.org/10.1088/1361-6404/aaa90e. [cit. 2024-02-14].

[52] Randhawa, T. S. *Mobile Applications*. 1st ed. Cham: Springer International Publishing, 2022. ISBN 3030023893.

[53] Rohith, M.; Sunil, A. and Mohana. Comparative Analysis of Edge Computing and Edge Devices: Key Technology in IoT and Computer Vision Applications. In: *2021 International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT)*. 2021, p. 722–727. 2021 International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT). ISBN 978-1-6654-3559-8. Available at: https://doi.org/10.1109/RTEICT52294.2021.9573996.

[54] Sarabia, J. M.; Moya Ramón, M.; Hernández Davó, J. L.; Fernandez Fernandez, J. and Sabido, R. The effects of training with loads that maximise power output and individualised repetitions vs. traditional power training. *PLOS ONE* online. Public Library of Science, october 2017, vol. 12, no. 10, p. 1–14. Available at: https://doi.org/10.1371/journal.pone.0186601. [cit. 2024-05-16].

[55] Schwaber, K. and Sutherland, J. *The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game* online. Nov 2020. Available at: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf. [cit. 2024-05-16].

[56] Shi, B.; Bai, X. and Yao, C. An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition. *CoRR*, 2015, abs/1507.05717, no. 61222308. Available at: http://arxiv.org/abs/1507.05717.

[57] SIGNORE, N. *Creating powerful athletes using Velocity-Based training* online. Human Kinetics. Available at: https://us.humankinetics.com/blogs/strength-conditioning-fitness/creating-powerful-athletes-using-velocity-based-training. [cit. 2024-03-29].

[58] Sinha, S. *State of IoT 2023: Number of connected IoT devices growing 16% to 16.7 billion globally.* IoT Analytics GmbH., Aug 2023. Available at: https://iot-analytics.com/number-connected-iot-devices/.

[59] Smith, T. *Habits of successful powerlifters — raw strength* online. September 2019. Available at: https://www.rawstrength.co.uk/blogpost/habits-of-successful-powerlifters9252019. [cit. 2024-03-30].

[60] Square, I. *Overview - OKHTTP* online. Block, Inc., 2022. Available at: https://square.github.io/okhttp/. [cit. 2024-01-04].

[61] Square, I. *RetroFIt* online. Block, Inc., 2023. Available at: https://square.github.io/retrofit/. [cit. 2024-01-04].

[62] Szeliski, R. *Computer vision: Algorithms and applications.* 2nd ed. Springer Nature, 2022.

[63] Tenenbaum, G. and Driscoll, M. P. *Methods of research in sport sciences: Quantitative and qualitative approaches.* Meyer & Meyer Verlag, march 2006. ISBN 1841261335.

[64] Travis, S. K.; Mujika, I.; Gentles, J. A.; Stone, M. H. and Bazyler, C. D. Tapering and peaking Maximal strength for powerlifting performance: A review. *Sports* online, september 2020, vol. 8, no. 9, p. 125. Available at: https://doi.org/10.3390/sports8090125. [cit. 2024-02-14].

[65] Valle, C. *Complete guide to bar Speed Trackers* online. July 2023. Available at: https://www.strongerbyscience.com/complete-guide-to-bar-speed-trackers/. [cit. 2024-03-30].

[66] Venkat, R. *Weightlifting in Olympics: Everything you need to know* online. February 2023. Available at: https://olympics.com/en/news/weightlifting-olympics-rules-history-snatch-clean-and-jerk. [cit. 2024-03-22].

[67] Votel, R. and Li, N. *NextGeneration Pose Detection with MoveNet and TensorFlow.js* online. Google LLC. Available at: https://blog.tensorflow.org/2021/05/next-generation-pose-detection-with-movenet-and-tensorflowjs.html. [cit. 2024-02-18].

[68] Wang, P.; Chen, P.; Yuan, Y.; Liu, D.; Huang, Z. et al. Understanding Convolution for Semantic Segmentation. In: *2018 IEEE Winter Conference on*

Applications of Computer Vision, WACV 2018, Lake Tahoe, NV, USA, March 12-15, 2018 online. IEEE Computer Society, 2018, p. 1451–1460. Available at: https://doi.org/10.1109/WACV.2018.00163. [cit. 2024-05-14].

[69] WHO, W. H. O. *Disability* online. World Health Organization: WHO, 2023. Available at: https://www.who.int/news-room/fact-sheets/detail/disability-and-health. [cit. 2024-01-01].

[70] Wu, Y.; Kirillov, A.; Massa, F.; Lo, W.-Y. and Girshick, R. *Detectron2* online. 2019. Available at: https://github.com/facebookresearch/detectron2. [cit. 2024-02-27].

[71] Xu, L.; Jin, S.; Zeng, W.; Liu, W.; Qian, C. et al. Pose for Everything: Towards Category-Agnostic Pose Estimation. In: Avidan, S.; Brostow, G. J.; Cissé, M.; Farinella, G. M. and Hassner, T., ed. *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part VI* online. Springer, 2022, vol. 13666, p. 398–416. Lecture Notes in Computer Science. Available at: https://doi.org/10.1007/978-3-031-20068-7_23. [cit. 2024-04-02].

[72] Yang, L.; Song, Q.; Wang, Z. and Jiang, M. Parsing R-CNN for Instance-Level Human Analysis. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, p. 364–373.

[73] Yu, H.; Xu, Y.; Zhang, J.; Zhao, W.; Guan, Z. et al. AP-10K: A Benchmark for Animal Pose Estimation in the Wild. In: *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*. 2021.

[74] Zhang, X.; Feng, S. and Li, H. The Effect of Velocity Loss on Strength Development and Related Training Efficiency: A Dose–Response Meta–Analysis. *Healthcare* online, 2023, vol. 11, no. 3. ISSN 2227-9032. Available at: https://doi.org/10.3390/healthcare11030337. [cit. 2024-03-29].

[75] Zhu, W.; Ma, X.; Liu, Z.; Liu, L.; Wu, W. et al. MotionBERT: A Unified Perspective on Learning Human Motion Representations. In: *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2023, p. 15039–15053.

Appendix A

Code References

Listing 15 Repetition detection from velocity data. The threshold for minimal magnitude of movement and minimal repetition duration is calculated to filter out noise and irrelevant movements.

```python
x = np.arange(v.shape[0])
reps = []
roots = []

# Determine the threshold based on the amplitude of the signal
amplitude = max(v) - min(v)
delta = amplitude * 0.1
zero_at = None

for i in range(1, len(v)):
    # Detect zero crossing
    if np.sign(v[i - 1]) != np.sign(v[i]):
        zero_at = i

    # If the value in/decreased enough, approve the zero crossing and reset
    if zero_at and abs(v[i] - v[zero_at]) > delta:
        roots.append(zero_at)
        zero_at = None

roots.append(len(v) - 1)
if len(roots) > 1:
    average_duration = np.mean(np.diff(x[roots]))
    min_duration = average_duration * 0.5
    start_index = roots[0]

    for i in range(0, len(roots), 2):
        if (x[roots[i]] - x[roots[i - 1]]) > min_duration:
            end_index = roots[i]
            if end_index == len(v):
                end_index = roots[-1]
            reps.append(v[start_index:end_index])

            # Set up the start index for the next rep
            start_index = end_index
return np.array(reps, dtype=object)
```

Appendix B

Android Application Use Case Diagram

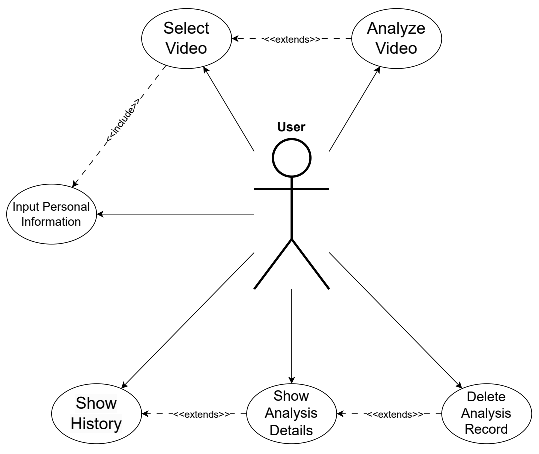


Figure B.1: Use Case Diagram for user's interactions in the application.

Appendix C

User Evaluation and Testing Methodology

In all types of sessions, users were asked to comment on their expectations before taking action and (dis)satisfaction with the result. This section is a formal version of verbal descriptions of user evaluation transcriptions present in attached media. There were three types of user evaluation sessions:

1. Proposal User Evaluation – This session type was conducted at the very beginning with MVP version of application. The application provided video selection and upload to the server, processing and data simple visualization. History was mocked.
2. Iterative User Testing – Many relevant remarks from the initial user evaluation were integrated. In this type of sessions, users should tell if the issues are now resolved. Also they were asked for further ideas or notes. Some users attended multiple iterative user testing sessions.
3. Final evaluation – Users were asked to give feedback on the final design and functionality of the application.

The first type of user evaluation was showing signs, that users do not know what to expect and thus giving only few ideas for improvements. The main focus of user was watched by UI with prepared visual elements for planned functionality, but which was not implemented at that time. The most valuable result was to be sure the UI elements are responsive and do some action if present. Also some ambiguity in button names or labels was discovered.

The second phase brought many new ideas from users. Some of them were integrated during the development, other relevant not operational thoughts are expanded in Section 7.9.

Third type of session produced affirmative results in essence. Users has been made familiar with the final version of the application. Users declared they would use such application to get better understanding of their training session and to support their current methods of evaluation. From one coach's perspective, we obtained the opinion, he would like to use the applicaiton for managing fatigue drop for his clients.

Appendix D

Mobile Devices Used in Experiments

Table D.1: Summary of devices used in experiments and evaluation sessions. Focal length is important because it has an impact on perspective and field of view and it affects the distances traveled in the capture hence impacting the velocity and acceleration.

	model	focal length [mm]	Qualisys	In the wild
Samsung	S22 Ultra	23	✓	✓
	Note 9	26	✓	✓
	Z Flip 4	24	✓	✓
Apple	iPhone X	28	✓	✓
	iPhone SE (2020)	28	✓	✓
Google	Pixel 6	25	✗	✓
Catepillar	CAT S60	unknown	✗	✓
LG	Nexus 5X	unknown	✗	✓
Realme	10	27	✓	✗
Xiaomi	Mi 5 Pro	29.3	✓	✓

Appendix E

Mobile Inference Setup

The setup of mobile inference using MMDeploy for experiments described in Section 7.4 was cumbersome. For that reason, a guide to pursue is presented. The complete setup includes the following steps:

- Clone example repository - PoseTracker-Android-Prototype.
- Download OpenCV-android SDK and set it as Java module. Disable OpenCV-android Kotlin language plugin.
- Clone the MMDeploy repository. (another project and documentation)[1]
- Download cmake >= 3.14.0 and create a symlink. Download Android NDK >= 19.0 and export its path.
- Export OpenCV-android SDK path.
- Clone the ncnn repository and compile it.
  - Assume proper paths and destinations.
- Build Java API (yet another project and documentation)[2]
  - Assume proper paths and destinations.
  - Warning, do not follow fully the instructions, return back to MMDeploy documentation before actually compiling the MMDeploy SDK for CPU and ncnn.
- Compile MMDeploy SDK for CPU and ncnn. Copy relevant libraries into jniLibs folder as described.
- Build and Run the application.

This procedure, while following the referenced links to the documentations, should lead you into demo application deploying RTMDet and RTMPose models with ncnn backend on Android platform. You can adjust the models either converting your own or on yor own or downloading some ncnn backend model from OpenMMLab Deploee websites[3].

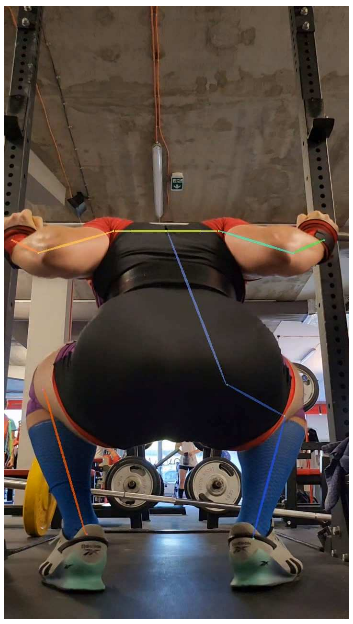
[1] https://github.com/open-mmlab/mmdeploy/blob/dev-1.x/docs/en/01-how-to-build/android.md
[2] https://github.com/open-mmlab/mmdeploy/blob/1.x/csrc/mmdeploy/apis/java/README.md
[3] https://platform.openmmlab.com/deploee

Appendix F

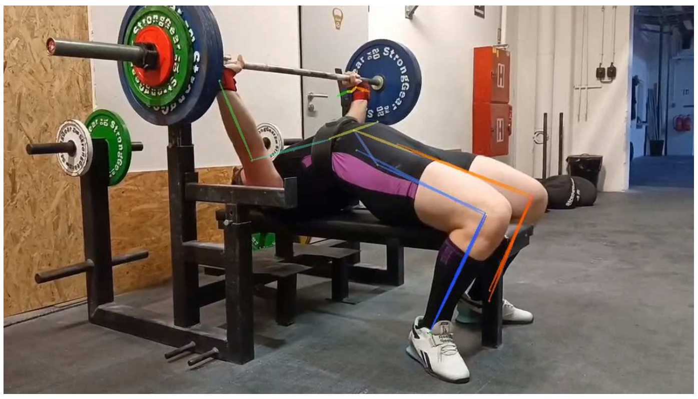
Experiments with Various Human Pose Estimation Models

AlphaPose

Additional example of investigated HPE model. AlphaPose produced mixed quality results for tested scenes as can be seen in Figure F.1.



(a) Result for back squat from lower back close camera position.



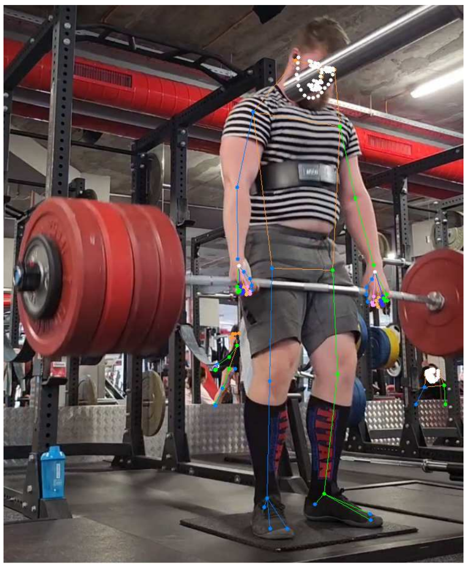
(b) Result for benchpress from parallel medium distance camera position.

Figure F.1: Examples of experiments with AlphaPose inference.

DWPose

DWPose is one of models in MMPose framework model zoo. It aims on RTMPose as a main competitor, while promising higher AP with the same computational complexity as RTMPose in a wholebody task. even though wholebody task is more complex as it covers more details/landmarks on a human body, it is not significant advantage in our context. We investigated the performance anyway.

From experiments done in an OpenXLab online environment[1] as the GitHub documentation[2] suggests, we found out, that indeed wholebody pose estimation did not bring better accuracy in comparison to RTMPose body model. Erected poses are estimated good enough to consider usage in the wild but complex body positions like in benchpress are not estimated properly as you can see in FigureF.2


(a) Erected pose is estimated successfully.

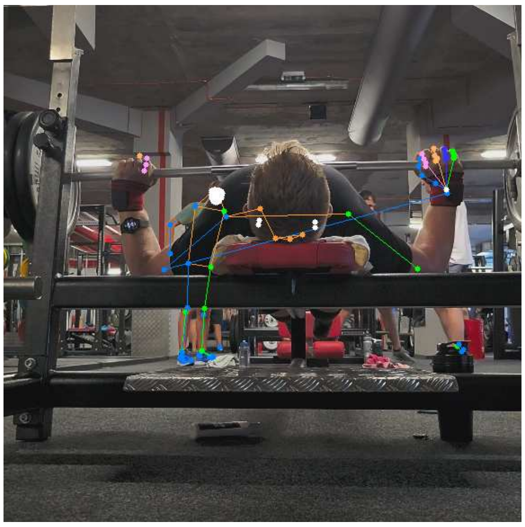

(b) Not satisfactory predictions for benchpress.

Figure F.2: Results from experiments with DWPose model.

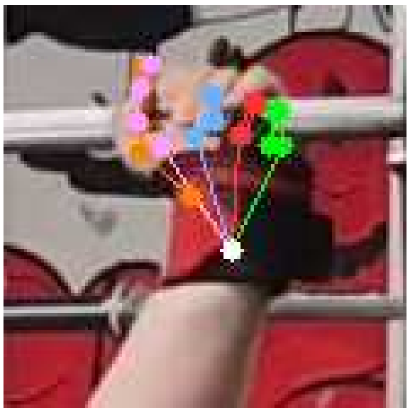
Hand Model

The main issue are occlueded scenes and unnatural body position. The secodn case is also an issue with benchpress exercise. The idea of overcoming this issues it to not detect the whole body but just the wrists with a hand model. Hand models are specialized on detectioon very fine details such as finger parts but this is not an issue for our usage if the model produce reliable resutls.

The results of experiments with the hand model can be seen in Figure F.3. Unfortunately, it is a chicken-and-egg problem. The model is able to predict the correct keypoints for a hand and fingers, but only if the hand is already the main object in the scene, as can be

[1]https://openxlab.org.cn/apps/detail/mmpose/RTMPose
[2]https://github.com/idea-research/dwpose

seen in Figure F.3a. If there is a regular scene with no crop just to the hand, it cannot find the hand properly and cannot predict the keypoints. The unsuccessful prediction is shown in Figure F.3b.


(a) On the cropped image, hand keypoints are correctly predicted.

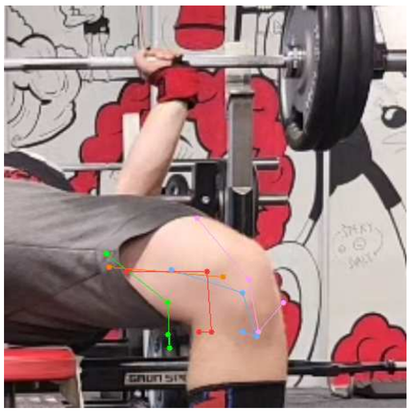

(b) On not cropped image, hand keypoints are incorrectly predicted.

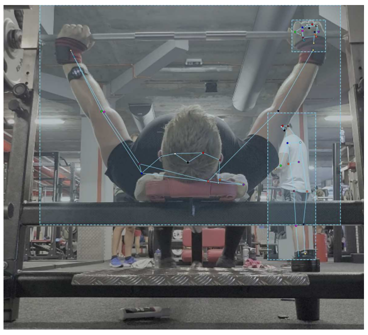
Figure F.3: Experiments with RTMPose model fine-tuned for detailed hand detection.

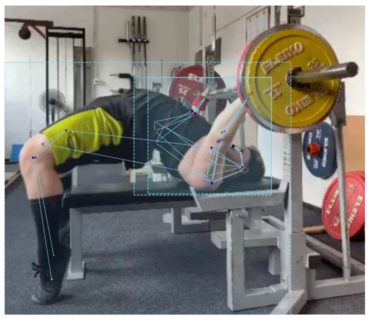
UniPose

Experiments with UniPose were conducted in the OpenXLab online environment[3]. The results were not satisfactory, largely because UniPose is not a traditional human pose estimation model. It uses joint input of an image and a textual prompt describing the type of pose, such as person, animal, car, etc. It is sensitive to bounding box and IoU thresholds, with lower values producing more instances but without a clear best option.

For example, the benchpress from the top view might seem correct, but there is a false prediction that legs are the left arm. As seen in Figure F.4b and F.4c, legs can often be folded into arms.

(a) Even not erected pose is estimated without error – see left wrist.


(b) Result for benchpress are not usable in the production solution.


(c) Results from side view are again not reliable.

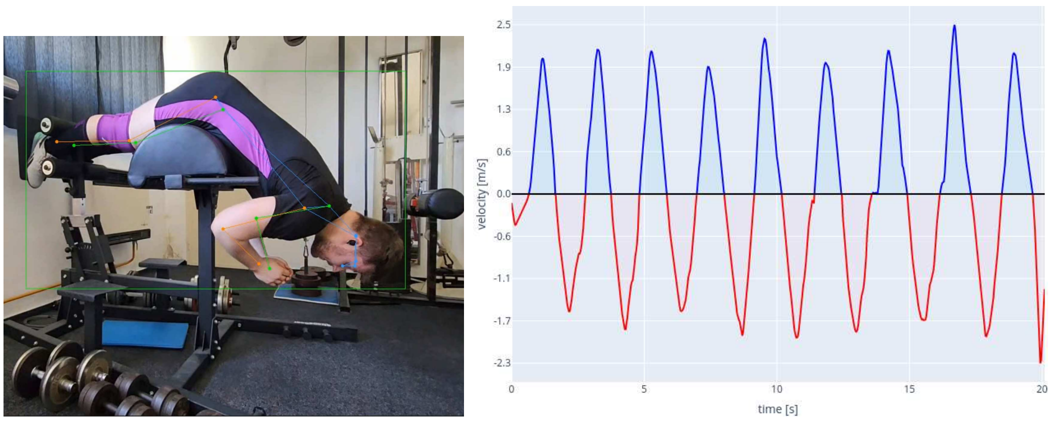
[3]https://openxlab.org.cn/apps/detail/IDEA-Research/IDEA

Appendix G

Experiments with RTMPose on Other Exercises

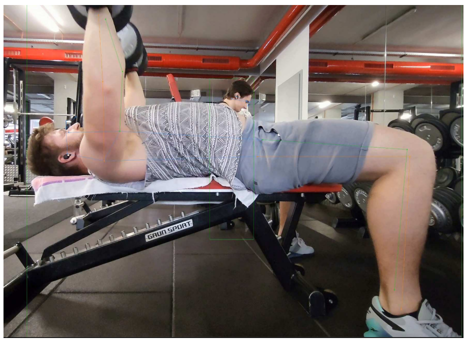
Additional exercises are presented. GHD and hyperextension have similar movements. In shown case in Figure G.1 and G.3 you can see such movements without any additional weight. The dumbbell press (free weight dumbbells) presented in Figure G.2 shows an example of a set with fatigue drop.

Finally, strongman exercise *log lift* is shown in Figure G.4. Please note, that *log loft* consists of three movements: transition from ground onto kneees, transition from knees to upper chest, and final push above head where the "concentric" phase of the movement finishes. After that the *log* is drop back to the ground, this is also observable in the velocity graph by the huge velicyt drop. You can also see significant feature of the processing and this is the asymetric y axis which is extra useful in such exercises.

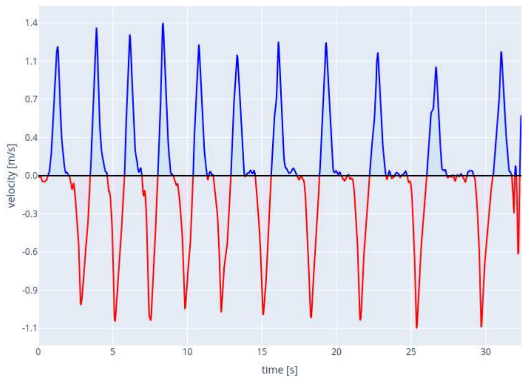


(a) Video snapshot of GHD. (b) Velocity graph with 9 repetitions.

Figure G.1: GHD exercise.

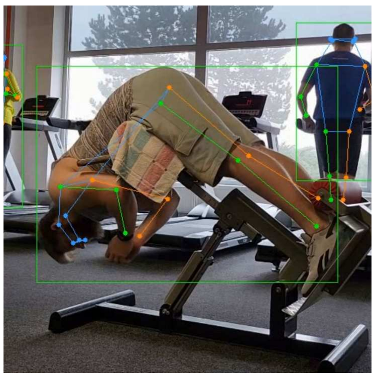


(a) Video snapshot of dumbbell press.

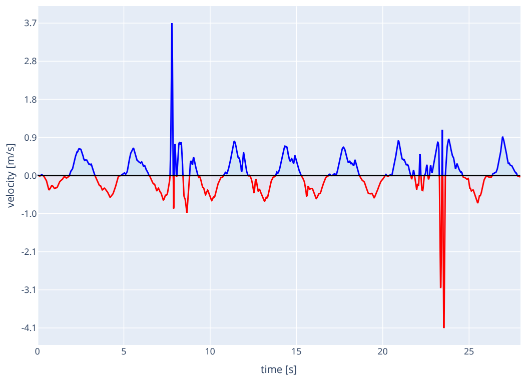


(b) Velocity graph with 11 repetitions.

Figure G.2: Dumbell press exercise. Fatigue drop is observable from velocity loss and prolonged repetition time.

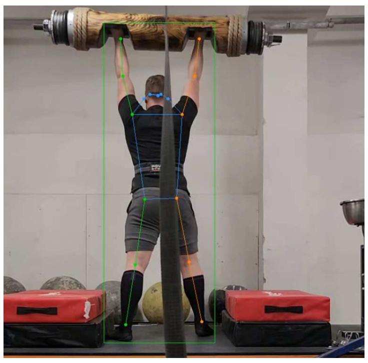


(a) Video snapshot of hyperextension.

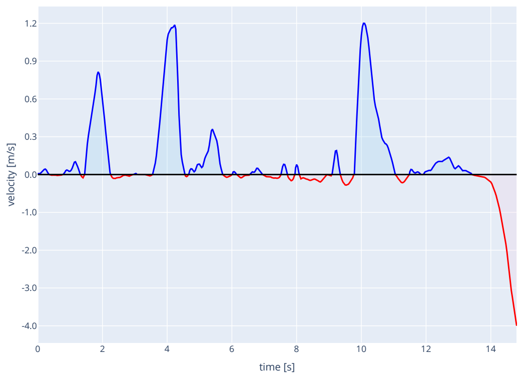


(b) Velocity graph with 9 repetitions.

Figure G.3: Hyperextension exercise. During the video recording, two people went in front of the camera, thus the unexpected spikes in the velocity graph.



(a) Snapshot from annotated video of *log lift*.



(b) Velocity graph with a single repetition.

Figure G.4: Strongman exercise *log lift*. Note the three phases of the exercise. The last one is usually the most crucial.

Appendix H

Camera Angle Consequences

Naive calculation of the distortion implied in the capture of d video footage due to the position and perspective of the camera. No image warping caused by different focal length of camera len is not used.

Please see Figure H.1 where closer position is captured. In Figure H.2 where the athlete is 1 meter farther from camera, significant drop in distorition is observable. The two sections pairs represent OHP and squat exercises.

The red line is denoting the camera view angle bisection. Sections SO and TN are perpendicular to red line bisection and represent the capturem video footage. Sections PO and ON are perpendicular to x axis and represent the actual movement length.

It can be seen the closer the athlete to camera the bigger the distortion. Similar the closer the movement to the edges of the camera view, in other words the farther from camera view center, again the larger the distortion.

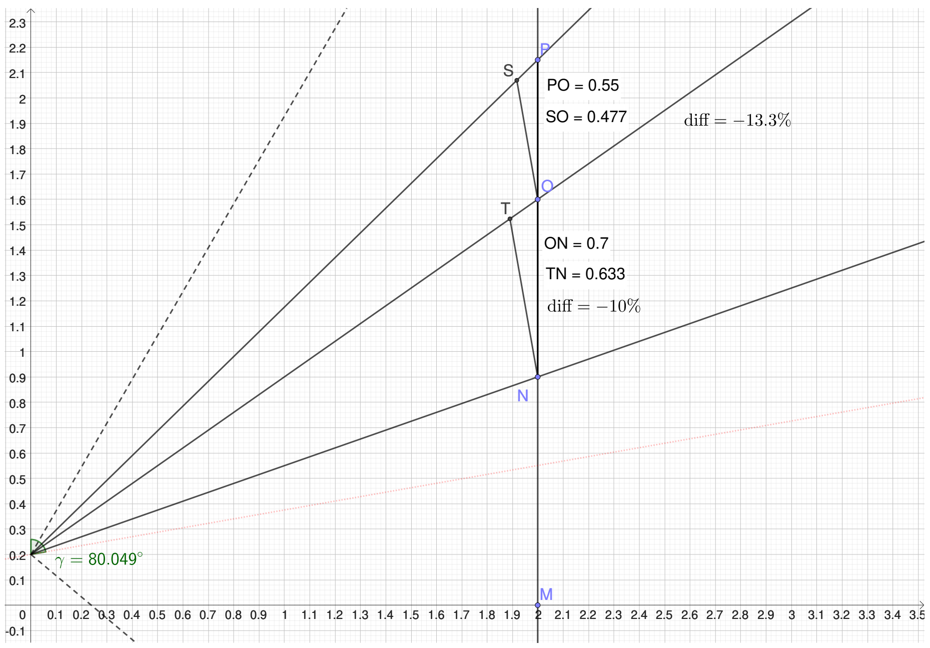


Figure H.1: Impact of camera angle on perceived length of the movement. Distance of camera at 2 meters from athlete. For above head movement, the distortion can get up to −13%.

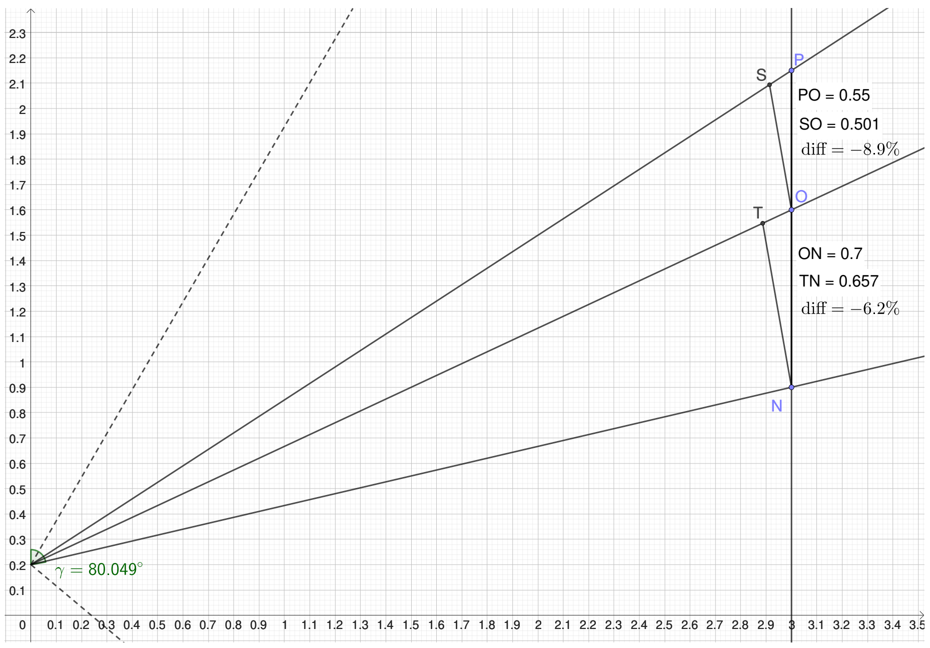


Figure H.2: Impact of camera angle on perceived length of the movement. Distance of camera at 3 meters from athlete. For above head movement, the distortion can get up to −9%.