VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ZAROVNÁNÍ OBRÁZKŮ SPORTOVNÍCH POZIC

ALIGNMENT OF SPORTS POSE IMAGES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JOSEF KNÍŽE**
AUTHOR

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

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Kníže Josef**
Program: Informační technologie
Název: **Zarovnání obrázků sportovních pozic**
Alignment of Sports Pose Images
Kategorie: Zpracování obrazu

Zadání:

1. Seznamte se s problematikou počítačového vidění, zaměřte se na lokalizaci částí lidských těl a na problematiku regrese transformací.
2. Sestavte vhodnou datovou sadu pro učení a vyhodnocení algoritmů pro zarovnání sportovních pozic.
3. Vyvíjejte algoritmy směřující k rychlému a kvalitnímu zarovnání sportovních pozic. Experimentujte jak s metodami dosahujícími maximální přesnosti, tak metodami minimálně náročnými na výpočetní zdroje.
4. Iterativně vylepšujte vyvíjené řešení a používanou datovou sadu. Demonstrujte použitelnost řešení na obrázcích z různých sportů.
5. Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování projektu; vytvořte plakátek a krátké video pro prezentování projektu.

Literatura:

- Goodfellow, Bengio, Courville: Deep Learning, MIT Press, 2016
- Bharath Ramsundar, Reza Bosagh Zadeh: TensorFlow for Deep Learning: From Linear Regression to Reinforcement Learning, O'Reily Media, 2018
- Gary Bradski, Adrian Kaehler: Learning OpenCV; Computer Vision with the OpenCV Library, O'Reilly Media, 2008
- Richard Szeliski: Computer Vision: Algorithms and Applications, Springer, 2011
- Toshev et al.:DeepPose: Human Pose Estimation via Deep Neural Networks, CVPR 2014
- DeTone, D. et al.: Deep Image Homography Estimation, https://arxiv.org/abs/1606.03798
- Andriluka, M. et al.: 2D Human Pose Estimation: New Benchmark and State ofthe Art Analysis, CVPR 2014

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2, značné rozpracování bodů 3 a 4.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Herout Adam, prof. Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 11. května 2022
Datum schválení: 1. listopadu 2021

Abstrakt

Tato práce se zabývá problémem zarovnání dvou obrázků na základě sportovní pozice zaujaté osobou na obrázcích. Hlavním výsledkem je návrh a implementace dvou systémů pro zarovnání fotografií na základě sportovní pozice. První systém je zaměřen na přesnost a byl využit k tvorbě datové sady, která byla dále využita k učení neuronové sítě. Druhý systém cílil na menší hardwarovou náročnost, které bylo dosáhnuto použitím neuronových sítí. Vytvořená neuronová síť úspěšně zarovnala 81,98 % obrázků. V rámci řešení byla vytvořena sada fotek rozřazených podle sportovní pozice.

Abstract

The following thesis deals with image alignment based on sport pose of person in the image. The main result of this thesis is design and implementation of two systems for image alignment based on sport pose. The first system's focus is accuracy and it will be used to create dataset, which will be use to train neural network. Second system aimed at minimal hardware requirement, which was acomplished by using neural networks. Implemented neural net managed to sucessfuly align 81.98 % of images. A set of images, sorted based on the sport pose, has been created as a part of the solution.

Klíčová slova

Zarovnání obrázků, Detekce lidské pózy, Neuronové sítě, Počítačové vidění, Regrese transformací

Keywords

Image alignment, Human pose detection, Neural networks, Computer vision, Transformations regression

Citace

KNÍŽE, Josef. *Zarovnání obrázků sportovních pozic*. Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. Ing. Adam Herout, Ph.D.

Zarovnání obrázků sportovních pozic

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana prof. Ing. Adama Herouta, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Josef Kníže
8. května 2022

Poděkování

Rád bych poděkoval svému vedoucímu panu prof. Ing. Adamu Heroutovi, Ph.D. za vedení této práce, cenné rady a poskytnutí videí k tvorbě datové sady.

Obsah

Kapitola 1

Úvod

Cílem této práce je průzkum a implementace technik počítačového vidění k zarovnání fotografií na základě sportovní pozice. Hlavním využitím výsledného zarovnání je porovnání sportovních pozic a sledování zlepšení či chyb v zaujatých pozicích. Zároveň bude prozkoumána možnost využití konvolučních neuronových sítí k řešení tohoto problému, čímž by bylo možné dosáhnout hardwarové nenáročnosti a výsledný algoritmus by byl využitelný i na mobilních platformách.

Porovnáváním zaujatých pozic s profesionálním sportovcem může začínající sportovec odhalit špatnou techniku a předejít závažnému zranění, které by si mohl přivodit přetěžováním svalů, kloubů či šlach, ke kterému může dojít při nesprávně zaujatých pózách.

První tři následující kapitoly obsahují seznámení s důležitou teorií a pojmy, která budou v této práci využity. Kapitola 2 popisuje obecné postupy zarovnávání obrázků a vyjádření transformace pomocí matic. Jelikož cílem práce je zarovnat obrázky na základě lidské pózy, v kapitole 3 jsou popsány prozkoumané postupy a technologie pro detekci lidských částí těl v obraze. Další následuje kapitola 4, která obsahuje pojmy a principy z oblasti neuronových sítí, se zaměřením na zpracování obrazu a také je v ní popsán aktuální stav výzkumu neuronových sítí, využitých k zarovnání obrázků. Kapitola 5 obsahuje stručný návrh systémů pro zarovnání na základě sportovní pozice, jejichž implementace je podrobněji popsána v kapitole 6. Implementovaný systém vytvořen pomocí neuronové sítě je v kapitole 7 otestován a jsou porovnány jeho různé varianty. Práce je ukončena kapitolou 8, ve které jsou shrnuty výsledky a možné rozšíření.

Kapitola 2

Problematika zarovnávání obrázků

Zarovnání obrázků [21] je jeden ze základních problému v počítačovém vidění. Je to proces, jehož cílem je odhadnou transformační matici tak, aby co největší počet klíčových bodů či pixelů byl po její aplikaci v překryvu. Jelikož se zarovnání na základě pixelů příliš nevyužívá a v rámci této práce nebylo využito, dále bude popsáno pouze zarovnávání na základě klíčových bodů. Zarovnávání je použito například při spojování obrázků (*image stitching*), kde je cílem získat z více obrázků jeden větší, což je základ tvorby panoramat, tvorby satelitních snímků a pořizování fotografií s více objektivy. Další úkony počítačového vidění, ve kterých se zarovnávání obrázků dá využít, je např. stabilizace nebo sumarizace videa.

2.1 Transformace obrázků

Pro zarovnání obrázků je potřeba definovat postupy, pomocí kterých je možné obrázky transformovat. V této části jsou popsány 2D transformace [14, 21] obrázků v počítači a jak je vyjádřit pomocí matic.

V počítačích jsou obrázky reprezentovány jako dvourozměrné pole hodnot, indikující intenzitu daného pixelu. Jedná-li se o barevný obrázek jsou využity 3 pole, reprezentující jednotlivé barevné složky (RGB). Na obrázek tedy můžeme nahlížet jako na dvourozměrný prostor, který má počáteční souřadnici (0, 0) v levém horním rohu, a na jednotlivé pixely jako body, které mají souřadnice x a y.

S definovaným prostorem je možné na jednotlivé pixely aplikovat lineární transformaci, která dokáže mapovat jejich souřadnice na nové. Lineární transformaci lze vyjádřit jako matici a její aplikace je pouhé vynásobení s danými souřadnicemi. V této kapitole stručně popíši jednotlivé části transformací, jak je matematicky vyjádřit a popíši kategorizaci transformací podle stupňů volnosti.

2.1.1 Prvky transformací

Translace

Translace neboli posun je jednoduché přičtení čísla k souřadnici x nebo y. Translaci je možné vyjádřit vektorem posunu $\vec{p} = [p_x, p_y]$ nebo transformační maticí, popsané v rovnici (2.1).

$$\begin{bmatrix} 1 & 0 & p_x \\ 0 & 1 & p_y \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + p_x \\ y + p_y \\ 1 \end{bmatrix} \tag{2.1}$$

Rotace

Rotací se rozumí otočení daného bodu ve směru hodinových ručiček kolem středu souřadného systému o vybraný úhel α. Maticové vyjádření je v rovnici (2.2).

$$M_R = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

Změna měřítka

Maticový zápis lze vidět v rovnici (2.3), kde S_x a S_y jsou koeficienty změny měřítka. Pokud by některý z těchto koeficientů byl záporný, došlo by k převrácení podél osy. V případě záporného koeficientu S_x podél osy y a v případě S_y podél osy x.

$$M_S = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

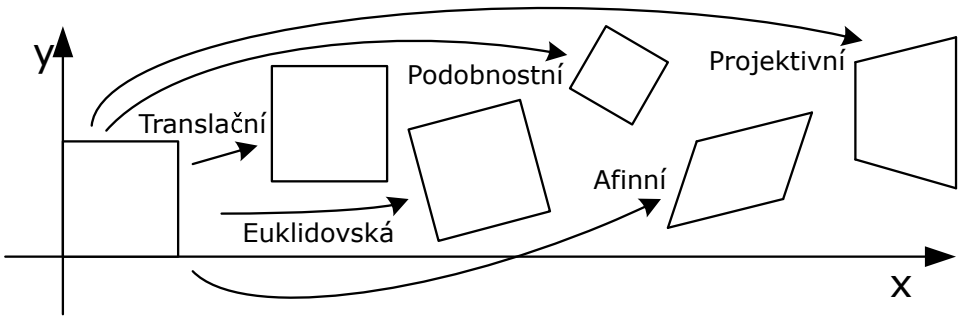
Zkosení

Maticové vyjádření transformace zkosení je v rovnici (2.4), kde jsou dva koeficienty zkosení ve směru os x a y Z_x a Z_y.

$$M_{Sh} = \begin{bmatrix} 1 & Z_x & 0 \\ Z_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

2.1.2 Stupně volnosti a klasifikace transformačních matic

Jelikož různé prvky transformací vyžadují více nezávislých koeficientů a zachovávají různé vlastnosti transformovaných objektů či obrázků, dělí se na základě stupňů volnosti (*Degrees of freedom*), které udávají kolik na sobě nezávislých koeficientů je využito. Dále se dělí do tříd podle toho, jaké vlastnosti zachovávají. Jednotlivé třídy transformací lze vidět na obrázku 2.1.



Obrázek 2.1: Zobrazení jednotlivých tříd transformací. Převzato z [21].

Mezi vlastnosti, které mohou transformace zachovávat patří:

- Orientace
- Délky
- Úhly
- Rovnoběžnost
- Rovnost

Tyto vlastnosti jsou vypsány v pořadí, ve kterém je třídy transformačních matic postupně přestávají zachovávat.

Translační transformace (Translation)

Tato třída matic je nejjednodušší, protože využívá pouze posun, který vyžaduje 2 koeficienty, což znamená že jedná o 2. stupeň volnosti. Zachovává všechny vlastnosti vypsané výše.

Euklidovská transformace (Rigid/Euclidean)

Do další třídy spadá jak posun, tak rotace, jedná se tedy o třídu se 3. stupněm volnosti a zachovává délky a vše následující.

Transformace podobnosti (Similarity)

Transformace podobnosti přidává k možným operacím změnu měřítka, čímž přestane zachovávat délky. Stupeň volnosti této třídy je 4.

Afinní transformace (Affine)

Tato třída transformací přidá zkosení. Z toho plyne že přestane zachovávat i úhly. Jelikož zkosení má dva koeficienty zvedá se stupeň volnosti na 6.

Projektivní transformace (Projective)

Poslední třída umožňuje využití perspektivních transformací. Jedinou vlastnost, kterou v obrazu udržuje, je rovnost čar či hran. Její stupeň volnosti je 8.

2.2 Regrese transformace zarovnání na základě klíčových bodů

Tato část se zabývá regresí transformace zarovnání z párů klíčových bodů [21, 10, 22] detekovaných na zarovnávaných obrázcích.

Pro zarovnání obrázků s detekovanými klíčovými body je potřeba vypočítat transformační matici, pro kterou platí, že po její aplikaci na zdrojový klíčový bod bude daný bod roven cílovému. Vztah $H \times P = P'$ je popsán v rovnici (2.5).

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \times \begin{bmatrix} x_z \\ y_z \\ 1 \end{bmatrix} = \begin{bmatrix} x_c \\ y_c \\ 1 \end{bmatrix} \tag{2.5}$$

V této rovnici je možné vidět zdrojové souřadnice označené spodním indexem z, cílové souřadnice označené spodním indexem c a transformační matici obsahující 9 neznámých h_{11} až h_{33}. Pro výpočet těchto neznámých, potřebujeme rovnici upravit do tvaru ukázaného v rovnici (2.6).

$$\begin{bmatrix} x_{z(i)} & y_{z(i)} & 1 & 0 & 0 & 0 & -x_{c(i)}x_{z(i)} & -x_{c(i)}y_{z(i)} & -x_{c(i)} \\ 0 & 0 & 0 & x_{z(i)} & y_{z(i)} & 1 & -y_{c(i)}x_{z(i)} & -y_{c(i)}y_{z(i)} & -y_{c(i)} \end{bmatrix} \times \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{2.6}$$

Při výpočtu homografie se počítá s pouze s 8 stupni volnosti, a tedy platí že $h_{33} = 1$. K vyjádření zbylých 8 proměnných pouze 2 rovnice nestačí. Musí se proto využít více párů klíčových bodů, přičemž z každého páru jsme schopni vytvořit 2 další rovnice. V případě vyjádření homografie jsou tedy potřeba 4 páry klíčových bodů. Jelikož je za h_{33} doplněna 1, je rovnice upravena do tvaru v rovnici (2.7).

$$\begin{bmatrix} x_{z(1)} & y_{z(1)} & 1 & 0 & 0 & 0 & -x_{c(1)}x_{z(1)} & -x_{c(1)}y_{z(1)} & -x_{c(1)} \\ 0 & 0 & 0 & x_{z(1)} & y_{z(1)} & 1 & -y_{c(1)}x_{z(1)} & -y_{c(1)}y_{z(1)} & -y_{c(1)} \\ & & & & \vdots & & & & \\ x_{z(i)} & y_{z(i)} & 1 & 0 & 0 & 0 & -x_{c(i)}x_{z(i)} & -x_{c(i)}y_{z(i)} & -x_{c(i)} \\ 0 & 0 & 0 & x_{z(i)} & y_{z(i)} & 1 & -y_{c(i)}x_{z(i)} & -y_{c(i)}y_{z(i)} & -y_{c(i)} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.7}$$

Z tohoto tvaru už je možné vyjádřit transformační matici, což lze vidět v rovnici (2.8).

$$H = P^{-1} \times \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.8}$$

2.3 RANSAC

RANSAC (*Random Sample Consensus*) [5] je iterační algoritmus, který slouží k robustnímu odhadu parametrů matematického modelu na základě dat. Na rozdíl od ostatních metod řešících tento problém, kterou je např. metoda nejmenších čtverců využívajících co nejvíce dat, RANSAC jich využívá minimum potřebné k odhadu daného modelu a pokud dostatek zbylých dat odpovídá danému modelu, jsou využity pro výpočet přesnějšího modelu. Jelikož algoritmus využívá pro odhad modelu náhodnou volbu prvků, je nedeterministický, a tedy při průchodu se stejným vstupem nemůže očekávat stejný výstup. Tento algoritmus byl představen v roce 1981 autory M. Fischlerem a R. Bollesem. Jeho použití bylo demonstrováno na určení místa pořízení leteckých fotografií na základě orientačních bodů v dané fotografii [7]. Je založen na možnosti rozdělení vstupních dat do dvou skupin – inliers (záznamy, které spadají pod cílový model) a outliers (záznamy, které jsou chybné)[1]. Cílem algoritmu je outliery vyloučit a vypočítat model pouze z inlierů. Výhodou tohoto algoritmu je, že s velkou přesností odhaduje model i z dat, které obsahují velký počet outlierů a to až k hranici 50 %, kde přestává mít dobré výsledky. Nevýhodou tohoto algoritmu je, že s narůstajícím počtem outlierů se zvedá počet iterací nutných k robustnímu odhadu, které mohou být výpočetně náročnější. Výpočet počtu iterací nutných k robustnímu odhadu je popsán v rovnici (2.9).

$$k = \frac{\log(1-p)}{\log(1-w)^n} \tag{2.9}$$

kde: p = Pravděpodobnost, že alespoň jeden výběr nebude obsahovat outlier
w = Poměr počtu inlierů s celkovým počtem záznamů
n = Počet vybíraných prvků pro odhad modelu

Algorithm 1: RANSAC

Input: (Data, Tolerance τ, Hranice úspěšného modelu D)
Output: *Model*

1. Náhodně zvol minimální počet prvků dat, ze kterých jde odhadnout model.
2. Ze zvolených prvků vytvoř odhad modelu.
3. Spočítej počet záznamů dat, které s tolerancí τ odpovídají modelu.
4. Pokud počet záznamů odpovídající modelu překročil hranici D, vytvoř nový odhad modelu ze všech odpovídajících záznamů. Urči chybu daného modelu. Pokud je chyba menší, než chyba dosavadního nejlepšího modelu, je tento model nový nejlepší.
5. Opakuj kroky 1 až 4 předem určeným počtem iterací.
6. Vrať nejlepší model.

2.3.1 Upravené varianty algoritmu RANSAC

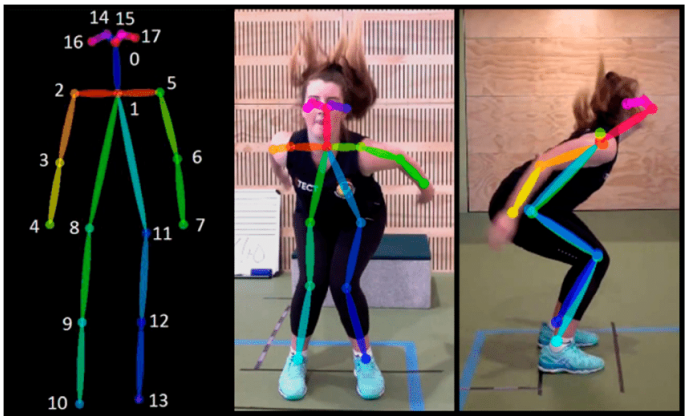
Jelikož je RANSAC široce využíván už od roku 1981, byly vytvořeny různé varianty, které si kladou za cíl vylepšit určité vlastnosti původního algoritmu. Jedním z takových vylepšení může být například deterministické vybírání dat pro odhad modelu. Další varianty například cílí na zmenšení náročnosti 3. kroku, ve kterém validují model pouze nad částí dat a hodnotí daný model poměrem inlierů a outlierů z dané části.

[1] Jelikož neexistuje jednoslovný překlad těchto záznamů, budu dále využívat jejich anglické názvy.

Kapitola 3

Detekce lidské pózy v obrazu

Detekce lidské pózy [2, 25, 1] je další rychle se vyvíjející oblast počítačového vidění. Velký rozvoj zaznamenala především díky širokému využití. Využít se dá například v herním průmyslu, rozšířené realitě, analýze sportu nebo při sledování chodců samořídícími auty. Cílem je z obrazu detekovat 2D nebo 3D souřadnice kloubů, kterými jsou např. ramena, lokty či kolena, jedné či více osob. Na obrázku 3.1 je ukázka 2D detekování pózy, ve které je vidět model obsahující 18 kloubů. Jedná se o model, vytvořený s datovou sadou COCO. Existují i varianty, ve kterých se detekuje obrys člověka, nebo dokonce 3D varianta, ve které se detekuje objem člověka. Mezi časté problémy detekce lidské pózy patří špatná kvalita fotografií či videa, volný oděv nebo zákryt částí těla.



Obrázek 3.1: Ukázka 2D detekce klubů s využitím modelu COCO. Převzato z [12].

3.1 Metody detekce pózy více osob

Jedním z problémů detekce lidské pózy je detekce více osob na jedné fotografii. Ať už se jedná o druhého cvičícího či náhodnou osobu v pozadí, je nutné identifikované klouby rozdělit na základě toho, jaké postavě patří, což značně komplikuje tento úkol. Existují dva přístupy k tomuto problému.

Shora dolů (Top-Down)

Tento přístup v prvním kroku využije už existující detektory osob v obraze, čímž získá ohraničenou část obrazu obsahující jednu osobu, nad kterou může spustit detektor lidské

pózy. Nevýhodou tohoto přístupu je, že počet spouštění detekce pózy je závislý na počtu osob a s narůstajícím počtem osob narůstá i časová náročnost. Na druhou stranu je výhodou možnost využití už existujících a volně dostupných detektorů osob.

Zdola nahoru (Bottom-Up)

Druhý způsob nejdříve detekuje všechny klouby v daném obrázku a dále je roztřídí do skupin, které reprezentují jednotlivé osoby. Tento přístup je obvykle rychlejší na vykonání, ale nevýhodou je vyšší míra nepřesností.

3.2 Hodnocení systémů pro detekci lidské pózy

Jelikož se tato oblast počítačového vidění velice rychle rozrůstá, existuje mnoho systému pro detekci lidské pózy a mnoho dalších bude ještě vytvořeno. Z toho důvodu byly vytvořeny metriky [1, 18] a datové sady, pro jejich hodnocení a porovnávání. Datové sady byly vytvořeny tak, aby pokrývali co nejvíce možných pozic při různých aktivitách, jako je třeba sportování, plavání či hraní na hudební instrumenty. Zároveň existují udržované webové stránky, které obsahují porovnání aktuálních, volně dostupných systémů, nad různými datovými sadami[1].

3.2.1 Metriky hodnocení systému pro detekci lidské pózy

PCP – Percentage of Correct Parts

Tato metrika indikuje procento správně odhadnutých kloubů v končetinách. Za správně odhadnuté umístění kloubu je považováno umístění se vzdáleností menší než 50% délky dané končetiny. Metrika takto počítaná se také označuje takto PCP@0.5, přičemž procento vzdálenosti nemusí být nutně 50 %.

PCPm

U původní metriky PCP nastával problém v tom, že klouby v příliš krátkých končetinách vyžadovaly až příliš velkou přesnost, aby byly považovány za správně odhadnuté. Proto v práci [1] zavedli metriku PCPm, ve které je maximální vzdálenost 50 % průměrné délky končetin z celé datové sady.

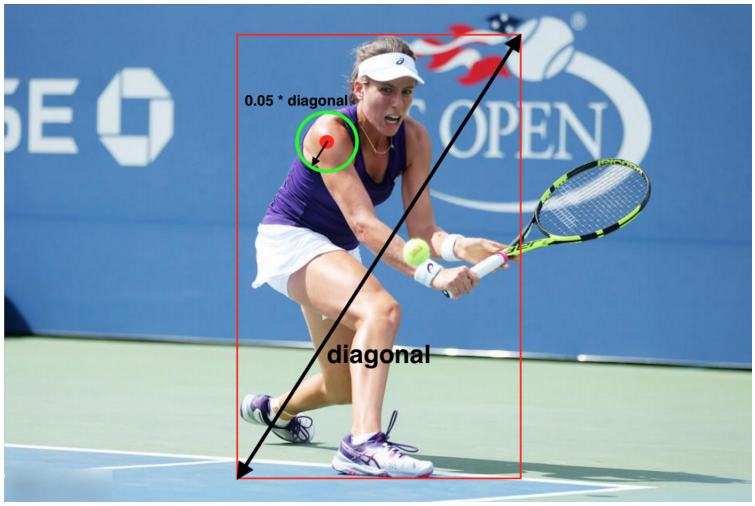
PCK – Percentage of Correct Keypoint

Tato metrika indikuje procento správně odhadnutých kloubů v těle a maximální vzdálenost od očekávané pozice se počítá ze zlomku průměru trupu. V některých případech je místo průměru trupu použita diagonála čtverce ohraničující postavu viz obrázek 3.2.

PCKh

Další upravená metrika vznikla modifikací PCK a opět je popsaná v práci [1], ve které se jako maximální vzdálenost od očekávané pozice využívá 50 % délky segmentu označující hlavu.

[1] https://paperswithcode.com/sota/pose-estimation-on-mpii-human-pose



Obrázek 3.2: Ukázka výpočtu maximální vzdálenosti vypočítané z diagonály ohraničujícího obdélníku využité v metrice PCK. Převzato z [18].

OKS – Object Keypoint Similarity

Tato metrika byla vytvořena pro testování nad datovou sadou COCO. Zaměřuje se především na fakt, že detekované klouby mohou mít různou velikost, a proto byly vytvořeny konstanty, které popisují velikost kloubu, a v kombinaci s velikostí plochy detekovaného člověka lze vypočítat přesnost odhadu pozice daného kloubu pomocí rovnice (3.1) [18]. Výsledek této rovnice je hodnota mezi 0 a 1, kde vyšší hodnota znamená přesnější odhad. Velikosti a tedy i váhy daných kloubů jsou popsány v obrázku 3.3.

$$OKS = \exp\left(-\frac{d_i^2}{2s^2k_i^2}\right) \tag{3.1}$$

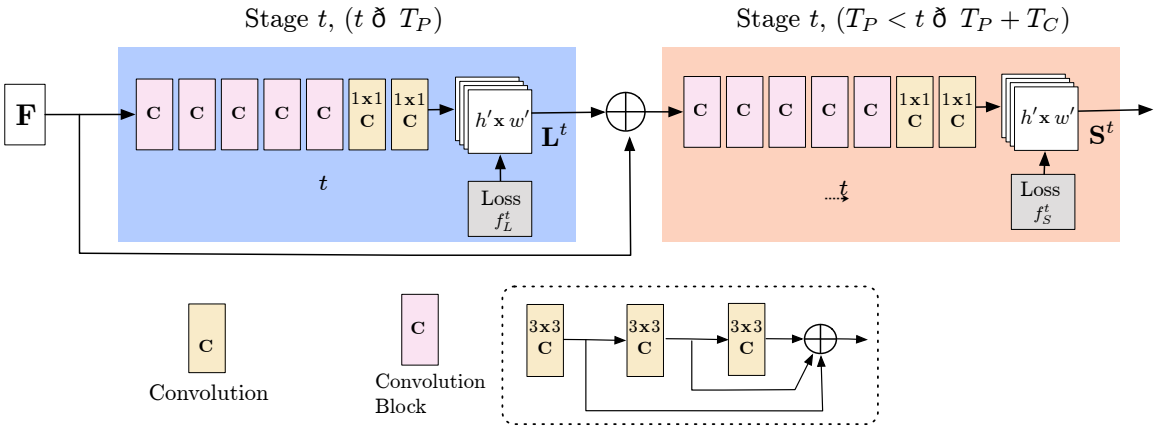
kde: d_i = Vzdálenost odhadnuté pozice kloubu od skutečné pozice
s = Druhá odmocnina velikosti plochy, kterou pokrývá osoba
k_i = Konstanta pro vybraný kloub

Keypoint	k_i
hips	0.107
ankles	0.089
knees	0.087
shoulders	0.079
elbows	0.072
wrists	0.062
ears	0.035
nose	0.026
eyes	0.025

Obrázek 3.3: Konstanty jednotlivých kloubů pro výpočet přesnosti odhadu. Převzato z [18].

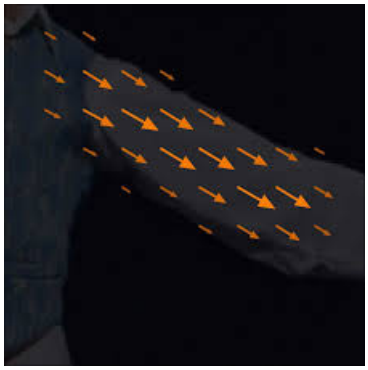
3.3 OpenPose

OpenPose [4] je volně dostupná knihovna pro nekomerční využití, využívající hlubokých neuronových sítí s více etapy (stage). Využívá přístup zdola nahoru, jak bude níže popsáno.



Obrázek 3.4: Architektura OpenPose, ukazující dvě sady etap. Modré etapy predikují *Part Affinity Fields* a béžové etapy predikují teplotní mapy jednotlivých kloubů. Převzato z [4].

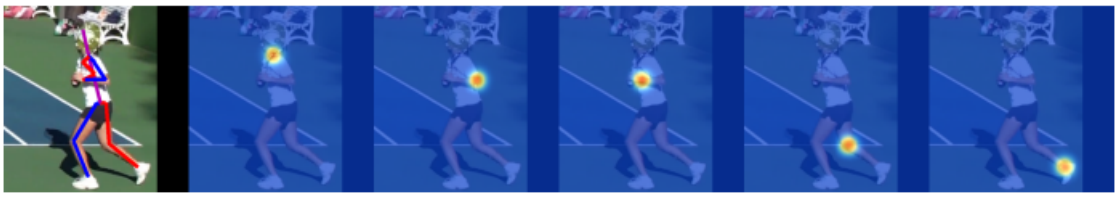
OpenPose architekturu tvoří dvě sady etap, které je možné vidět na obrázku 3.4. První sada etap predikuje 2D vektory, které tvoří *Part Affinity Fields* (pole vektorů indikující vztahy částí těla) viz obrázek 3.5, na základě kterých budou detekované klouby rozděleny do shluků, tvořící jednotlivé osoby na obrazu.



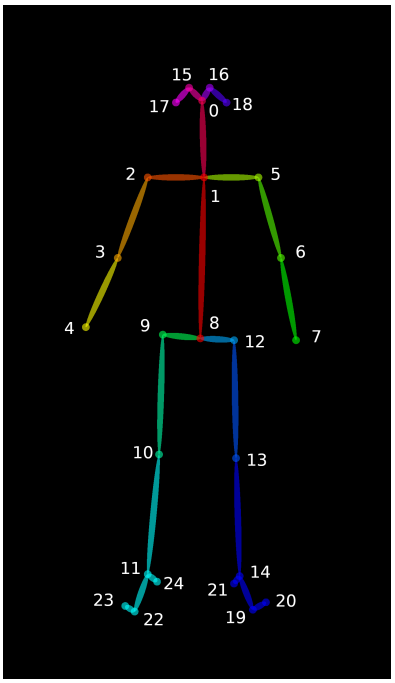
Obrázek 3.5: Ukázka pole vektorů indikující spřízněnost kloubů (*Part Affinity Field*). Převzato z [4].

Druhá sada etap predikuje teplotní mapy (*heat maps*). Teplotní mapy jsou šedotónové obrazy, ve kterých vyšší hodnoty indikují vyšší pravděpodobnost umístění jednotlivých kloubů. Pro jednotlivé klouby musí být tvořeny oddělené teplotní mapy. Ukázku teplotních map pro jednotlivé klouby je možné vidět v obrázku 3.6.

V rámci vývoje OpenPose byl navržen nový model obsahující 25 kloubů, který k původním modelům přidal 3 klouby do každého chodidla a jeden kloub reprezentující pánev. Tento model je možné vidět na obrázku 3.7 Kromě klasické detekce pózy těla, umožňuje OpenPose také detekci kloubů ve dlani a klíčových bodů v obličeji.



Obrázek 3.6: Ukázka teplotních map, indikujících pozice kloubů. Převzato z [16].



Obrázek 3.7: Model kloubů nazvaný Body25 vytvořen v rámci OpenPose. Převzato z [4].

Tato knihovna patří mezi nejpopulárnější především proto, že podporuje různé operační systémy, je snadná na použití a podporuje různé vstupy, jako třeba fotografie, složky obsahující fotografie, videa nebo živý záznam z webkamery. Zároveň je schopná detekovat všechen potřebný hardware jako grafické karty či webkamery. Stejně jako u vstupu poskytuje různé možnosti výstupu. Výstup může být prezentován za běhu detekce, při kterém se mohou přepínat různé způsoby zobrazení, například je možnost zobrazovat teplotní mapy pro jednotlivé klouby místo celé detekované kostry, nebo existuje možnost mít na pozadí vstupní obrázek. Zároveň umožňuje vše ukládat na disk, včetně pozic kloubů exportovaných ve formátu JSON.

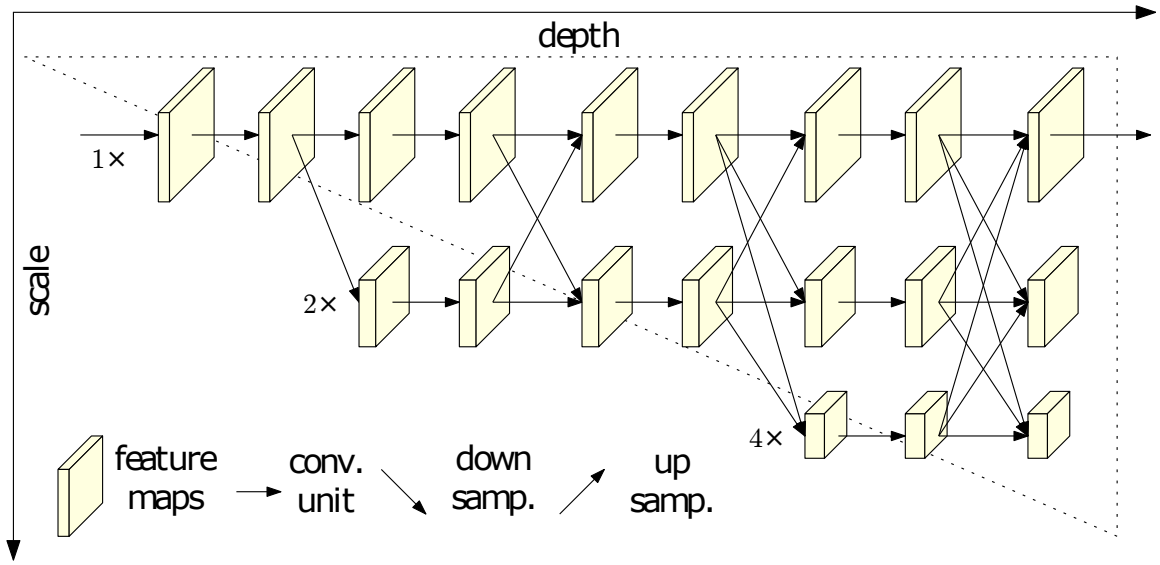
3.4 DeepPose

DeepPose byl jedním z prvních projektů, který využíval konvoluční neuronové sítě k detekci lidské pózy. Článek, ve kterém byl popsán [23], byl vydán už v roce 2013. Jedná se o rekurentní síť, která v první iteraci detekuje pózu z celého obrázku, který kvůli výpočetní náročnosti musí být převeden do rozlišení 220×220 a v následujících iteracích pracuje pouze

s částí původního obrázku, ve které byl detekován klíčový bod, čímž dokáže využít detail z plného rozlišení obrázku.

3.5 High-Resolution Net (HRNet)

Tato neuronová síť[2] k detekci lidské pózy dosáhla velmi dobrých výsledků, především kvůli novému přístupu, který zachová plné rozlišení vstupních obrázků. Architektura sítě začíná s jednou větví, ve které má obrázek plné rozlišení a postupně vytváří paralelní větve, které pracují s redukovaným rozlišením. Tyto větve si navzájem vyměňují informace, jak je možné vidět v obrázku 3.8. Princip fungování této sítě je podrobněji popsán v práci [20].



Obrázek 3.8: Architektura sítě HRNet, ukazující postupné větvení sítě se snižujícím se rozlišením. Převzato z [20].

[2] https://github.com/leoxiaobin/deep-high-resolution-net.pytorch

Kapitola 4

Neuronové sítě pro zpracování obrazu

Neuronové sítě [3, 24, 9] jsou stále populárnější řešení problémů, především v počítačovém vidění, kde konvoluční neuronové sítě už posledních několik let stabilně překonávají výsledky klasických algoritmů. S narůstající hardwarovou i softwarovou podporou pro strojové učení můžeme očekávat, že jejich popularita bude dále narůstat. V této kapitole jsou stručně popsány neuronové sítě, princip jejich fungování a jejich využití k zarovnání obrázků.

Neuronové sítě jsou jedním z výpočetních modelů strojového učení. Slouží především k řešení problémů, které jsou pro lidi jednoduché a intuitivní na vykonávání, ale obtížné na formální popis a definici, protože problém řeší na základě znalostí získaných z datové sady. Jedná se tedy o učení s učitelem. Díky tomu je možné, že odhalí vzory v datech, o kterých ani nevíme, že existují. Jsou inspirovány lidským mozkem, který je složen přibližně z 10^{11} základních jednotek zvaných neurony, které si mezi sebou posílají signály.

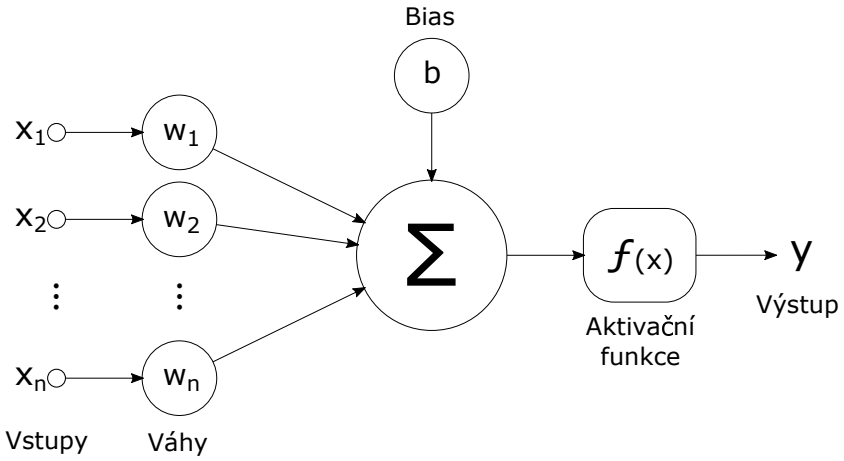
4.1 Biologický neuron

Biologický neuron se skládá z těla, které se také nazývá *soma*, do kterého vede velké množství výběžků tzv. *dendritů*, které slouží jako vstup neuronu. Poslední část neuronu je *axon*, který slouží jako výstup daného neuronu, který se napojuje na dendrity ostatních neuronů. Propojení dendritu a axonu se nazývá synapse. Neuron funguje jako spínač, který sepne, pokud má na vstupu dostatečně velké impulzy. Při sepnutí vyšle velice krátký impulz na axon, kterým je schopný aktivovat další neurony. Pomocí obrovského množství takto propojených neuronů je mozek schopný vykonávat komplexní funkce.

4.2 Umělý neuron

Na základě biologického neuronu byl navrhnut zjednodušený umělý neuron, který zastává podobnou funkci. Funkce implementovaná umělým neuronem je popsána v rovnici (4.1). Na vstupu dostane vektor signálů, které vynásobí s příslušnými vahami. Tyto vážené vstupy sečte a přičte hodnotu b (*bias*). Na výsledek bázové funkce je aplikovaná aktivační funkce f a výsledná hodnota y se šíří dále. Schéma neuronu je možné vidět na obrázku 4.1.

$$y = f(\sum_{i=1}^{n} w_i x_i + b) \tag{4.1}$$



Obrázek 4.1: Schéma umělého neuronu.

4.2.1 Aktivační funkce

Jelikož je bázová funkce umělého neuronu lineární, musí být aktivační funkce f využitá v rovnici (4.1) nelineární, aby zajistila schopnost neuronové sítě aproximovat nelineární funkci. Aktivační funkce definuje výstup z neuronu. Nejčastěji využívanou aktivační funkcí je *ReLU* (4.2), především pro jednoduchost jejího výpočtu. Při využití ReLU však může nastat k „umírání neuronů“ s negativním výstupem, proto byla vytvořena *Leaky ReLU* (4.3), která propouští malý zlomek negativního výstupu neuronu. Funkce *Sigmoid* má výstup omezený v rozmezí 0 až 1 viz (4.4), proto se využívá při práci s pravděpodobností. Funkce sigmoid se využívá při klasifikaci, ve které daný vstup může patřit do více tříd. Na rozdíl od sigmoidy aktivační funkce *Softmax*, popsaná v rovnici (4.6), se využívá při klasifikaci tříd, které jsou vzájemně výlučné. Výlučnosti tříd se dosáhne tím že je celkový součet pravděpodobností roven 1 a tedy pokud neuronová síť zvedá pravděpodobnost jedné třídy, snižuje tím pravděpodobnosti ostatních. *Tanh* je podobná funkci sigmoid, ale její výstup je v rozmezí -1 a 1 (4.5). Průběhy jednotlivých funkcí lze vidět v obrázku 4.2.

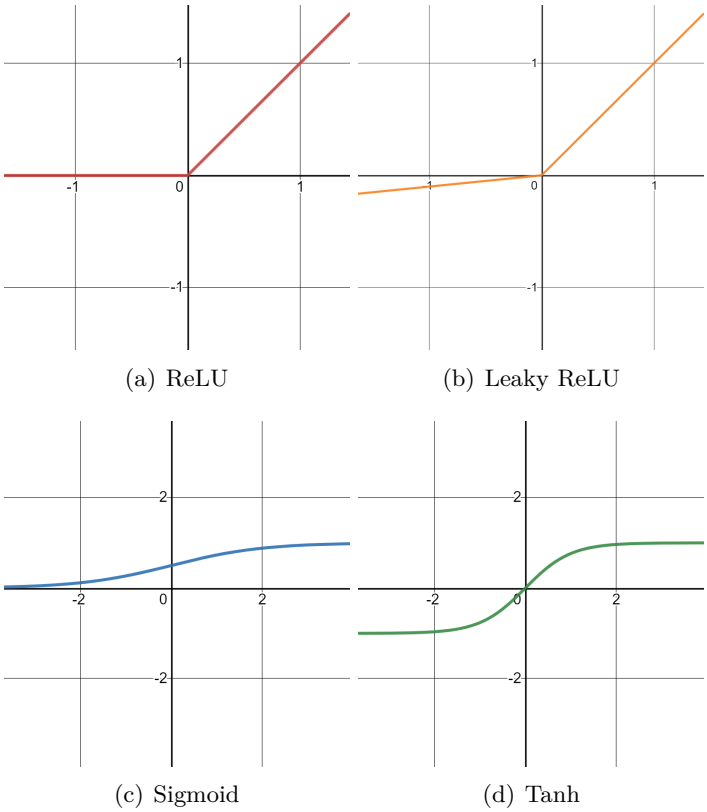
$$ReLU(x) = \begin{cases} x & prox \geq 0 \\ 0 & jinak \end{cases} \tag{4.2}$$

$$LeakyReLU(x) = \begin{cases} x & prox \geq 0 \\ 0,01x & jinak \end{cases} \tag{4.3}$$

$$Sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{4.4}$$

$$Tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{4.5}$$

$$SoftMax(x) = \frac{\exp(x)}{\sum_j \exp(x_j)} \tag{4.6}$$



(a) ReLU (b) Leaky ReLU

(c) Sigmoid (d) Tanh

Obrázek 4.2: Ukázka průběhů aktivačních funkcí.

4.3 Konvoluční neuronové sítě

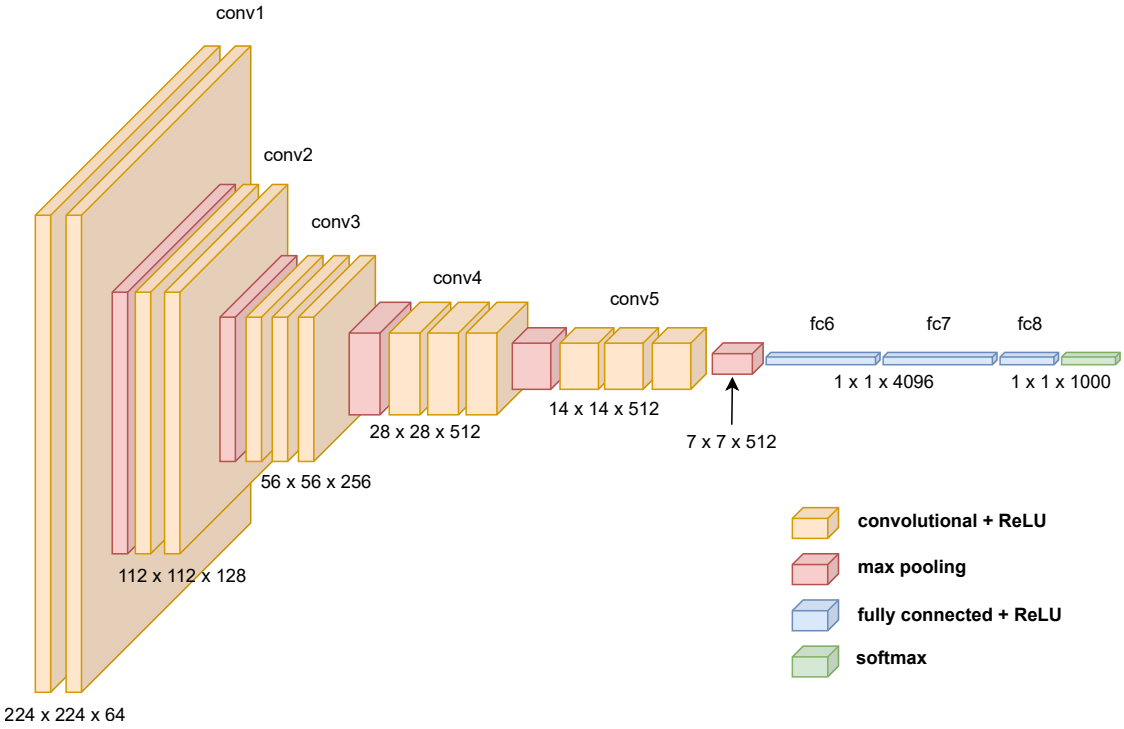
Konvoluční neuronové sítě jsou speciální kategorií neuronových sítí, které se vyznačují tím, že využívají konvoluci, která je nejčastěji implementována jako samostatná vrstva. Dále využívají pooling vrstvy a plně propojené vrstvy. Typická architektura konvoluční sítě je tvořena opakováním konvoluční vrstev, následovaných pooling vrstvami a je ukončena jednou či více plně propojenými vrstvami. Tato architektura se označuje jako pyramidová a je její ukázku je možné vidět na obrázku 4.3. Další typickou architekturou je architektura přesýpacích hodin, která se využívá, pokud chceme mít na výstupu obraz. Tato architektura se využívá například k segmentaci obrazu nebo získání teplotních map.

Základní myšlenka, využitá v konvolučních sítích, byla popsána v práci *Neocognitron* [8] už v roce 1980, ve které se autoři snažili navrhnout systém pro rozpoznávání vzorů (*patterns*) nezávisle na posunu. Tato práce vycházela z výzkumu částí mozkové kůry koček, odpovědné za zrakové funkce [11].

4.3.1 Vrstvy neuronových sítí

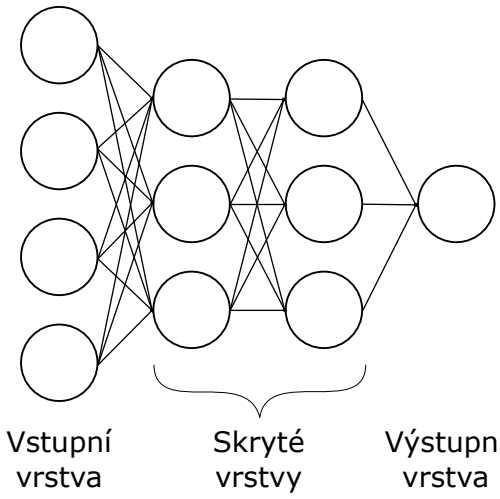
Při vývoji neuronových sítí se nepracuje se samostatnými neurony, nýbrž s vrstvami, které obsahují velké množství neuronů. Obvykle se dají tyto vrstvy dělit na vstupní, skryté a výstupní, viz obrázek 4.4. Propojováním skrytých vrstev určujeme architekturu sítě, a to jakým způsobem se v ní budou šířit informace.

[1] https://github.com/kennethleungty/Neural-Network-Architecture-Diagrams



Obrázek 4.3: Ukázka konvoluční sítě s pyramidovou architekturou. Jedná se o architekturu s názvem VGG16. Převzato z [1].

Podle toho můžeme sítě dělit na dopředné (*feedforward*), ve kterých se data předávají postupně od vstupu, přes skryté vrstvy až na výstup, a rekurentní (*recurrent*), ve kterých se vytváří smyčky a data mohou procházet skrz určité vrstvy i několikrát.

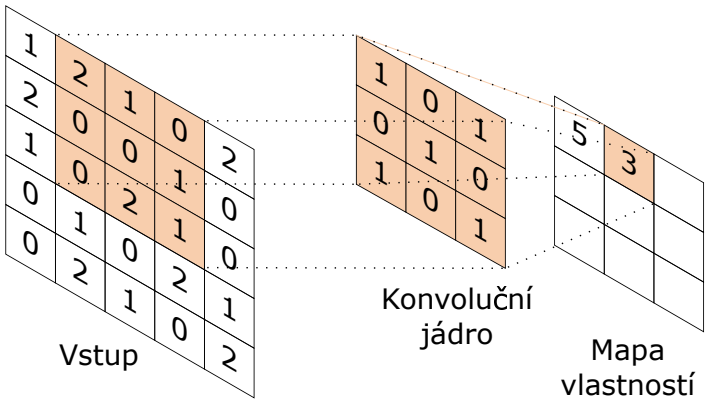


Obrázek 4.4: Ukázka abstraktní neuronové sítě se 4 vrstvami.

Konvoluční vrstva

Konvoluční vrstva je hlavní stavební blok konvolučních neuronových sítí, sloužící k extrakci vlastností jako jsou hrany, rohy a objekty. V konvoluční vrstvě se aplikuje jedna či více konvolucí, na vstupní data, které mohou mít více rozměrů např 2D při zpracování obrazu. Parametry konvoluční vrstvy, které je možné trénovat, jsou pouze jádra konvolucí, které bývají v rozměrech 3x3 až 5x5.

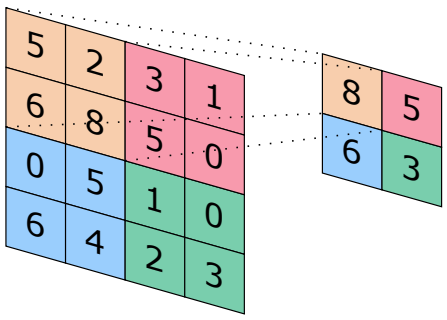
Konvoluce je matematická funkce, kterou si lze představit jako posouvání konvolučního jádra (*kernel*) po vstupních datech viz obrázek 4.5, překrývající se hodnoty jsou vynásobeny a výsledky sečtou. Výhodou konvoluce je že dokáže odhalovat vlastnosti nezávisle na posunu.



Obrázek 4.5: Ukázka 2D konvoluce.

Pooling vrstva

Tato vrstva provede pod-vzorkování (*subsampling*) vstupu, a tedy zmenší rozlišení mapy vlastností, které jsou výstupem konvoluční vrstvy. Postupnou redukcí rozlišení, pomocí pooling vrstev, umožňujeme následujícím konvolučním vrstvám extrahovat vlastnosti pokrývající vetší plochu původního vstupu. Samotná pooling vrstva nemá parametry, které by se daly trénovat. Nejpoužívanější variantou je *max pooling*, která podle velikosti filtru zredukuje plochu na jednu hodnotu, která je maximální hodnotou z dané plochy. Další variantou je *average pooling*, ve kterém je výsledek zredukované plochy roven průměru hodnot.



Obrázek 4.6: Ukázka pooling vrstvy, která pod-vzorkuje obraz na poloviční rozlišení. Jedná se o max pooling.

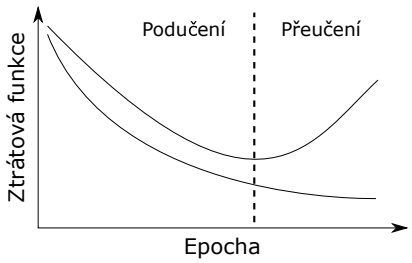
Plně propojená vrstva

Plně propojená vrstva (*Fully connected layer*) má na vstupu vektor hodnot, jehož každá hodnota je připojena na vstup všech neuronů v dané vrstvě. V rámci konvolučních neuronových sítí následují po extrakci vlastností pomocí konvolučních a max pooling vrstev a z daných vlastností počítají samotný výsledek sítě.

4.4 Trénování neuronových sítí

Trénování obvykle probíhá jako iterativní proces, ve kterém jsou síti předána vstupní data a její výsledky jsou porovnány s cílovými hodnotami (*Ground truths*) tím, že se z nich vypočítá chybová funkce (*Loss function*), která indikuje, jak moc se liší výstup neuronové sítě od očekávaného výstupu. Dále se na základě vypočítané chyby upraví váhy neuronů pomocí optimalizačních metod, založených na gradientním sestupu (*Gradient descent*) tak, aby co nejvíce zmenšily, chybu pro daná data. Z toho plyne, že aktivační funkce musí být diferencovatelné, jinak by možné z nich vypočítat gradient. Tyto metody využívají zpětného šíření chyby (*Backpropagation*), které začíná na výstupní vrstvě a končí na vstupní. Celý tento proces je prováděn nad datovou sadou několikrát, kde se jeden průchod nazývá epocha.

Cílem učení však není pouze snížení chyby na trénovacích datech, ale také na datech, na kterých nebyla síť trénovaná, jinak by nebyla využitelná v praxi. Tento cíl se nazývá generalizace a během vývoje neuronové sítě se vykonává mnoho kroků, jejichž cílem je zařídit, aby byla generalizace co největší. Nejzákladnějším krokem je rozdělení datové sady na trénovací a testovací sady, kdy je k učení neuronové sítě využita pouze trénovací sada, ale sledují se průběhy chybové funkce jak trénovací, tak testovací sady. Z průběhu můžeme pozorovat, jak dobře neuronová síť generalizuje daný problém, a můžeme odhalit přeučení (*overfitting*) či podučení (*underfitting*). Přeučení nastává, když začne růst rozdíl mezi trénovací a testovací chybou. Podučení nastává, když model dostatečně nesnížil ani trénovací chybu. Ukázku průběhu ztrátových chyb lze vidět na obrázku 4.7. Oba případy často souvisí s kapacitou modelu a počtem provedených epoch, kde nedostatečný počet epoch neumožní modelu snížit trénovací chybu a příliš velká kapacita a počet epoch umožní modelu si zapamatovat vlastnosti trénovacích dat, ale zamezí generalizaci problému. Možným řešením přeučení je tzv. *Early stopping*, při kterém se zastaví trénování, pokud začne růst hodnota ztrátové funkce nad testovacími daty.



Obrázek 4.7: Vizualizace průběhu trénování při kterém došlo k přeučení. Zatímco hodnoty trénovací ztrátové funkce stále klesají, testovací ztrátová funkce začíná růst.

Dalším řešením přeučení je *droupout vrstva*, která při každé iteraci učení se zadanou pravděpodobností vyřadí náhodné neurony a jejich propojení, což znemožní síti využít své kapacity k zapamatování trénovacích příkladů [17].

4.4.1 Chybové funkce

Jak bylo výše zmíněno chybová funkce se využívá k výpočtu odlišnosti odhadu od cílové hodnoty. Je tedy nutné aby její výstup byla skalární hodnota, která je rovna 0 při dokonalé schodě a roste spolu se vzdáleností odhadu $\hat{y}$ od cílové hodnoty y_i. Mezi nejčastěji využívané chybové funkce patří:

Střední kvadratická chyba

Zkratkou MSE z anglického *mean square error*. Tato chyba je využívaná v regresních modelech a je popsána v rovnici (4.7), kde jsou rozdíly odhadů a cílových hodnot umocněny a zprůměrovány. Umocnění se využívá pro zamezení záporné chyby a zároveň pro zdůraznění větších odchylek.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{4.7}$$

Střední absolutní chyba

Zkratkou MAE z anglického *mean absolute error*. Stejně jako MSE je tato funkce využívána při regresi. Výpočet je stejný jako u MSE, pouze místo druhé mocniny je vypočítána absolutní hodnota, jak je možné vidět v rovnici (4.8). Není tedy kladen důraz na vetší odchylky.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{4.8}$$

Křížová entropie

Křížová entropie (*cross entropy*) je funkce využívána při trénování klasifikačních modelů. Její definici je možné vidět v rovnici (4.9). Pokud trénovací data obsahují vždy jen jednu správnou třídu a hodnoty y_i ostatních tříd jsou rovné 0, je možné nepočítat sumu pro všechny třídy ale pouze pro cílovou třídu s očekávanou hodnotou 1.

$$CE = -\sum_{i=1}^{n} y_i \cdot \log \hat{y}_i \tag{4.9}$$

4.5 Existující řešení zarovnávání obrazu pomocí neuronových sítí

V rámci řešení byl proveden průzkum existujících řešení, které využívají neuronové sítě k odhadu zarovnání. V posledních letech bývá preferován přístup k detekci klíčových bodů, využívající neuronové sítě, namísto klasických postupů využívající algoritmu SIFT či jiných algoritmů. Detekované klíčové body se následně filtrují a počítá se z nich transformace, například pomocí algoritmu RANSAC, který je popsán v sekci 2.3. Jelikož filtrace nemusí být dokonalá a spolu s výpočtem transformace může být tento postup stále velmi časově náročný, začal vývoj neuronových sítí implementujících celý tento proces, jejichž výsledkem je přímo transformace zarovnání.

Přestože se v nalezených pracích podařilo vytvořit systémy využívající neuronové sítě pro výpočet zarovnání, nebyla nalezena žádná práce, která by zarovnání určovala na základě

lidské pózy. Nicméně dohledané práce obsahovaly velmi užitečné techniky, které byly dále využity.

4.5.1 Deep Image Homography Estimation

Jedním z existujících řešení je práce *Deep Image Homography Estimation* [6], v rámci které byla vytvořena konvoluční neuronová síť, schopná odhadnout transformační matici zarovnání dvou obrázků, čímž bylo dokázáno, že neuronové sítě jsou schopné řešit úkoly tohoto typu.

Jelikož neexistují datové sady pro trénovaní neuronové sítě, která počítá zarovnání obrázků, bylo nutné generovat trénovací data pomocí aplikace náhodně vytvořených transformací na obrázky z již existujících datových sad, využívaných například ke klasifikaci nebo detekci lidské pózy. Trénování proběhlo nad obrázky s rozlišením 128×128, kde výstup neuronové sítě nebyla transformační matice, nýbrž čtveřice bodů kterou je možné 1:1 mapovat na matici. Tento výstup byl zvolen z důvodu snazšího výpočtu chybové funkce a celkového hodnocení různých postupů na odhad zarovnání.

V rámci tohoto výzkumu byla na dané datové sadě neuronová síť porovnána s dvěma standardními algoritmy. Tyto algoritmy využívají detektor klíčových bodů *ORB*, ze kterých je počítána transformace, přičemž jeden při výpočtu využil algoritmu RANSAC pro filtraci outlierů.

Další dva nově vytvořené přístupy využívají konvoluční neuronovou síť. Přístupy se lišily v architektuře sítě pouze výstupní vrstvou, kde v jedné variantě byl výstup 8 hodnot z plně propojené vrstvy, jedná se tedy o regresi, zatímco druhá varianta je implementována jako klasifikace, kde jsou obory hodnot původní regrese, segmentovány na 21 tříd. Výstupní vrstva obsahuje 8x21 hodnot, na které je aplikována aktivační funkce softmax.

4.5.2 DAN Deep Alignment Network

Výsledek práce [13] je systém pro zarovnání obličeje, využívající rekurentní neuronovou síť s více etapami, kde v každé etapě aplikují zarovnání, čímž postupně zvětšují přesnost. Na rozdíl od *Deep Image Homography Estimation*, kde je jeden obrázek zarovnávaný k druhému, je v této síti na vstupu pouze jeden obrázek, který se zarovnává na střed a se správnou rotací.

Kapitola 5

Návrh řešení

V této kapitole jsou popsány návrhy systémů pro zarovnání obrázků. Cílem je navrhnout dva systémy, vhodnou datovou sadu pro tvorbu daných systému a jejich testování.

První systém bude využívat klasické metody pro zarovnávání obrázků a bude zaměřen na přesnost zarovnaní, nehledě na výpočetní náročnost.

Druhý systém bude vytvořen za použití konvoluční neuronové sítě a bude trénován na datové sadě vytvořené pomocí prvního systému. Díky využití konvoluční neuronové sítě by se měla snížit hardwarová náročnost a tím pádem i zvýšit rychlost.

5.1 Sada obrázků sportovních pozic

Prvním krokem je vytvoření sady obrázků se sportovními pozicemi, které budou k sobě zarovnávány. Obrázky musí být roztříděné na základě sportovní pozice a úhlu, ze kterého byly pořízeny. Třídění na základě úhlu pořízení je nutné, protože s narůstajícím úhlem si pozice zaujaté na fotografiích přestávají být podobné z hlediska 2D detekce lidské pózy.

Po konzultaci s vedoucím jsem se rozhodl vytvářet a testovat systémy na obrázcích jógy, především proto, že v józe jsou pozice dostatečně různorodé a je možné předpokládat, že výsledný systém by měl být schopný zarovnávat i obrázky pozic z jiných sportů. Dalším důvodem je vetší množství unikátních pozic v rámci krátkého cvičení, na rozdíl od ostatních sportů.

5.2 Systém využívající klasické postupy pro zarovnávání

Tento systém bude zaměřen na přesnost zarovnávání a primárním účelem tohoto systému bude získat záznamy o zarovnání, které bude možné využít při trénování konvoluční neuronové sítě. Systém bude využívat mírně upravený postup pro zarovnání obrázků, který je tvořen z následujících kroků:

1. Detekce klíčových bodů
2. Filtrace bodů a výpočet transformační matice
3. Aplikace zarovnání a uložení výsledků

5.2.1 Detekce klíčových bodů

Jelikož je cílem zarovnávat obrázky na základě lidské pózy, budou jako klíčové body využity jednotlivé části těla. Detekce částí lidského těla bude provedena pomocí vhodně vybrané knihovny, která bude spuštěna nad vytvořenými sadami obrázků. Detekované pozice budou uloženy do složky, která je příslušná pro danou pozici, a jména těchto souborů budou odpovídat obrázkům až na jejich příponu. Na základě této schody bude možné propojit dvojice, složené z obrázku a detekovaných kloubů

Detekci klíčových bodů jsem se rozhodl oddělit od zbytku procesu, protože při vývoji nedává smysl opakovaně detekovat lidské pózy ze stejných obrázků. Výsledný systém by mohl detekovat klíčové body automaticky při každém zarovnání, ale jelikož bude tento systém vytvořen spíše pro tvorbu datové sady a porovnání s neuronovou sítí, rozhodl jsem se tuto možnost odložit jako možné rozšíření.

5.2.2 Filtrace bodů a výpočet transformační matice

Po detekci klíčových bodů je možné začít zarovnávat dva obrázky obsahující stejnou pózu. Prvně budou detekované body načteny a zpracovány. Po načtení klíčových bodů, detekovaných na daných obrázcích, bude nutné vyfiltrovat chybně detekované klíčové body. K filtraci bodů a odhadu matice bude využit algoritmus RANSAC, který bude přizpůsoben pro tento úkol. Odhadovaný model bude transformační matice, vypočítána z náhodně vybraných párů klíčových bodů. Páry klíčových bodů budou náležet do modelu, pokud jejich eukleidovská vzdálenost po aplikaci transformační matice na zarovnávaný bod bude menší než předem určená tolerance. Hranici úspěšného modelu pravděpodobně nebudu implementovat, jelikož i z minima bodů je stále možné vypočítat správnou transformaci, pokud budou všechny detekovány správně. Model bude hodnocen klasicky podle počtu párů, které pod něj spadají.

5.2.3 Aplikace zarovnání a uložení výsledků

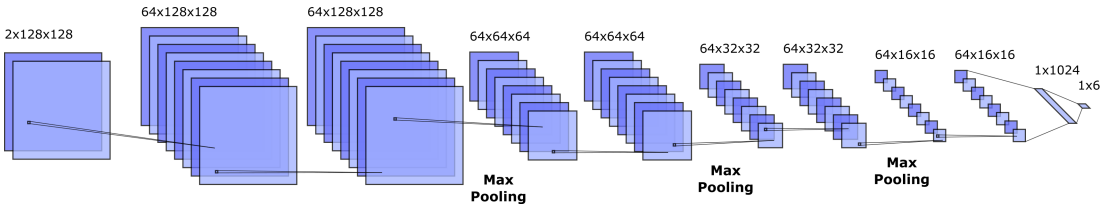
K aplikaci zarovnání bude využita vhodná knihovna pro úpravu obrázků. Po zarovnání bude z cílového i zarovnaného obrázku vytvořen gif, pro snadné porovnání. Tento gif bude uložen do složky *aligned* a bude pojmenován podle obou obrázků. Dále bude vytvořen záznam o zarovnání, který bude možné dále využít, pro trénování neuronové sítě.

5.3 Systém využívající konvoluční neuronové sítě

Druhý systém bude využívat konvoluční neuronovou síť pro odhad transformace. Základ architektury bude převzat z práce *Deep Image Homography Estimation*, popsané v sekci 4.5.1. Na vstupu budou dva čtvercové šedotónové obrázky, jejichž rozlišení bude měněno během experimentů. V rámci implementace bude vyzkoušeno více formátů výstupu, mezi které patří transformační matice, jednotlivé prvky transformací a rohové body, ze kterých je možné vypočítat transformační matici. Tyto formáty bude neuronová síť počítat pomocí regrese nebo klasifikace segmentů oboru hodnot původní regrese.

Pro trénování budou využity záznamy o zarovnání vytvořené prvním systémem, které budou náhodně rozděleny do trénovací a testovací sady. Při trénování bude sledován průběh ztrátové funkce na obou sadách.

Po natrénování sítě proběhne poslední testování, jehož výsledky budou uloženy do textových souborů pro porovnání a zároveň budou odhadnuté transformace aplikovány na



Obrázek 5.1: Model využitý k implementaci systému.

obrázky, které budou uloženy pro demonstraci výsledků systému. Natrénovaná síť bude uložena na disk, pro následné použití.

Kapitola 6

Implementace

V této kapitole kapitole je popsána postupná implementace obou systémů, které jsou popsány v návrhu, tvorbu sady obrázků, která je využívána oběma systémy. Při implementaci bylo experimentováno s různými postupy, které zde budou popsány. Také zde stručně popíši využité technologie a důvody proč byly zvoleny.

Python

K implementaci obou systémů byl zvolen jazyk python[1] verze 3.9.8. Python je interpretovaný, objektově orientovaný a vysokoúrovňový jazyk, který je vhodný pro manipulaci se složitými objekty dat. Využívá dynamického typovaní a jeho paměť je spravována pomocí *garbage collectoru*. Důraz je kladen na čitelnost kódu a produktivitu při programování, proto je vhodný k prototypování programů. Další výhodou je řada knihoven, které implementují řešení řady problémů z různých obrů, např. počítačové vidění, analýza dat a tvorba neuronových sítí.

OpenCV2

OpenCV2[2] je volně dostupná, multiplatformní knihovna zaměřená na zpracování obrazu a úkoly počítačového vidění. K řadě jejích výpočtu je využita akcelerace pomocí grafické karty.

Pytorch

Pytorch[3] [19] je volně dostupná knihovna pro strojové učení, která díky objektově orientovanému přístupu umožní snadné prototypování a vývin neuronových sítí. Podporuje také akceleraci výpočtů pomocí grafické karty. K této akceleraci je nutné mít nainstalovanou verzi této knihovny, která může běžet na grafické kartě. Zároveň je potřeba mít grafickou kartu podporující technologii CUDA. Tato knihovna zároveň implementuje známe ztrátové funkce a třídy, které usnadní tvorbu částí učící smyčky jako třeba načítání dat či optimalizátoru. Další výhodou této knihovny je automatické počítání gradientů nutných pro optimalizaci.

[1] https://www.python.org/
[2] https://opencv.org/
[3] https://pytorch.org/

6.1 Tvorba sady obrázků

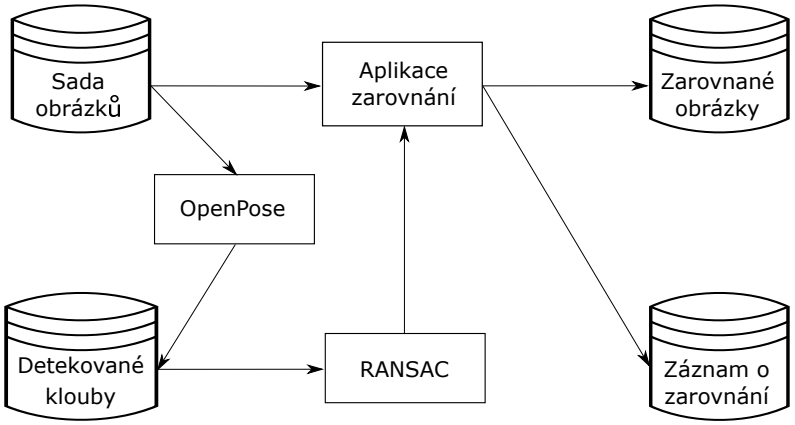
Obrázky pozic byly získávány manuálně z videí obsahující cvičení jógy, pomocí funkce pro zachycení aktuálního snímku. Část z využitých videí jsem získal z YouTube, přesněji z kanálu *Yoga With Adriene*[4], ale většina mi byla poskytnuta vedoucím práce. Výsledná sada fotek tedy obsahuje pouze 2 osoby, což není dostatečný počet pro vytvoření systémů použitelných v praxi, ale pro testování a demonstraci konceptů by měla tato sada stačit. Navíc co sada ztrácí v počtu osob získává ve variabilitě pozadí a oblečení cvičících. Videa poskytnutá vedoucím jsou natáčena pouze v jedné místnosti, mění se ovšem pozice i úhel natáčení. Výsledné datové sady obsahují 1 506 obrázků rozdělených do 53 pozic.

Další zvažovanou možností bylo využít veřejně dostupné datové sady[5][6] pro klasifikaci jógových pozic, ukázalo se ovšem, že tyto sady obsahují různé varianty dané pózy, jsou foceny z příliš odlišných úhlů a obecně nebyly vhodné pro řešení této práce.

Výsledné sady obrázků jsou rozdělené do dvou složek *DataYouTube* a *DataHerout* na základě zdroje videí, ze kterých byly získány. V těchto složkách jsou dále děleny podle pozic a úhlů pořízení. Mimo obrázků jsou zde uložena příslušná data, která budou popsána dále.

6.2 Implementace systému využívajícího RANSAC

Jak již bylo popsáno v návrhu, tento systém bude využívat klasický postup pro zarovnání obrázků. Celý systém lze vidět na schématu 6.1 a popis implementace jednotlivých kroků bude níže.



Obrázek 6.1: Schéma systému pro výpočet transformace na základě sportovní pozice.

6.2.1 Detekování částí lidských těl

K detekci částí lidských jsem zvolil knihovnu OpenPose, především na základě doporučení vedoucího této práce, ale i pro její jednoduché použití a dobrou dokumentaci. Kvůli této volbě se bude pravděpodobně na celý výsledek této práce vztahovat nekomerční licence knihovny OpenPose. Samotnou knihovnu jsem nepožíval, jelikož autoři této knihovny vytvořily demonstrační nástroj, který implementuje vše co budu v rámci této práce potřebovat.

[4] https://www.youtube.com/channel/UCFKE7WVJfvaHW5q283SxchA
[5] https://arxiv.org/abs/2004.10362
[6] https://www.kaggle.com/datasets/niharika41298/yoga-poses-dataset?resource=download

Jelikož OpenPose využívá rozsáhlou konvoluční síť, jejíž výpočty jsou prováděny na grafické kartě, vyskytl se problém s nedostatkem grafické paměti. Tento problém bylo snadné vyřešit, jelikož OpenPose umožňuje změnit rozlišení vstupního obrázku a tím omezit paměťovou náročnost. Postupným snižování vstupního rozlišení byla získána maximálnímu stabilní hodnota rozlišení -1x208, kde -1 je flexibilní hodnota.

Vydaný nástroj se spouští pomocí příkazové řádky, ve které se mu předají argumenty obsahující cesty a nastavení. Je tedy spuštěn například příkazem:

```
.\bin\OpenPoseDemo.exe -net_resolution="-1x208" -image_dir="..\Images"
-write_json="..\Jsons" -write_images="..\Processed"
```

kde jednotlivé vlajky nastavují:

-net_resolution	rozlišení sítě
-image_dir	cesta ke složce obsahující obrázky ke zpracování
-write_json	složka do které se uloží JSON soubory obsahující detekované body
-write_images	složka do které se uloží obrázky s detekovanou kostrou

Výsledky detekce pózy je možné sledovat na obrázcích s vloženou kostrou viz 6.3.

(a) (b) (c)

Obrázek 6.2: Ukázka obrázků s vizualizací detekované pózy.

OpenPose generuje JSON soubor pro každou fotografii. Jelikož umožňuje detekci více osob, generované JSON soubory obsahují pole „people“, ze kterého však bude vždy využita pouze první osoba. Objekt reprezentující osobu obsahuje další pole s názvem „pose_keypoints_2d“. Jedná se o pole obsahující 75 hodnot s plovoucí desetinou čárkou. Tyto hodnoty reprezentují po sobě jdoucí trojice ve formátu [souřadnice x, souřadnice y, jistota], které představují jednotlivé klouby. Ve výsledku pole obsahuje 25 detekovaných kloubů, jejichž pořadí je popsáno v dokumentaci OpenPose[7]. Pořadí je také možné vidět na obrázku modelu *Body25* 3.7. Klouby, které se nepodařilo detekovat jsou reprezentovány pomocí 3 nul, aby bylo zachováno pořadí kloubů.

6.2.2 Načítání detekovaných částí těl

Detekované pózy uložené ve formátu JSON je nutné načíst a zpracovat pro snadnější použití. Po načtení je ze souboru získána první osoba. Pokud nebyla detekována je soubor smazán. Dále je z jednorozměrného pole vytvořeno pole dvourozměrné, ve kterém první sloupec

[7] https://cmu-perceptual-computing-lab.github.io/openpose/web/html/doc/md_doc_02_output.html#output-format

obsahuje souřadnice osy x, druhý sloupec obsahuje souřadnice osy y a jistota je vyfiltrována, jelikož nebude k ničemu využita, stejně jako posledních 10 bodů, které reprezentují obličej a chodidla. Zároveň je přidán třetí sloupec který vždy obsahuje hodnotu 1, která usnadní násobení matic. Takto zpracované pole obsahuje 15 bodů ve formátu [x, y, 1] a je uloženo do asociativního pole spolu s cestou k obrázku ze kterého byla pozice detekována.

6.2.3 Zrcadlení polohy

Před výpočtem transformace z klíčových bodů je potřeba počítat s možností, že zarovnávané obrázky obsahují stejnou pózu, která je ovšem zaujatá zrcadlově a je tedy potřeba zarovnávat pravé končetiny s levými a naopak.

(a) (b)

Obrázek 6.3: Ukázka obrázků se zrcadlenou pózou.

Z tohoto důvodu se v této části program větví do dvou částí. V jedné je počítáno zarovnání z původních bodů a v druhé je změněno pořadí bodů, tak aby byly zrcadleny. Změnu pořadí bodů je v jazyce python možné implementovat pomocí indexace seznamem.

```
pointsMirrored = Points[[0, 1, 5, 6, 7, 2, 3, 4, 8, 12, 13, 14, 9, 10, 11]]
```

Po výpočtu obou transformačních matic jsou porovnány jejich hodnocení a je vrácena matice s lepším ohodnocením. V některých případech je možné, že tento postup špatně určí, kterou matici vrátit, ale to bývá nejčastěji způsobeno různě zaujatou pózou na zarovnávaných obrázcích.

6.2.4 RANSAC

I přes to že knihovna OpenCV2 již implementuje variantu algoritmu RANSAC, která počítá transformační matici zarovnání z párů klíčových bodů, rozhodl jsem se implementovat vlastní verzi, která bude lépe přizpůsobená k tomuto problému. RANSAC je implementován funkcí:

```
CalculateAffineTansformation(srcPoints, dstPoints, eps, allowYFlip)
```

Tato funkce má na vstupu dvě pole obsahující 15 klíčových bodů, jejichž formát je popsán výše, toleranci vzdálenosti *eps* a příznak, indikující, zda byly body zrcadleny, jehož využití bude popsáno níže. Návratové hodnoty funkce jsou výsledná transformační matice

a asociativní pole obsahující dodatečné hodnoty. Ve finální verzi obsahuje pouze jednu hodnotu *InliersCount*, ale v rámci implementace obsahovala další hodnoty, podle kterých byl tento systém laděn. Příkladem takové hodnoty je informace o tom, v kolikátém cyklu byl nejlepší model získán.

V návrhu tohoto systému nebylo specifikováno, jakou třída transformační matice bude počítána, protože se předpokládalo, že stejně jako v klasickém zarovnání, bude počítána matice projektivní. Ukázalo se ovšem, že matice s takto velkým stupněm volnosti dokáže zarovnat i špatně detekované body, čímž značně deformuje obrázek. Z toho důvodu jsem se po konzultaci s vedoucím rozhodl využívat pouze afinní transformaci, která omezí vypočítanou transformaci tak, aby nedocházelo k poškození obrázku, což znehodnotilo část implementované práce.

6.2.5 Příprava dat

Před samotným výpočtem pomocí RANSACu jsou pole bodů vertikálně spojeny do pole o rozměrech 6x15, kde jeden řádek obsahuje pár korespondujících kloubů z obou obrázků. Dále jsou filtrovány páry, ve kterých alespoň jeden bod má nulové souřadnice, protože OpenPose nebyl schopný detekovat daný kloub. Pokud výsledné pole neobsahuje dost párů pro výpočet zarovnání, je vrácena jednotková matice s počtem Inlierů -1.

6.2.6 Hlavní smyčka

Dále začíná hlavní smyčka algoritmu RANSAC. Pro výpočet počtu iterací byla využita rovnice, která je popsána v teoretické části (2.9). Hodnoty byly voleny tak, abych cílil na co nejlepší přesnost v proveditelném čase. Cílenou pravděpodobnost byla zvolena 99,9 %. Poměr inlierů a outlierů, byl problematický na odhad, jelikož nedetekované body jsou filtrovány, a celkový počet záznamů se tedy pro každý pár obrázků mění. Nakonec byl raději zvolen poměr výrazně horší, než jaký je ve skutečnosti a to 15:9. Poslední hodnotou v dané funkci je počet náhodně vybraných párů, který je pro výpočet afinní transformace 3. Výsledná hodnota je tedy 27.62 iterací, která se tedy zaokrouhlí na 28.

6.2.7 Odhad modelu

V prvním kroku jsou vybrány 3 náhodné páry bodů, ze kterých je vypočítána afinní transformace. K výpočtu byla původně využitá metoda, která počítá transformační matici z dvojic klíčových bodů pomocí singulárního rozkladu[8] (zkratka SVD z anglického *Singular value decomposition*), ale pro vetší počty byla využita funkce knihovny OpenCV2 *getAffineTransform*, která implementuje tento výpočet optimálněji a výsledná transformace je normalizovaná.

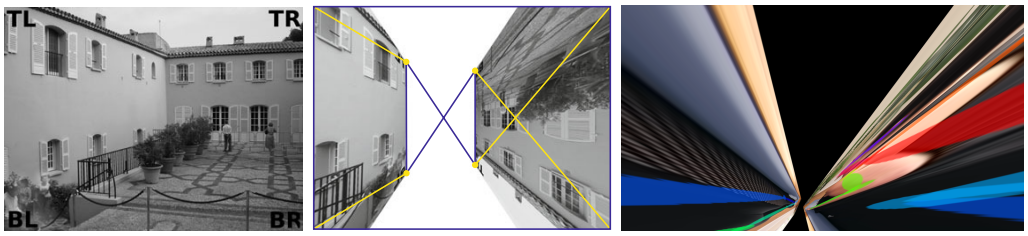
Vypočítaná matice je následně kontrolována metodami:

1. IsRigidTransformation
2. IsTransformExtreme

[8] Výpočet v metodě byl převzán z https://math.stackexchange.com/questions/3509039/calculate-homography-with-and-without-svd

IsRigidTransformation

Tato metoda kontroluje, zda nebyla vypočítána degenerativní transformace, která zarovnaný obraz ohne „za kameru“, čímž ho značně deformuje, jelikož bude rozdělen do dvou částí viz 6.4.



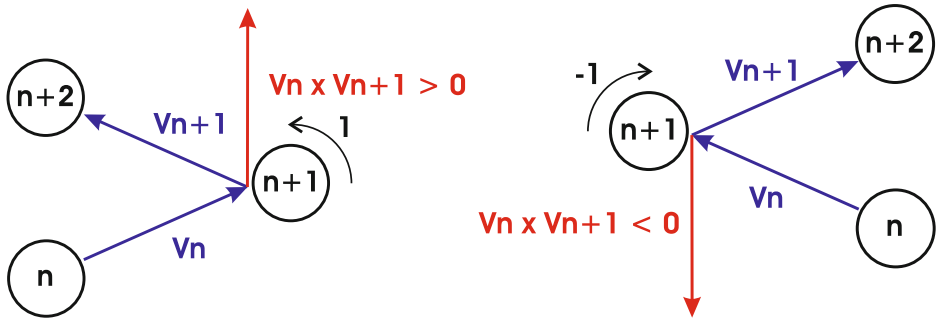
(a) Originální obrázek budovy převzato z [15].

(b) Deformovaný obrázek budovy převzato z [15].

(c) Deformovaný obrázek cvičení jógy

Obrázek 6.4: Ukázka obrázků po aplikaci degenerativní transformace.

Tato metoda využívá postup popsaný v knize *Advanced Concepts for Intelligent Vision Systems* [15]. Tento postup je založen faktu, že znaménko vektorového součinu udává směr otočení druhého vektoru k prvnímu, jak je možné vidět na obrázku 6.5.



(a) Vektorový součin se směrem otočení doleva. Převzato z [15].

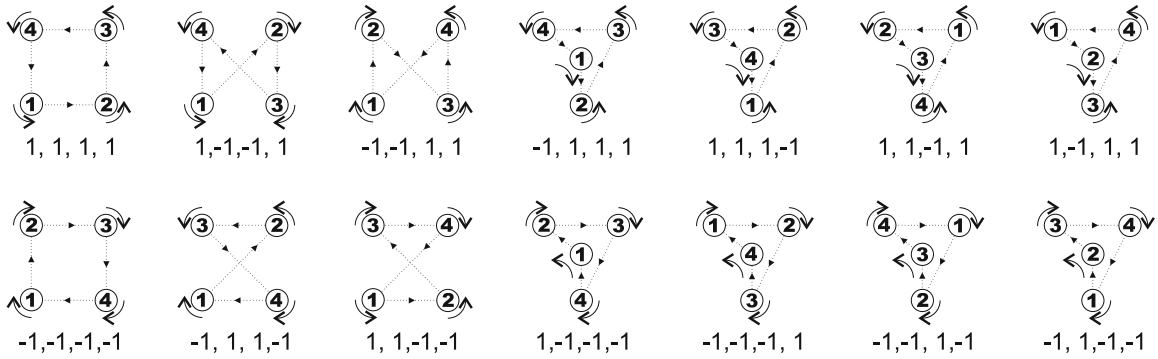
(b) Vektorový součin se směrem otočení doprava. Převzato z [15].

Obrázek 6.5: Ukázka měnícího se znaménka v závislosti na směru natočení druhého vektoru.

Na základě této znalosti je možné určit, zda je vypočítaná transformační matice degenerativní, což se vyznačuje tím, že matice poruší konkávnost objektů. Na obrázku 6.6 můžeme vidět, že konkávní tvary v prvním sloupci mají vždy stejné natočení vektorů, které tvoří obvod tvaru, zatímco zbylé sloupce obsahují konvexní tvary, ve kterých se směr alespoň jednou změní.

Pro kontrolu transformace tedy vytvořím 4 body tvořící obdélník, který představuje okraj obrázku. Na tyto body je aplikována transformace a jsou vypočítány vektory tvořící obvod tvaru. Z těchto vektorů je pomocí skalárního součinu určeno natočení a pokud objektu byla zachována konkávnost, je transformace v pořádku.

Po snížení třídy vypočítané matice z projektivní na afinní už tato funkce není potřeba, jelikož afinní transformace nemůže takto deformovat obrázek.



Obrázek 6.6: Ukázka závislosti konkávnosti na základě směru natočení vektorů, tvořící obvod. V prvním sloupci jsou tvary konkávní, protože mají všechny směry natočení stejné. Převzato z [15].

IsTransformExtreme

Druhá metoda pouze omezuje extrémní transformace, které by například měly rotaci vetší jak 50°, výrazně by měnily poměr stran nebo převrátily obrázek kolem osy x. Zároveň pokud byly vstupní body zrcadleny, povolí tato metoda zrcadlení kolem osy y, k tomuto se využívá výše zmíněný příznak *allowYFlip*. Tato metoda byla využívána spíše jako hrubý filtr, který by měl zachytit špatně roztříděné obrázky pozic.

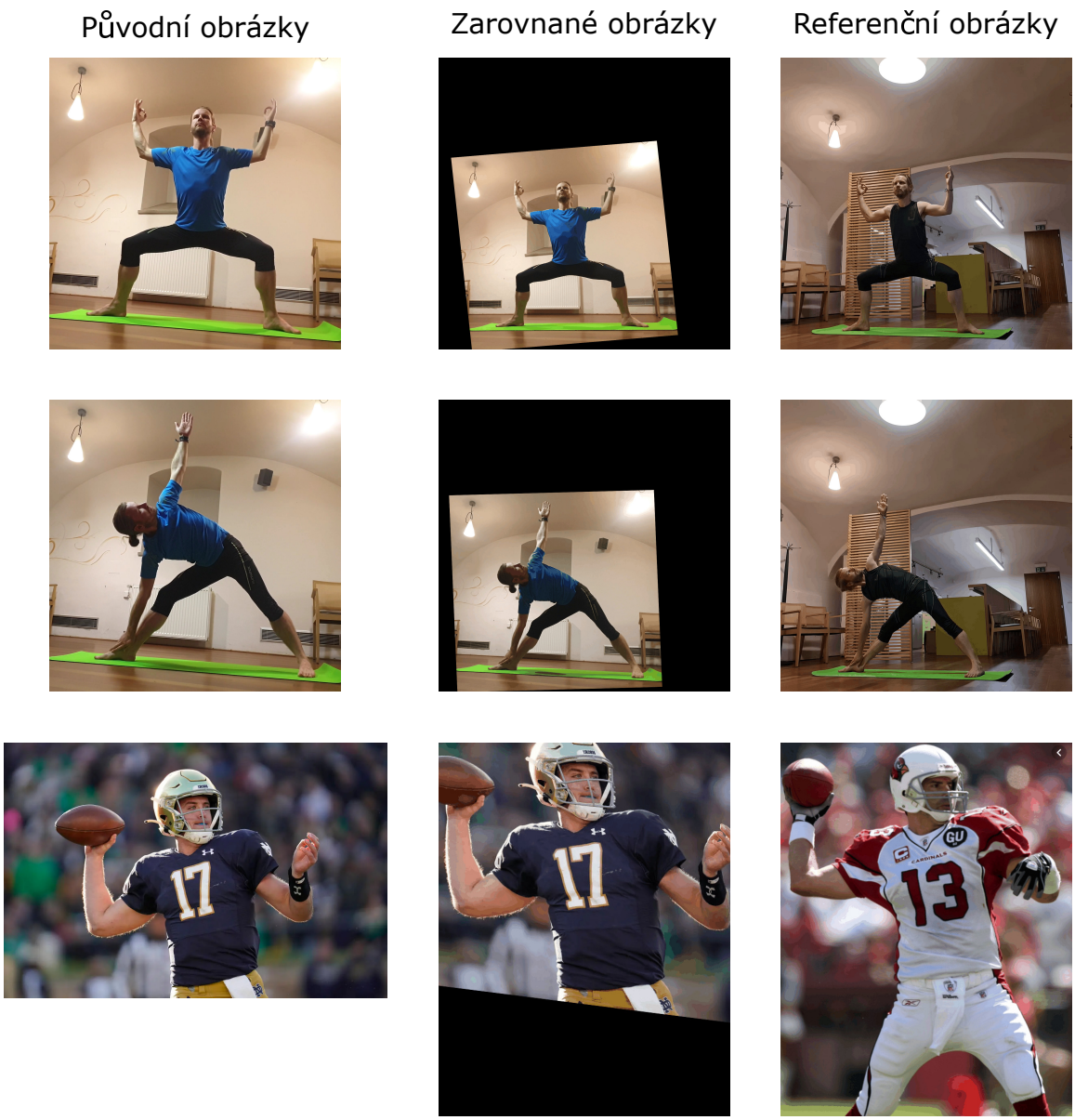
6.2.8 Hodnocení modelu

Model bude hodnocen klasickým způsobem, tedy podle počtu párů, ve kterých se po aplikaci transformace body překrývají. Jelikož části těla v obrazu zabírají značnou část pixelů a nemusí být vždy detekovány správně, body budou považované za překryté, pokud jejich vzdálenost bude menší jak 50 pixelů.

V raném stádiu implementace byly navrženy dvě další možnosti hodnocení, které po snížení stupňů volnosti už nebylo potřeba implementovat, ale stojí za zmínku. První hodnocení by se počítalo podobně jako klasické, ale páry kloubů by byli ohodnocené. Pro lepší zarovnání podle pózy se zdá rozumnější klást vetší důraz na body tvořící páteř, než na konce rukou, jejichž pozice se často liší kvůli jinému zaujmutí dané pozice. Druhé hodnocení by se počítalo jako plocha, kterou pokrývá mnohoúhelník, tvořený inliery. Tato metoda by značně podhodnotila matice, které zarovnají hlouček blízkých bodů, ale nezarovnají správně končetiny a obecně příliš deformují obrázek. Takovéto hloučky většinou vznikaly v oblasti krku a pánve v pozicích, ve kterých byly v této oblasti ruce.

6.2.9 Aplikace transformace a uložení výsledků

Po dokončení výpočtu transformace jsou načtené oba obrázky, ze kterých byly získány klíčové body. Na tyto obrázky je aplikována transformace pomocí funkce *warpPerspective* z knihovny *OpenCV* a výsledné obrázky jsou uloženy do složky *aligned* jako gif soubor. Zarovnané obrázky je možné vidět v obrázku 6.7. Zároveň je uložen záznam o zarovnání ve formátu JSON, který obsahuje cestu k oběma obrázkům a transformační matici. Záznamy jsou ukládány do složky *MachineData*, ve které jsou dále děleny na trénovací a testovací sady.



Obrázek 6.7: Ukázka zarovnaných sportovních pozic. Na třetím obrázku lze vidět, že tento systém zvládá zarovnat i obrázky s různým rozlišením.

6.3 Implementace konvoluční neuronové sítě

K implementaci tohoto systému byla využita knihovna *pytorch*, spolu s platformou *Jupyter notebook*. Jupyter notebook byl zvolen především kvůli možnosti spouštění jednotlivých buněk v různém pořadí, díky kterému nebylo nutné opakovaně načítat data při změnách parametrů. Výpočty neuronových sítí byly akcelerovány grafickou kartou díky technologii CUDA. Průběh trénování byl zaznamenávám pomocí knihovny TensorBoard.

6.3.1 Tvorba modelu

Prvním krokem je vytvoření samotného modelu sítě. Při využití knihovny pytorch se jedná o implementaci abstraktní třídy `torch.nn.Module`. Tato třída musí implementovat klasický konstruktor `__init__`, ve kterém se vytvoří všechny vrstvy, a metodu `forward`, která definuje průchod dat skrz implementovanou síť tím, že postupně předává data do jednotlivých vrstev. Zároveň byla implementována metoda *countParameters*, která vrátí počet parametrů v dané síti.

6.3.2 Vstup neuronové sítě

Vstupem neuronové sítě jsou vždy dva čtvercové obrázky s předem určeným rozlišením. Rozlišení je možné nastavit do proměnné na začátku implementovaného notebooku, podle které zbytek systému bude upravovat načítaná data a vrstvy neuronové sítě. Důležité je si dát pozor, aby hodnota byla dělitelná 8, jinak může nastat chyba při výpočtu velikosti plně propojené vrstvy následující po konvolučních vrstvách.

6.3.3 Výstupní formáty

Výstupem je transformace zarovnání, kterou jsem zkoušel reprezentovat různými způsoby a sledoval jsem, jak ovlivní učící cyklus. Některé formáty vyžadovaly různé funkce a úpravy při načítání dat.

Transformační matice

První formát, který byl vyzkoušen je prvních 6 hodnot transformační matice, které jsou potřeba pro afinní transformaci. Jelikož se v tomto formátu překrývají hodnoty rotace a změny měřítka, bylo pro neuronovou síť obtížné správně odhadovat transformace. Další nevýhodou bylo velmi odlišné rozložení hodnot jednotlivých prvků v matici. Z tohoto důvodu bylo nutné vytvořit vážené ztrátové funkce, ale i tak nevykazovala síť dobré výsledky.

Oddělené prvky transformací

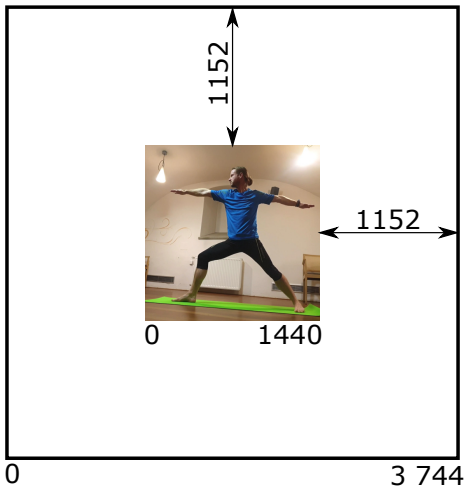
Druhým formátem byly jednotlivé části transformací, jelikož jsem sklon nepovažoval za podstatnou část transformace pro zarovnání fotografií, použil jsem pouze 5 hodnot, a to posuny a změnu měřítka na obou osách a rotaci. Jelikož záznamy generované prvním systémem obsahují celou transformační matici, byly tyto prvky získávány pomocí dekompozice transformační matice. Tento formát dosáhl lepších výsledků, než přímým odhadem transformační matice, ale výsledky stále nebyly dostačující.

Rohové body

Po neúspěchu s předešlými formáty byl vyzkoušen formát popsaný v práci *Deep Image Homography Estimation* [6], který reprezentuje transformaci zarovnání pomocí rohových bodů, které je možné 1:1 mapovat na transformační matici. Mapování do tohoto formátu je možné pomocí aplikace matice na předem zvolené rohové body a z těchto bodů je možné získat transformační matici regresí, která je popsána v sekci 2.2.

Klasifikace segmentů regrese

Další variantou výstupu je některý z předešlých formátu reprezentovat pomocí klasifikace segmentů regrese. Tento přístup byl vyzkoušen pouze na formátu rohových bodů. Hlavní myšlenkou je rozdělit obor hodnot regrese do tříd, které budou následně klasifikovány neuronovou sítí. Při implementaci byl hlavní problém neznalost oboru hodnot odhadnutých souřadnic rohů. Možným řešením by bylo využít obor hodnot o velikosti trojnásobného rozlišení obrazu, čímž bych jistě pokryl všechny transformace, ale většina krajních tříd by nebyla využita. Nakonec byl využil obor hodnot, který rozšíří souřadný systém o 80 % na všechny strany a byly vyfiltrovány některé extrémní zarovnání, které do tohoto oboru nespadaly. Rozšíření oboru hodnot ve kterém je možné detekovat roh lze vidět na obrázku 6.8.



Obrázek 6.8: Rozšíření oboru hodnot o 80 % na všechny strany, pro klasifikaci rohů. Výsledný prostor má rozlišení 3744 × 3744.

6.3.4 Načítání dat

V dalším kroku bylo implementováno načítání dat. Tento krok byl opět z velké části vyřešen pomocí abstraktní třídy `torch.utils.data.Dataset`, ve které je potřeba definovat metody `__init__`, `__len__` a `__getitem__`. Tato třída byla implementována dvěma způsoby.

První způsob načte a připraví všechna data z disku, uloží si je do seznamu a v metodě *getitem* pouze vrací hodnotu z seznamu. Tento přístup je více náročný na paměť a tato příprava může trvat i několik desítek minut, ale při více epochách není nutné opakovaně načítat a upravovat data. Hlavním důvodem pro využití tohoto způsobu bylo generování dat v prvních fázích implementace, při kterém bylo z načítaného obrázku vytvořeno několik záznamů v seznamu, pomocí náhodné transformace, která byla aplikována na vstupní obrázky. Po aplikaci transformace na obrázky bylo nutné aplikovat inverzní transformaci na matici zarovnání, aby zarovnání stále platilo. Původně byly obrázky po těchto úpravách ořezávány, aby po transformaci obrázky neobsahovaly černé pixely, ale ukázalo se, že nemají na učení sítě zásadní vliv.

V druhém přístupu se v konstruktoru vytvoří seznam s názvy souborů, které jsou v metodě getitem pokaždé načteny a zpracovány. Zároveň byl postup využitý ke generování dat využit jako augmentace dat.

V obou případech se jedná o stejné zpracování dat, které spočívá v konverzi na šedotónové obrázky, změny rozlišení obrázků tak, aby odpovídalo vstupnímu rozlišení sítě, a konverzi transformační matice do výstupního formátů.

Díky tomu k implementaci načítání dat byla využita třída Dataset je možné využít `torch.utils.data.DataLoader`, kterému se v konstruktoru předá instance třídy pro načítání dat a parametry, které určují, jak se mají načítat. Hlavním parametrem je *batch_size*, který určuje kolik záznamů dat bude v jednom průchodu sítě, při trénování. Díky možnosti paralelního zpracování dat je možné několikanásobně zkrátit čas nutný k trénování sítě a proto se nastavuje hodnota tohoto parametru tak vysoká, jak umožňuje velikost paměti. Obvykle se velikost dávky nastavuje v mocninách dvou, kvůli architektuře a optimalizaci na grafických kartách. V mé implementaci je využita velikosti dávek 16 a 32. Dalším parametrem je *shuffle*, kterým se nastaví náhodné zamíchání dat. Tato funkce byla využita, jelikož jsou záznamy o zarovnání organizovány podle pozice a může nastat situace, ve které by několik dávek jdoucí po sobě, obsahovalo stejnou pózu, což by mohlo negativně ovlivnit učení.

6.3.5 Trénování sítě

Po vytvoření modelu a systému pro načítání následovala implementace trénování sítě. Nejdříve je vytvořena instance tříd pro načítání dat a instanci třídy `SummaryWriter`, která slouží k ukládání dat o průběhu učení do souborů na disk, které následně lze procházet ve webové aplikaci *TensorBoard*. Název ukládaných záznamů je tvořen z data spuštění, verze neuronové sítě a parametrů optimalizátoru. Dále je vytvořena instance optimalizátoru *Adam* s parametrem *learning rate* nastaveným na 0,0001 a parametrem *weight decay* na 0,000001. V rámci implementace byl vyzkoušen optimalizátor SGD, u kterého ovšem nastal jev zvaný *Exploding Gradient* (vybuchující gradient) a síť vracela hodnoty NaN. Tento problém byl vyřešen pomocí ořezávání gradientu, ale ukázalo se že s SGD byl průběh ztrátové funkce mírně horší, a proto nebyl dále nevyužíván.

Samotná trénovací smyčka byla implementována velmi standardně. Jak bylo popsáno v části 6.3.4, k načítání dat byla využita třída DataLoader, která je iterovatelná a tedy je možné ji použít ve for smyčce. Program tedy prochází data po dávkách a předává je síti, jejíž výsledky jsou porovnány pomocí ztrátové funkce s cílovou hodnotou. V rámci implementace byly vyzkoušeny různé předem implementované ztrátové funkce, mezi které patří MSE, MAE a CrossEntropy, které jsou popsané v sekci 4.4.1. Zároveň byla implementována vlastní funkce *Mean Corner Error* (dále jen MCE). Při implementaci vlastní ztrátové funkce bylo nutné počítat s možností výpočtu ztráty nad celou dávkou, při které byly hodnoty jednotlivých záznamů průměrovány. Po výpočtu funkce je provedena optimalizace parametrů a výsledek ztrátové funkce každé 4. dávky je uložen.

Po průchodu skrz trénovací sadu následuje průchod skrz testovací sadu, který je stejný, až na to že je síť přepnuta do evaluačního režimu, při kterém jsou vypnuty dropout vrstvy a zároveň je testování prováděno v *torch.no_grad()* kontextu, ve kterém nejsou počítány gradienty pro optimalizaci, čímž se zmenšuje výpočetní náročnost a testování může být provedeno rychleji.

Průchod skrz trénovací a testovací smyčku je zakončen uložením průměrných hodnot ztrátové funkce z obou datových sad. Jak bylo popsáno v sekci 4.4, jeden takovýto průchod se nazývá epocha a rámci mé implementace bylo vyzkoušeno trénování sítě s počtem epoch v rozmezí 50 až 200, ale finální experimenty měli horní hranici epoch nastavenou na 100. Jelikož u regresních sítí nedocházelo k přetrénování a hodnota ztrátová funkce nad

testovacími daty ve vyšších epochách spíše začala stagnovat, nebylo nutné implementovat *early stopping*. U klasifikace k přetrénování ovšem docházelo, a proto byla implementována jednoduchá varianta, která přeruší učení, pokud se třem po sobě jdoucím epochám nesníží hodnota ztrátové funkce na testovacích datech.

6.3.6 Hodnocení a uložení výsledků

Po natrénování sítě následuje poslední průchod testovací sadou pro finální ohodnocení a uložení zarovnaných obrázků. Sada je procházena po dávkách o velikosti jednoho záznamu, ze kterého je vypočítána transformace. Z vypočítané transformace je dále vypočítána hodnota MCE. Obrázky jsou zarovnány podle vypočítané matice i podle cílové hodnoty a jsou spolu s referenčním obrázkem uloženy do gif souboru. Ukázku zarovnání je možné vidět na obrázku 6.9.

Obrázek 6.9: Ukázka obrázků zarovnaných pomocí neuronové sítě. V první póze je naznačeno mapování na odhadnuté rohy.

Hodnoty MCE ze všech záznamů jsou uloženy do textového souboru pro případné další využití. Dále je vypočítán průměr hodnot MCE z celé sady, který slouží jako primární ohodnocení modelu. Jelikož průměr může být zavádějící kvůli různému rozložení hodnot, implementoval jsem jednoduchý výpočet procentuálního podílu správně zarovnaných obrázků, který je počítán s různou tolerancí. Určit toleranci, při které je možné obrázek považovat za zarovnaný nebylo snadné, jelikož hodnota MCE slouží spíše jako hrubý odhad chyby zarovnání. Obrázek s hodnotou MCE 180 se může od pohledu zdát hůře zarovnaný než obrázek s hodnotou MCE 230, jelikož pozice cvičícího a velikost plochy, kterou na obrázku zabírá, ovlivňuje význam hodnoty MCE. Pro většinu obrázků platí, že po aplikaci transformace s hodnotou MCE menší jak 200, jsou dobře zarovnané.

Kapitola 7

Experimenty

V této kapitole jsou popsány experimenty s implementovanou neuronovou sítí. Architektura sítě zůstává v rámci většiny experimentů stejná, mění se pouze vstupní a výstupní vrstva tak, aby odpovídala vstupnímu rozlišení a výstupnímu formátu transformace. Experimenty byly prováděny na notebooku HP Pavilion Power s procesorem Intel Core i5 9300H a grafickou kartou NVIDIA GeForce GTX 1650 s pamětí 4GB. Pro porovnání budou využity výsledky, jejichž ukládání je popsáno v sekci 6.3.6.

Jak je popsáno v sekci 6.3.3, zdaleka nejlepší výstupní formát jsou rohové body, ze kterých se následně počítá transformace. Z tohoto důvodu jsou experimenty prováděny pouze nad tímto formátem. Provedené experimenty sledují kvalitu zarovnání při využití regresní a klasifikační neuronové sítě, při různých vstupních rozlišeních. Dále je také vyzkoušena varianta, ve které se na vstup sítě předávají barevné obrázky. V tabulce 7.1 lze vidět všechny varianty, které budou v této kapitole popsány.

Před provedením experimentů bylo vyzkoušeno zda je neuronová síť schopná zrcadlit obrázky, stejně jako první systém. Neuronová síť tento úkol nezvládla a bylo nutné takovéto záznamy filtrovat při dalších experimentech. Zároveň bylo vyzkoušeno jak neuronová síť dokáže zarovnat obrázky z jiných sportů, ale kvůli jejich rozlišení a odlišnosti od datové sady na které byla neuronová síť trénována, je nebyla schopná zarovnat.

Typ sítě	**Barevný vstup**	**Rozlišení**	**MCE**
Regresní	ano	128×128	140,21
Regresní	ne	128×128	149,66
Regresní	ne	164×164	131,71
Regresní	ne	224×224	130,81
Klasifikační	ne	128×128	153,25
Klasifikační	ne	164×164	149,77

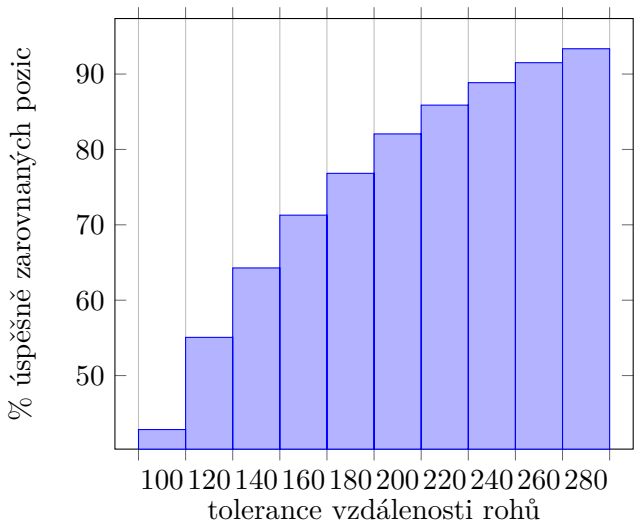
Tabulka 7.1: Přehled experimentů s průměrnou hodnotou MCE.

7.1 Regresní model

Pro experimenty s regresní sítí byla využita ztrátová funkci MAE, která byla zvolena kvůli snaze minimalizovat chybu všech rohů, místo minimalizace větších odchylek, které provádí funkce MSE.

Podle očekávání se přesnost zarovnání zvedá se zvyšujícím rozlišením, ale u přechodu z rozlišení 164×164 na 224×224 klesla průměrná hodnota MCE pouze o 0,9. Důvodem pro tento jev je pravděpodobně zmenšení velikosti dávek z 32 na 16, které zároveň velmi prodloužilo dobu učení. Kvůli tomuto jevu nebude experiment s tímto rozlišením dále prezentován a nejlepších výsledků tedy dosáhl model s rozlišení 164×164, jehož distribuci procentuální úspěšnosti je možné vidět na obrázku 7.1.

Dále bylo vyzkoušeno trénování sítě s barevným vstupem, čímž se trojnásobně zvětšil počet informací na vstupu sítě. Podle očekávání tyto dodatečné informace pomohly síti zpřesnit vypočítané transformace. Zpřesnění ovšem nebylo tak velké jako při zvětšení rozlišení vstupního obrazu.



Obrázek 7.1: Graf ukazující procentuální úspěšnost zarovnání, na základě tolerance. Hodnoty grafu byly získány z regresního modelu se vstupním rozlišením 164×164.

7.2 Klasifikační model

Pro experimenty s klasifikačním modelem byla zvolena ztrátová funkce Cross Entropy. Výsledek těchto experimentů je značně horší než u regresního modelu. Hodnota MCE klesla s narůstajícím rozlišením, ale výsledky těchto experimentů byly nestabilní.

7.3 Shrnutí

Výsledky experimentů dokazují že neuronové sítě jsou schopné se naučit řešit tento problém. Přesnost zarovnání reprezentováné hodnotou MCE 131.71 získánou regresním modelem se vstupním rozlišením 164×164 se blíží hodnotě MCE 103,5[1], jenž dosáhli v práci *Deep Image Homography Estimation* [6]. Jelikož je zarovnání v této práci počítané na základě pozice a ze dvou různých obrázků, na rozdíl od generovaných dat využitých k trénování sítě *HomographyNet*, je rozdíl v přesnosti detekovaných rohů menší než jaký byl očekáván.

[1] Tato hodnota vznikla přepočítáním z rozlišení 128×128 na mnou požívané rozlišení 1440×1440.

Kapitola 8

Závěr

Cílem této práce bylo nastudovat techniky zarovnání obrazu, regrese transformací a detekce lidské pózy, následně navrhnout systémy pro zarovnání obrázků na základě sportovní pozice, využívající nastudovaných znalostí, pro navržení systému využít jak metody k dosažení přesného zarovnání, tak i metody které nejsou výpočetně náročné a navržené systémy implementovat a otestovat na vhodně vytvořené datové sadě.

V práci je popsán úvod do oboru zarovnání obrázků, ve kterém jsou popsány transformace a jejich výpočet ze dvojic klíčových bodů. Zároveň je zde popsán algoritmus RANSAC, který se využívá pro filtraci špatně detekovaných bodů.

Jelikož se k detekci lidské pózy téměř vždy využívají neuronové sítě, místo studování technik je popsán průzkum existujících řešení a jejich základní principy fungování.

Dále jsou popsány principy fungování neuronových sítí a znalosti potřebné k jejich implementaci. Zároveň jsou zde popsány dvě existující řešení, které využívají neuronové sítě k výpočtu zarovnání.

Na základě získaných znalostí byly navrhnuty implementovány dva systémy pro výpočet zarovnání. První systém využívající knihovnu OpenPose k detekci lidské pózy, na základě které se vypočítá transformace zarovnání, pomocí algoritmu RANSAC. Tento systém bude vypočteněji náročnější, ale zarovnání bude velmi přesné. Druhý systém je založen na konvolučních neuronových sítích, díky čemuž by měl být méně výpočetně náročný než první systém.

Pro vylepšení implementované neuronové sítě byla provedena řada experimentů, porovnávajících výsledky na základě různých formátů vstupu a výstupu. Díky těmto experimentům jsem našel optimální model, který dokázal zarovnat 81,98 % obrázků s tolerancí hodnoty MCE 200 pixelů. I přes to, že síť byla navrhnuta tak, aby byla méně náročná na hardware, dosáhla relativně dobrých výsledků na mnou vytvořené datové sadě. Jelikož mnou vytvořená sada pravděpodobně není dostatečně různorodá, nelze očekávat dobré výsledky na obrázcích, které jsou pořízeny v jiných podmínkách.

Největší potenciál vidím v rozšíření datové sady, především by bylo dobré do ní zahrnout pozice z více sportů a prostředí. Pokud by byl tento či podobný systém využit v aplikaci, určitě by bylo dobré využít *crowdsourcing* k získání fotografií, jelikož získávání fotografií z jiných sportů se ukázalo dost obtížné. Dalším možným rozšířením je upuštění od požadavku na minimální hardwarovou náročnost a rozšířit architekturu neuronové sítě, v takovém případě by pravděpodobně bylo nutné provozovat zarovnání spíše jako *cloudovou* službu.

Literatura

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

[2] Barla, N. *A comprehensive guide to human pose estimation* [[online]]. 2022. [cit. 30-03-2022]. Dostupné z: https://www.v7labs.com/blog/human-pose-estimation-guide#what-is-human-pose-estimation.

[3] Bishop, C. M. Neural networks and their applications. *Review of Scientific Instruments*. 1994, sv. 65, č. 6, s. 1803–1832. DOI: 10.1063/1.1144830. Dostupné z: https://doi.org/10.1063/1.1144830.

[4] Cao, Z., Hidalgo, G., Simon, T., Wei, S.-E. a Sheikh, Y. *OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields*. DOI: 10.48550/ARXIV.1812.08008. Dostupné z: https://arxiv.org/abs/1812.08008.

[5] Derpanis, K. G. Overview of the RANSAC Algorithm. In:. 2005.

[6] DeTone, D., Malisiewicz, T. a Rabinovich, A. Deep Image Homography Estimation. *CoRR*. 2016, abs/1606.03798. Dostupné z: http://arxiv.org/abs/1606.03798.

[7] Fischler, M. A. a Bolles, R. C. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. New York, NY, USA: Association for Computing Machinery. 1981, sv. 24, č. 6. DOI: 10.1145/358669.358692. ISSN 0001-0782. Dostupné z: https://doi.org/10.1145/358669.358692.

[8] Fukushima, K. Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. *Biological Cybernetics*. 1980, sv. 36, s. 193–202.

[9] Goodfellow, I., Bengio, Y. a Courville, A. *Deep Learning*. MIT Press, 2016. Dostupné z: http://www.deeplearningbook.org.

[10] (https://math.stackexchange.com/users/741822/hypnomaki) hypnomaki. *Calculate Homography with and without SVD* [Mathematics Stack Exchange]. Dostupné z: https://math.stackexchange.com/q/3509039.

[11] Hubel, D. H. a Wiesel, T. N. Receptive fields of single neurones in the cat's striate cortex. *The Journal of Physiology*. 1959, sv. 148, č. 3, s. 574–591. DOI: https://doi.org/10.1113/jphysiol.1959.sp006308. Dostupné z: https://physoc.onlinelibrary.wiley.com/doi/abs/10.1113/jphysiol.1959.sp006308.

[12] Hébert Losier, K., Hanzlíková, I., Zheng, Streeter, L. a Mayo, M. The 'DEEP' Landing Error Scoring System. *Applied Sciences*. Leden 2020, sv. 10, s. 892. DOI: 10.3390/app10030892.

[13] Kowalski, M., Naruniec, J. a Trzcinski, T. Deep Alignment Network: A Convolutional Neural Network for Robust Face Alignment. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. July 2017.

[14] Mitchener, P. D. Planar Transformations. 2013. Dostupné z: http://www.mitchener.staff.shef.ac.uk/Transformations.pdf.

[15] Monnin, D., Bieber, E., Schmitt, G. a Schneider, A. An Effective Rigidity Constraint for Improving RANSAC in Homography Estimation. In: Blanc Talon, J., Bone, D., Philips, W., Popescu, D. a Scheunders, P., ed. *Advanced Concepts for Intelligent Vision Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, s. 203–214. ISBN 978-3-642-17691-3.

[16] Newell, A., Yang, K. a Deng, J. Stacked Hourglass Networks for Human Pose Estimation. *CoRR*. 2016, abs/1603.06937. Dostupné z: http://arxiv.org/abs/1603.06937.

[17] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. a Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*. 2014, sv. 15, č. 56, s. 1929–1958. Dostupné z: http://jmlr.org/papers/v15/srivastava14a.html.

[18] Stasiuk, A. *Pose estimation. metrics.* [[online]]. 2020. [cit. 31-03-2022]. Dostupné z: https://stasiuk.medium.com/pose-estimation-metrics-844c07ba0a78.

[19] Stevens, E., Antiga, L. a Viehmann, T. *Deep Learning with PyTorch*. Manning Publications, 2020. ISBN 9781617295263. Dostupné z: https://books.google.cz/books?id=89BlwwEACAAJ.

[20] Sun, K., Xiao, B., Liu, D. a Wang, J. Deep High-Resolution Representation Learning for Human Pose Estimation. *CoRR*. 2019, abs/1902.09212. Dostupné z: http://arxiv.org/abs/1902.09212.

[21] Szeliski, R. Image Alignment and Stitching: A Tutorial. *Found. Trends Comput. Graph. Vis.* 2006, sv. 2, č. 1. Dostupné z: https://www.microsoft.com/en-us/research/wp-content/uploads/2004/10/tr-2004-92.pdf.

[22] Szeliski, R. *Computer vision algorithms and applications*. London; New York: Springer, 2011. ISBN 9781848829343 1848829345 9781848829350 1848829353. Dostupné z: http://dx.doi.org/10.1007/978-1-84882-935-0.

[23] Toshev, A. a Szegedy, C. DeepPose: Human Pose Estimation via Deep Neural Networks. *CoRR*. 2013, abs/1312.4659. Dostupné z: http://arxiv.org/abs/1312.4659.

[24] Yamashita, R., Nishio, M., Do, R. a Togashi, K. Convolutional neural networks: an overview and application in radiology. *Insights into Imaging*. Červen 2018, sv. 9. DOI: 10.1007/s13244-018-0639-9.

[25] Zheng, C., Wu, W., Yang, T., Zhu, S., Chen, C. et al. Deep Learning-Based Human Pose Estimation: A Survey. *CoRR*. 2020, abs/2012.13392. Dostupné z: https://arxiv.org/abs/2012.13392.