

# GYMNASIUM JANA KEPLERA

Parléřova 2/118, 169 00 Praha 6



## SimonPose

### Maturitní práce

Autor: Šimon Brávek

Třída: 4.B

Školní rok: 2025/2026

Předmět: Informatika

Vedoucí práce: Jiří Matas

Praha, 2026





Gymnázium Jana Keplera

Kabinet informatiky

# ZADÁNÍ MATURITNÍ PRÁCE

---

- *Student:* Šimon Brávek
  - *Třída:* 4.B
  - *Školní rok:* 2025 / 2026
  - *Vedoucí práce:* Jiří Matas
  - *Název práce:* Vylepšení odhadu pózy člověka z jednoho vstupního snímku
  - *URL repozitáře:* <https://github.com/simonbravek/pose-detection-project>
- 

## Pokyny pro vypracování:

Cílem práce je navrhnut, implementovat a experimentálně ověřit metodu pro **přesnější odhad lidské pózy** z jednoho RGB snímku.

Student využije již existující model **DensePose** pro získání hustého korespondenčního mapování povrchu těla a naváže na něj vlastní optimalizační nebo post-processingovou metodu s cílem:

- snížit chyby v odhadu pozice a orientace klíčových kloubů,
- případně zvýšit robustnost vůči šumu a odlišnostem v pozici či osvětlení vstupního snímku.

## Součástí práce je:

- rešerše současných přístupů k odhadu pózy,
- implementace vlastního řešení v prostředí Python (TensorFlow),
- návrh metrik pro hodnocení přesnosti a srovnání s výchozím stavem,
- experimentální vyhodnocení na vybraném datasetu,
- prezentace dosažených výsledků a postupu práce formou závěrečné zprávy a ukázky funkčního systému.

Výstupy práce:

- zdokumentovaný software v repozitáři,
  - technická zpráva shrnující postup, metodiku a dosažené výsledky,
  - prezentace práce a její obhajoba.
- 

Podpis vedoucího práce:

A handwritten signature in black ink, appearing to read "J. Matas".

## **Prohlášení**

Prohlašuji, že jsem svou práci vypracoval samostatně a použil jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů. Nemám žádné námitky proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších předpisů.

V Praze dne 1. března 2026

Šimon Brávek



## **Deklarace užití nástrojů umělé inteligence**

V rámci přípravy této maturitní práce byly využity nástroje generativní umělé inteligence (AI). Ta byla použita k řešení dílčích úkolů, zejména pro ladění a optimalizaci zdrojového kódu a stylistickou úpravu textu. Autor práce provedl kritické zhodnocení všech výstupů a nese plnou odpovědnost za konečný obsah a správnost textu.



## **Poděkování**

Děkuji Jiřímu Matasovi za skvělou příležitost strávit měsíc v létě na katedře strojového učení a být součástí soudobého výzkumu a za jeho skvělé nápady v tomto oboru. Bezpochyby tato zkušenost formovala mé rozhodnutí studovat obor AI na MFF UK a propůjčila mi vášeň k nejnovějším technologiím. Děkuji také Matěji Suchánkovi jako člověku, co mě provedl tou nejtěžší asimilací s práci s grafickými kartami a modely jež jsem používal. Děkuji také lidem z katedry, jež ke mě byli přátelští a v neposlední řadě Igoru Vujovičovi za rozvíjení informatiky na našem gymnáziu i přez mnohé překážky.



## **Abstrakt**

Jde o explorativní analýzu s cílem přinést metody pro zlepšení human pose estimation.

## **Klíčová slova**

počítacové vidění, odhad lidské pozice, DensePose, SMPL/SMPL-X,  
rekonstrukce z 2D do 3D, hluboké učení, vylepšení modelu

## **Abstract**

## **Keywords**

computer-vision, human-pose-estimation, DensePose, SMPL/SMPL-X,  
2D-to-3D reconstruction, deep-learning, model improvement



# Obsah

<b>Slovník pojmů</b>	<b>3</b>
<b>Úvod</b>	<b>5</b>
0.1 Analýza postavy v digitálním světě . . . . .	5
0.2 Problematika . . . . .	5
0.3 Cíl práce . . . . .	5
<b>1 Teoretická část</b>	<b>7</b>
1.1 Základní teorie . . . . .	7
1.1.1 Co je to model? . . . . .	7
1.1.2 Druhy detekcí . . . . .	7
1.1.3 Odhad 2D a jeho limitace . . . . .	7
1.1.4 Od 2D bodů k 3D objemu . . . . .	8
1.2 Skinned Multi-Person Linear Model (SMPL) . . . . .	8
1.2.1 Výpočet . . . . .	8
1.2.2 Výhody . . . . .	10
1.3 DensePose . . . . .	10
1.3.1 DensePose v1: mapování pomocí (I, U, V) . . . . .	11
1.3.2 DensePose v2: Continuous Surface Embeddings (CSE) . . . . .	11
1.3.3 Silné a slabé stránky . . . . .	12
<b>2 Explorativní část</b>	<b>15</b>
2.1 Embedding fitter . . . . .	15
2.1.1 Návrh metody . . . . .	15
2.1.2 Implementace . . . . .	16
2.1.3 Hodnocení . . . . .	17
2.2 Euklidovský fitter . . . . .	18
2.2.1 Návrh metody . . . . .	18
2.2.2 Implementace . . . . .	19
2.2.3 Hodnocení a výsledky . . . . .	20
2.3 Zaměření na přesnost . . . . .	23
2.3.1 Návrh metody . . . . .	23
2.3.2 Implementace . . . . .	23
2.3.3 Výsledky . . . . .	23
<b>3 Technická dokumentace</b>	<b>25</b>
3.1 Struktura repozitáře . . . . .	25
3.2 Instalace . . . . .	25
3.3 Spuštění . . . . .	25
<b>Závěr</b>	<b>27</b>
<b>Seznam použité literatury</b>	<b>29</b>
<b>Seznam obrázků</b>	<b>31</b>
<b>Seznam tabulek</b>	<b>32</b>



# Slovník pojmu

**model** naučená neuronová síť, podrobněji v teoretické části

**segmentace** (segmentation) rozdělení obrázku na skupiny pixelů, typicky dle objektů, které popisuje. v kontextu této práce tím referuji ke skupině pixelů patřící detekovanému lidskému tělu.

**bounding box** je nejmenší obdélník obsahující celou segmentaci nějakého objektu, typicky se značí pomocí dvou bodů (levého horního rohu a pravého dolního rohu)

**detekce** (detection) je objekt zaznamenaný modelem, typicky pomocí bounding boxu

**maska** (mask) výsledek segmentace, je to černo-bílý obraz stejné velikosti jako originál překrývající segmentaci objektu, může být i binární maskou, kde k pixely patřící k objektu mají hodnotu 1 a ostatní hodnoty 0

**embedding** je vektorové pole zachycující význam obrázku, více v teoretické části



# Úvod

## 0.1 Analýza postavy v digitálním světě

V současné informatice a počítačovém vidění (Computer Vision) představuje automatické porozumění lidského pohybu jednu z největších výzev. Nejde již o prostou detekci zda se v obrázku nachází člověk, či kolik jich je, ale o snahu přenést lidskou biomechaniku do digitálního prostoru. Schopnost přesně interpretovat lidskou pózu z běžného 2D obrazu (např. z mobilního telefonu) otevírá dveře aplikacím, které byly dříve nemyslitelné bez drahých studiových systémů pro snímání pohybu (Motion Capture).

Tyto technologie se dnes nachází již v mnoha podobách a můžou tak pomoci v celé řadě aplikací. Dnes již můžeme najít rozpoznání člověka a obličeje v mobilní aplikaci galerie, kde napomáhá chytrému třízení fotek nebo v bezpečnostních systémech pro detekci větřelce. Detekce konkrétní pozice, tedy kde se nachází jednotlivé klouby, může zase pomoci zastavit automatický vozík před kolizí s člověkem nebo zavolat pomoc pokud uvidí nehybně ležícího chodce.

Kromě přesnosti těchto metod se vědci čím dál více zaměřují také na rychlosť. Mnohé aplikace totiž potřebují živou analýzu videa a to často i z několika kamer najednou. Tak tomu je u autonomního řízení automobilů, které se ukazuje být velkým tahounem tohoto odvětví.

Právě rozmanitost této disciplíny, její užitečnost a také možnost setkat se s ní do hloubky na FEL ČVUT mě vedlo k její volbě jako tématu mé maturitní práce.

## 0.2 Problematika

Současné SOTA (State-of-the-Art) modely sice dosahují vynikajících výsledků na laboratorních datasetech, ale často selhávají v reálných podmínkách (in-the-wild). Mezi kritické faktory patří:

- **Zákryty (Occlusions):** Části těla zakryté předměty nebo jinými částmi těla.
- **Perspektivní zkreslení:** Extrémní úhly kamery, které deformují vizuální proporce těla.
- **Vizuální šum:** Volné oblečení nebo špatné světelné podmínky.
- **Propletení více těl:** Situace v davu, kde není jasné, komu patří jaká část těla a jak je správně rozdělit.

Právě kombinace modelu SMPL s technologií DensePose (která mapuje pixely přímo na povrch těla) nabízí unikátní cestu, jak tyto problémy řešit skrze hustou korespondenci dat.

## 0.3 Cíl práce

Cílem této maturitní práce není vytvořit nový, revoluční model, který by překonal stávající vědecké rekordy. Ambicí je metodický průzkum možností, jak stávající proces fittingu (pasování) modelu SMPL do dat z DensePose zpřesnit a učinit jej odolnějším (robustnějším).

Zvolená metodika se opírá o princip Fail-Fast:

- Rychlá formulace hypotéz o vylepšení optimalizačního procesu.
- Implementace prototypů a jejich testování na hraničních případech.
- Analýza selhání jakožto hlavního zdroje poznání.

Výsledkem práce je ucelený přehled vyzkoušených metod, jejich kritické zhodnocení a dokumentace slepých i perspektivních uliček, které mohou sloužit jako inspirace pro další vývoj v oblasti monokulární 3D rekonstrukce člověka.

# 1. Teoretická část

## 1.1 Základní teorie

### 1.1.1 Co je to model?

V této práci se často budu věnovat programům, které využívají metodu strojového učení k detekci a odhadu pozice člověka v obrázku, který nazýváme model. Tento program má mnoho parametrů na základě nichž přetváří vstup na výstup. Nejdříve projde procesem, které nazýváme trénování. Během toho se jako vstup použije obrázek a program k němu vygeneruje souřadnice, kde odhaduje klouby, na základě prvního nastavení parametrů, neboli inicializace. Inicializace bývá velmi složitá a často je předmětem celých studií [1] a nebudeme se jí věnovat v této práci. Tento výsledek se pak porovná s ukázkovým příkladem pro tento obrázek, tedy pro anotaci, která byla předtím připravena člověkem.

Dále zvolíme metodu pro výpočet chyby výstupu (loss). Proces generování výstupu opakujeme mnohokrát pro velké množství obrázků a anotací toho. Skupině těchto dat říkáme trénovací dataset. Je nutné mít velké množství obrázků a anotací ještě před začátkem trénování. Ty pochází v drtivé většině případů od lidských anotátorů a jsou tak velmi drahé. Navíc jsou předmětem lidské chybovosti. Součtem, nebo jinou metodou tak spočítáme ztrátovou funkci (loss function) jako metriku chybovosti odhadu našeho modelu od anotací

### 1.1.2 Druhy detekcí

K tomuto účelu se využívá hned několik metod. Tou úplně nejzákladnější jsou rámečky k ohraničení detekovaných lidí. skládají se ze dvou souřadnic.

### 1.1.3 Odhad 2D a jeho limitace

Tradiční metody odhadu pózy se po léta soustředily na tzv. sparse keypoints – detekci klíčových bodů, jako jsou lokty, kolena či ramena – které detekují jako dvojici souřadnic v daném obrázku. Přestože jsou tyto modely (např. OpenPose) rychlé a efektivní, trpí zásadním nedostatkem: ztrátou prostorové informace a tělesného objemu. Zatímco mají perfektní výsledky v laboratorních podmínkách, selhávají v reálných situacích (in-the-wild), kde jim 2D obrázek neposkytuje dost jasné informace pro detekci pozice. Mezi faktory ovlivňující přesnost výstupu patří:

- **Zákryty (Occlusions):** Části těla zakryté předměty nebo jinými částmi těla.
- **Perspektivní zkreslení:** Extrémní úhly kamery, které deformují vizuální proporce těla.
- **Vizuální šum:** Volné oblečení nebo špatné světelné podmínky.
- **Propletení více těl:** Situace v davu, kde není jasné, komu patří jaká část těla a jak je správně rozdělit.

### 1.1.4 Od 2D bodů k 3D objemu

Řešení těchto složitých případu významně napomáhá představa 3D modelu těla a jeho pozice na dané scéně. Vezmeme-li situaci člověka skákajícího na lyžích zespodu, jeho tělesné proporce budou zcela nestandardní (malá hlava, velké nohy, spousta zakrytých částí těla). Pokud si ale představíme model lidského těla a promítnu ho do obrázku, pak jsem značně omezen a najít správné orientace končetin se značně zjednoduší. Představa těla ve 2D mi naopak dovoluje zvažovat pozice, jež by byly lidské tělo zcela nepřirozené, či anatomicky nemožné. Modely, které spolu s pozicí kloubů odhadují také 3D orientaci těla se díky tomu stávají lepší v samotném odhadování pozice kloubů.

Samotné rozpoznávání 2D pozice se dostává ke svým limitům také proto, že jejich přesnost se bliží datům na kterých jsou trénovány. [2]. Další k rozpoznání 3D pozice těla je zle

## 1.2 Skinned Multi-Person Linear Model (SMPL)

Přechod od 2D chápání člověka k pochopení objemové struktury lidského těla si vyžaduje zcela nové nástroje. Jedním z nich je způsob jak modelovat lidské tělo v prostoru. Abychom mohli tělo vyrendrovat na obrazovku pomocí standardních postupů, musíme povrch těla zapsat jako množinu bodů a zapamatovat si všechny možné trojice bodů tak, aby nám vznikla síťovina (mesh) reprezentující povrch 3D tělesa.

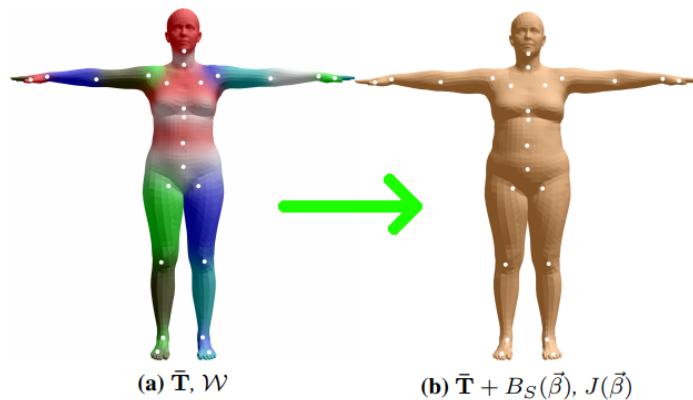
Namodelovat realistické lidcké tělo na základě bodů je velmi náročná disciplína. My bychom takové tělo chtěli modelovat automaticky, v reálném čase a s tělesnými proporcemi a pozicí, jakou si zadáme. Proto vznikl SMPL model, který na základě parametrů  $\beta$  a  $\theta$  vytvoří síťovinu libovolného člověka.

### 1.2.1 Výpočet

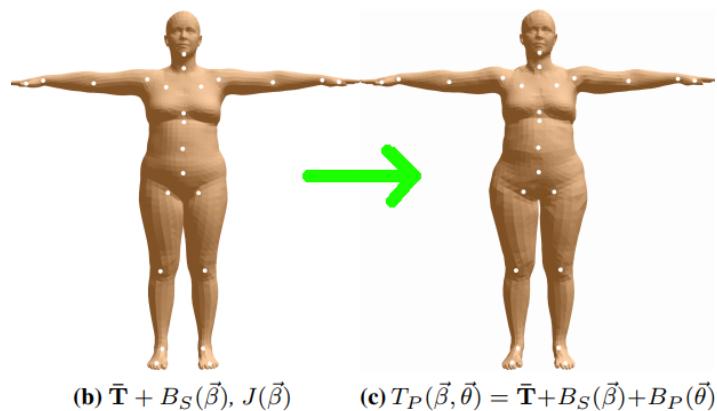
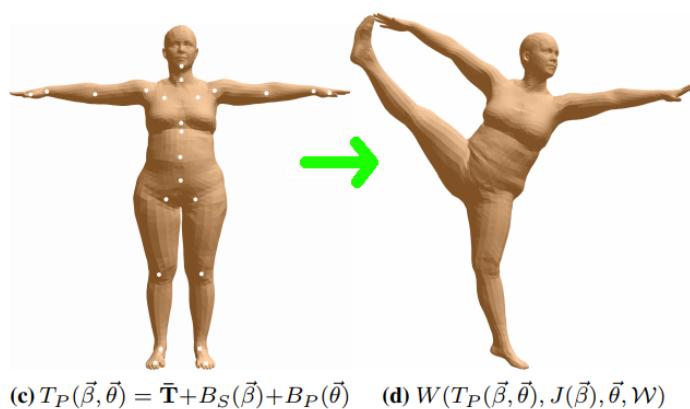
Celý model těla vzniká tak, že se začne se souřadnicemi bodů kanonického těla  $\tilde{T}$  v klidu (tedy univerzálního zvoleného těla, které zobrazuje průměrného člověka ve všech proporcích). Ke každému bodu se následně přičte posunutí na základě parametru tělesných proporcí  $\beta$  příspěvkem  $B_s(\beta)$ . Tak vznikne T-pose správných tělesných proporcí. Také vypočteme počáteční pozici kloubů  $J(\beta)$  na základě tělesných proporcí. Viz Obrázek 1.1.

Při pohybu se naše tkáně napínají a ohýbají a s tím musíme počítat i u počítačového modelu člověka. Dále tedy upravíme proporce těla tak, aby odpovídali po pozici do které chceme model dostat tím, že k modelu přidáme příspěvek  $B_p(\theta)$  a vznikne tak pozice  $T_p(\beta, \theta)$ . Viz Obrázek 1.2.

Nakonec se vše poskládá dohromady pomocí funkce  $W(\cdot)$  ještě s konečnou pozicí končetin  $\theta$  a maticí  $W$ . Tato matice spolu s  $B_s(\beta)$  a  $B_p(\theta)$  jsou natrénované hodnoty na tisících lidských skenů. Viz Obrázek 1.3.



Obrázek 1.1: SMPL – stage 1

(b)  $\bar{\mathbf{T}} + B_S(\vec{\beta}), J(\vec{\beta})$       (c)  $T_P(\vec{\beta}, \vec{\theta}) = \bar{\mathbf{T}} + B_S(\vec{\beta}) + B_P(\vec{\theta})$ (c)  $T_P(\vec{\beta}, \vec{\theta}) = \bar{\mathbf{T}} + B_S(\vec{\beta}) + B_P(\vec{\theta})$     (d)  $W(T_P(\vec{\beta}, \vec{\theta}), J(\vec{\beta}), \vec{\theta}, \mathcal{W})$

### 1.2.2 Výhody

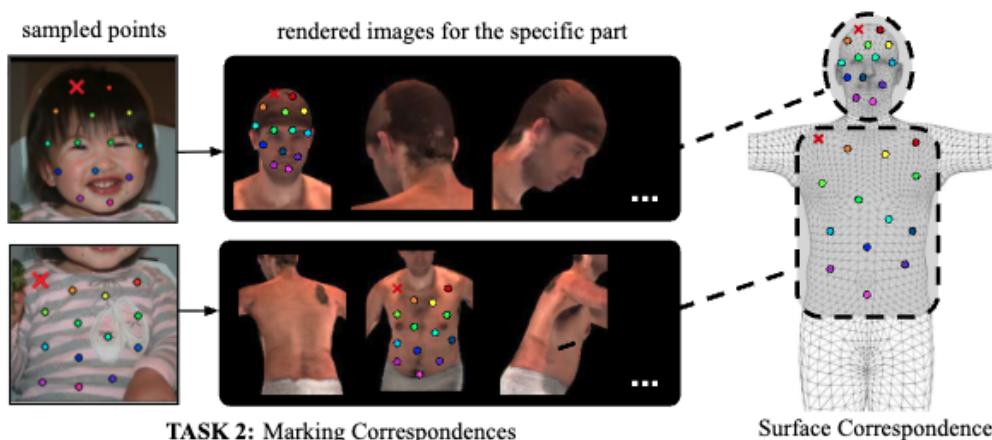
Tento model má výhodu v tom, že parametry  $\theta$  a  $\beta$  usměrní výslednek tak, aby bylo velmi těžké zdeformovat obrázek do nelidských proporcí. Zároveň však dokáže popsat celou řadu lidských těl. Je díky tomu vhodná pro

Tato knihovna je pro komerční účely zpoplatněná, ale pro vědecké účely je zdarma a stačí se registrovat na <https://smpl.is.tue.mpg.de/index.html>. Díky tomu se z ní stal standard v oblasti počítačového vidění.

## 1.3 DensePose

DensePose je metoda z oblasti počítačového vidění, která z obyčejného 2D obrázku odhaduje **hustou korespondenci** mezi pixely a povrchem lidského těla. Na rozdíl od klasického odhadu pózy pomocí klíčových bodů (např. lokty a kolena) se zde nepracuje s několika desítkami bodů, ale s tisíci pixelů, které lze přiřadit ke konkrétním místům na těle. Prakticky to znamená, že pro každý pixel patřící člověku dokáže DensePose určit, *kam na 3D povrch těla by tento pixel patřil*, pokud bychom měli k dispozici standardizovanou šablonu těla.

Jádrem celé myšlenky je kanonická reprezentace těla (tedy povrch „průměrného“ člověka v jednotných souřadnicích) a síť, která se učí předpovídat mapování z obrazu do této kanonické reprezentace. DensePose je typicky používán jako nadstavba detekce osob: nejprve se naleze ohrazení rameček (bounding box) postavy a teprve v jeho rámci se provede husté mapování pixelů na povrch těla. Díky tomu se výpočet soustředí na relevantní část obrazu a je možné pracovat s výstupem po jednotlivých lidech.



Obrázek 1.4: DensePose – DenseAnnoFigure, Zobrazuje korespondenci 2D pixelu s vybraným bodem na kanonické 3D síti lidského těla.

### 1.3.1 DensePose v1: mapování pomocí (I, U, V)

Původní verze DensePose (v této práci ji budu označovat jako DensePose v1) používá parametrisaci povrchu těla pomocí několika tělesných „chartů“ (segmentů). Výstupem je pro každý pixel na postavě trojice (I, U, V):

- **I:** index tělesného segmentu (např. levou paži, trup apod.),
- **U, V:** souřadnice v rámci 2D parametrizace daného segmentu.

Tento výstup je vhodný například pro přenos textur nebo pro vizualizaci mapování povrchu, ale pro následný fitting parametrického modelu (např. SMPL) je potřeba navíc řešit převod z (I, U, V) do konkrétních bodů nebo vrcholů (vertices) na síť ovině. To je možné, ale v praxi to přidává další vrstvu komplikací, protože optimalizační algoritmus pak nepracuje přímo s jednoznačnou korespondencí na SMPL mřížce.

### 1.3.2 DensePose v2: Continuous Surface Embeddings (CSE)

Novější varianta, kterou v této práci používám (DensePose v2), přechází od explicitních (I, U, V) souřadnic k tzv. Continuous Surface Embeddings (CSE). Místo toho, aby síť přímo vracela parametrické souřadnice na těle, vrací pro každý pixel vektor embeddingu (tedy bod v naučeném vektorovém prostoru), který reprezentuje odpovídající místo na povrchu těla. Současně vrací i hrubou segmentaci (coarse segmentation), která říká, které pixely v daném rámečku vůbec patří postavě.

Z praktického hlediska tak DensePose v2 produkuje dvě hlavní struktury:

- **Masku S** o rozměrech  $H \times W$  (binární nebo vícetřídní), která určuje pixely patřící tělu.
- **Matici vektorů E** o rozměrech  $H \times W \times D$ , kde  $D$  je dimenze embeddingu, a každý pixel tak nese svůj vektor  $e \in \mathbb{R}^D$ . DensePose byl natrénován tak, aby vektory korespondující s blízkými body byli podobné (skalární součin je maximální) a pro vzdálené body rozdílné (skalární součin je minimální).

Tyto mapy jsou definované v souřadnicích rámečku detekované osoby a lze je přepočítat na libovolné rozlišení (typicky interpolací na nové  $H$  a  $W$ ). V praxi je to důležité, protože výstup se dá škálovat podle toho, zda preferuji rychlosť (nižší rozlišení) nebo přesnost a počet korespondenčních bodů (vyšší rozlišení). Pro účely optimalizace je pak možné z embeddingů odvodit i jednoznačný index vrcholu na SMPL síť ovině: pro každý pixel se vybere ten vrchol, jehož předpočítaný embedding je vektorově nejbližší (typicky podle kosinové podobnosti). Výsledkem je tedy mřížka indexů, která mapuje pixely přímo na vrcholy SMPL (např. 6890 vrcholů), a tu lze přímo použít v loss funkci.

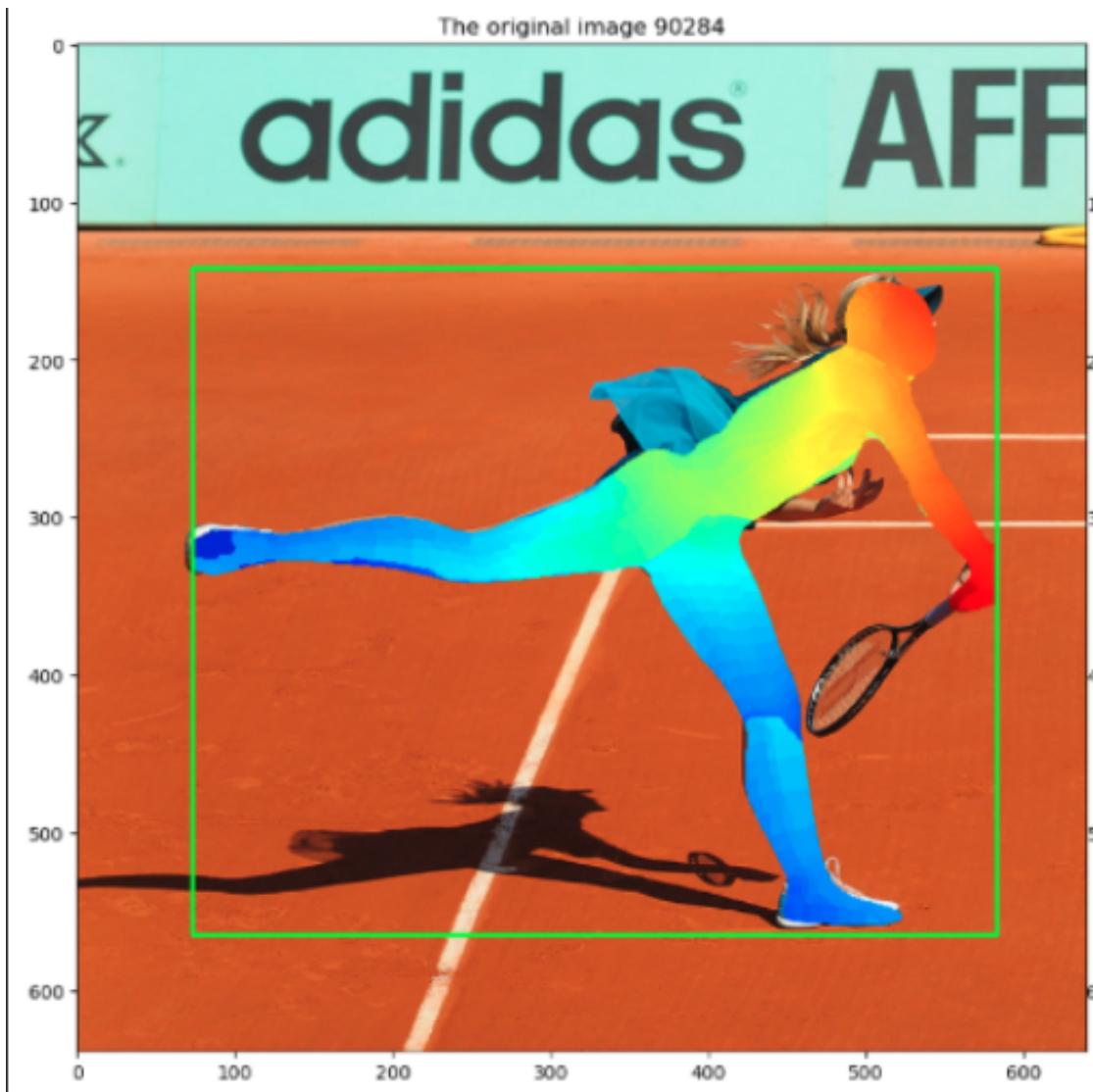
DensePose v2 jsem zvolil z několika důvodů. Především mě překvapila jeho přesnost i v podmínkách mimo laboratorní datasety a zároveň jde o open-source řešení postavené nad ekosystémem Detectron2, které je dobře integrovatelné do vlastního pipeline. Zároveň je férové říct, že DensePose jako projekt dnes působí spíše jako *vědci trochu zapomenutá technologie*: nové práce často preferují holistické modely, které rovnou odhadují parametry těla nebo celé 3D reprezentace. DensePose však zůstává velmi vhodný pro navazující práci právě kvůli tomu, jaký typ výstupu poskytuje. Moderní holistické modely mívají výstup „hotový“ (např. klouby nebo SMPL parametry), ale neposkytují jednoduchý způsob, jak si vyžádat hustou, škálovatelnou korespondenci pixelů na povrch těla, kterou lze přímo zapojit do vlastní optimalizace.

### 1.3.3 Silné a slabé stránky

Největší výhodou DensePose je, že poskytuje hustý signál: i když část těla není jasně vidět, zbytek povrchu často dává dostatek informací pro smysluplné omezení 3D řešení. Tento typ výstupu je navíc přirozeně kompatibilní s fittingem SMPL, protože umožňuje formulovat ztrátovou funkci nad velkým množstvím korespondenčních bodů.

Zároveň má DensePose několik slabin, které se v praxi projeví velmi rychle. Výstup bývá často šumový a místo hladkého povrchu připomíná „lupinky“ (lokálně nekonzistentní přiřazení sousedních pixelů). To vede k tomu, že i při vizuálně správné detekci může být lokální korespondence nekvalitní. Dalším typickým problémem je záměna symetrických částí těla: model může například označit obě nohy jako levé, případně prohodit levou a pravou stranu. V neposlední řadě se chyby objevují i na hranách segmentace (např. u volného oblečení), kde maska zahrne pixely, které ve skutečnosti neodpovídají povrchu těla.

Právě tyto slabiny jsou hlavním důvodem, proč jsem se rozhodl na DensePose navázat. Místo toho, abych DensePose bral jako „konečný“ výsledek, beru jej jako velmi bohatý, ale nedokonalý signál, který je potřeba dále zpracovat a zpřesnit tak, aby byl fitting SMPL stabilní a robustní i v hraničních případech.



Obrázek 1.5: DensePose – Showcase, Použili jsme barevnou mapu (colormap) Jet pro ilustraci. Obě nohy bledě modrou a tedy detekovány jako pravé. Útržek tmavě modré a tedy jiné končetiny je vidět vlevo a odporuje spojitosti těla. Vpravo na obrázku je zase vidět problém s "lupínky".



## 2. Explorativní část

V rámci své explorativní analýzy jsem navrhl 3 možná metodická zlepšení. Všechna z nich využívají metodu napasování (fitting) SMPL postavy.

### 2.1 Embedding fitter

#### 2.1.1 Návrh metody

Nejdříve jsem se rozhodl věnovat optimalizační úloze, kde zvolím nějakou chybu pro libovolně promítnutý SMPL model a minimalizací této chyby bych se měl dostat k modelu těla ve 3D, které co nejlépe koresponduje s detekcí člověka v obrázku. Tento nový odhad bych považoval za správnější, jelikož jeho lidská omezení zamezují standardním chybám DensePose.

DensePose v2 (CSE) mi pro každý pixel postavy nevrátí jen informaci „tady je člověk“, ale také odhad „kde na kanonickém těle tento pixel leží“. Konkrétně: DensePose pracuje v rámci ohraňujícího rámečku (bounding boxu) detekované osoby a vytvoří

- masku postavy  $S$  a
- embedding  $E$  pro každý pixel.

Maska je množina pixelů, které DensePose označí jako tělo. Pro každý pixel  $p \in S$  DensePose predikuje vektor  $e_p \in \mathbb{R}^D$  (embedding), kde  $D$  je pevně daná dimenze (typicky desítky čísel).

Na druhé straně úlohy nějak inicializují první pozici SMPL. Poté potřebuji vypočítat, jaký „bod na těle“ se v každém pixelu skutečně nachází. To není totéž jako vzít nejbližší promítnutý vrchol, protože projekce do 2D ignoruje zákryty: zadní část těla může být promítnuta do stejné oblasti obrazu jako přední část, ale ve skutečnosti ji kamera nevidí. Proto jsem uvažoval o renderování SMPL z pohledu kamery, ideálně tak, aby pro každý pixel bylo jasné, který trojúhelník síť je opravdu viditelný (tj. nejblíže kamere).

Výsledkem renderování pro pixel  $p$  je trojúhelník na SMPL síťovině, do kterého paprsek narazil. Trojúhelník popíše třemi indexy vrcholů ( $i, j, k$ ). Zároveň dostanu tzv. barycentrické váhy ( $\alpha, \beta, \gamma$ ), které říkají, jak přesně leží bod průsečíku uvnitř trojúhelníku (platí  $\alpha + \beta + \gamma = 1$ ). Díky tomu umím z vrcholových hodnot dopočítat hodnotu přímo v bodě průsečíku jako „vážený průměr“ vrcholů.

DensePose CSE má důležitou vlastnost: pro SMPL vrcholy existuje také předpočítaná tabulka embeddingů. Označím ji jako  $v_i \in \mathbb{R}^D$ , kde  $i$  je index vrcholu SMPL. Pro pixel  $p$  pak spočtu embedding bodu na povrchu SMPL, který je v daném pixelu vidět, jako

$$\tilde{e}_p = \alpha v_i + \beta v_j + \gamma v_k.$$

Tím mám dvě porovnatelné věci: DensePose embedding  $e_p$  (z obrázku) a „SMPL embedding“  $\tilde{e}_p$  (z vyrenderované síťoviny). Ztrátovou funkci (loss) pak chci definovat tak, aby byla malá, když oba embeddingy odpovídají stejnemu místu na těle. V praxi se embeddingy často normalizují na jednotkovou délku (L2-normalizace), a potom se podobnost měří kosinovou podobností, což je

skalární součin:

$$\langle \hat{\mathbf{a}}, \hat{\mathbf{b}} \rangle,$$

kde  $\hat{\mathbf{a}}$  je vektor  $\mathbf{a}$  po normalizaci (má délku 1). Skvělé na tom je, že hodnota je blízko 1, když vektory míří „stejným směrem“ (tedy jsou si podobné), a menší, když si podobné nejsou. Ztrátu tedy mohu napsat například jako

$$\mathcal{L}_{\text{CSE}} = \sum p \in S (1 - \langle \hat{\mathbf{e}}_p, \hat{\mathbf{e}}_p \rangle).$$

Význam symbolů: -  $S$  je množina pixelů, které DensePose označilo jako tělo. -  $p$  je jeden konkrétní pixel z této množiny. -  $\mathbf{e}_p$  je embedding z DensePose pro pixel  $p$ . -  $\hat{\mathbf{e}}_p$  je embedding, který jsem získal z aktuálního SMPL odhadu renderováním a interpolací. -  $\langle \cdot, \cdot \rangle$  je skalární součin (po normalizaci odpovídá kosinové podobnosti). - výraz  $(1 - \text{podobnost})$  je malý, když se embeddingy shodují. Optimalizace pak probíhá stejně jako u běžného fittingu SMPL: měním parametry SMPL tak, aby se minimalizovala ztráta. Prakticky jde o globální rotaci a posun těla vůči kameře, parametry pózy  $\theta$  (ohyby kloubů) a parametry tvaru  $\beta$  (tělesné proporce). Cíl je, aby při renderování SMPL vycházelo pro každý pixel „stejné místo na těle“, jaké predikuje DensePose.

Hlavní očekávaný přínos je v tom, že SMPL funguje jako humanoidní omezení: DensePose může lokálně „šumět“ (nespojitosti, přeskakování mezi částmi těla, záměna levé a pravé strany), ale SMPL vždy musí zůstat jedna konzistentní lidská síťovina s realistickými proporcemi. Fitting tak může DensePose využít jako hustý signál, ale zároveň chybou „vyhladit“ tím, že nejlepší řešení musí být globálně lidské. Vedlejším přínosem je, že výsledkem není jen sada 2D bodů, ale rovnou 3D model těla, ze kterého lze odvodit klouby i prostorovou pózu.

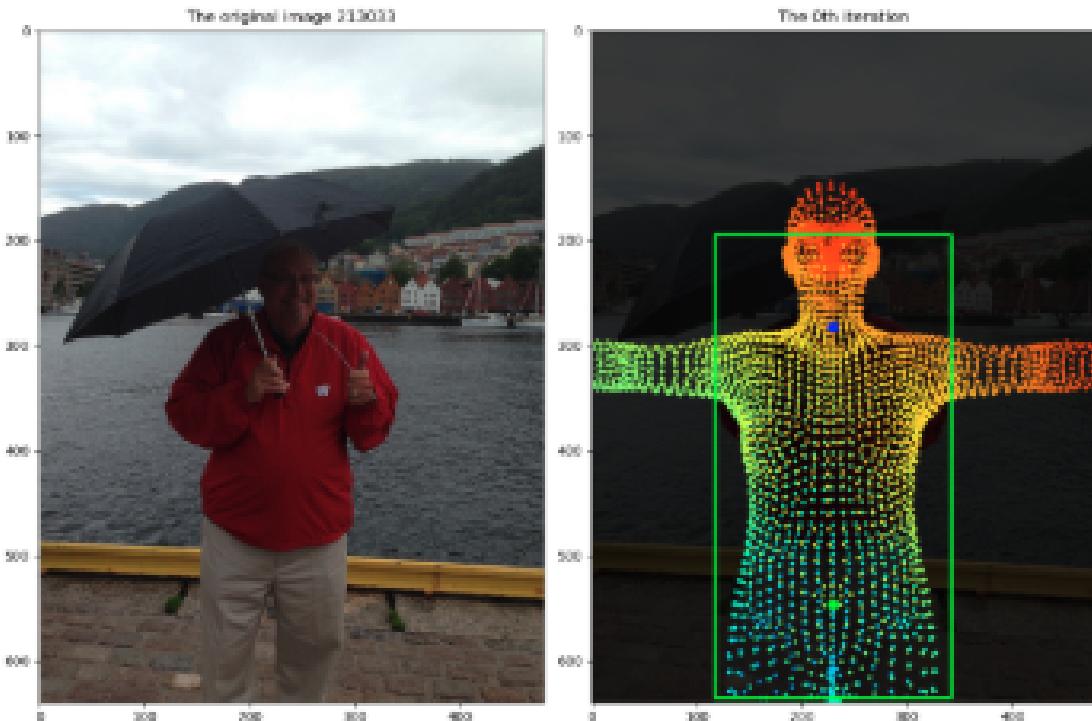
### 2.1.2 Implementace

V implementaci jsem se snažil co nejpříměji zrealizovat veličiny z návrhu metody: z DensePose beru masku pixelů těla  $S$  a embeddingy  $\mathbf{e}_p$ , ze SMPL v každé iteraci získám projekci do obrazu a pro stejné pixely určuji, který bod na síťovině je z pohledu kamery skutečně vidět. Teprve potom má smysl embeddingy porovnávat, protože bez ošetření zákrytů by do stejné oblasti obrazu „padaly“ i části těla, které kamera ve skutečnosti nevidí.

Nejdříve si připravím DensePose výstup pro jednu detekovanou osobu (bounding box, maska  $S$  a embedding pole). SMPL inicializuju pouze heuristicky v neutrální póze a k detekci jej přiblížím hrubým odhadem posunu pomocí `get_translation` v `common/utils.py`. Tato inicializace je čistě praktická: cílem není odhadnout přesnou pozici, ale dostat model do rozumné hloubky a měřítka, aby optimalizace nezačínala v degenerovaném stavu. Pro to, aby moje metoda začala fungovat mi stačí rozumný překryv těl a dále by se měl program optimalizovat sám.

Pro projekci používám matici intrinsics  $K$  sestavenou funkcí `get_camera_intrinsics` (také v `common/utils.py`). V kódu je použita varianta `K_flipped`, která odpovídá obrazovým souřadnicím (osa  $y$  roste směrem dolů), takže se vyhnou časté chybě, kdy se render „zrcadlí“ přes vodorovnou osu. Zároveň jsem musel explicitně ošetřit body za kamerou: čistě matematicky projekce vrátí 2D souřadnice i pro body s  $z \leq 0$ , ale tyto body jsou fyzikálně neviditelné a při optimalizaci působí jako falešné korespondence. Proto v části, která počítá viditelnost, filtruji průsečíky pod minimální hloubkou (v implementaci je na to parametr `min_depth`).

Viditelnost jsem realizoval ray tracingem ve funkci `visible_vertices_gpu`. Prakticky to znamená: pro každý dotazovaný pixel hledám trojúhelník sítě, jehož projekce pixel obsahuje, a z kandidátů



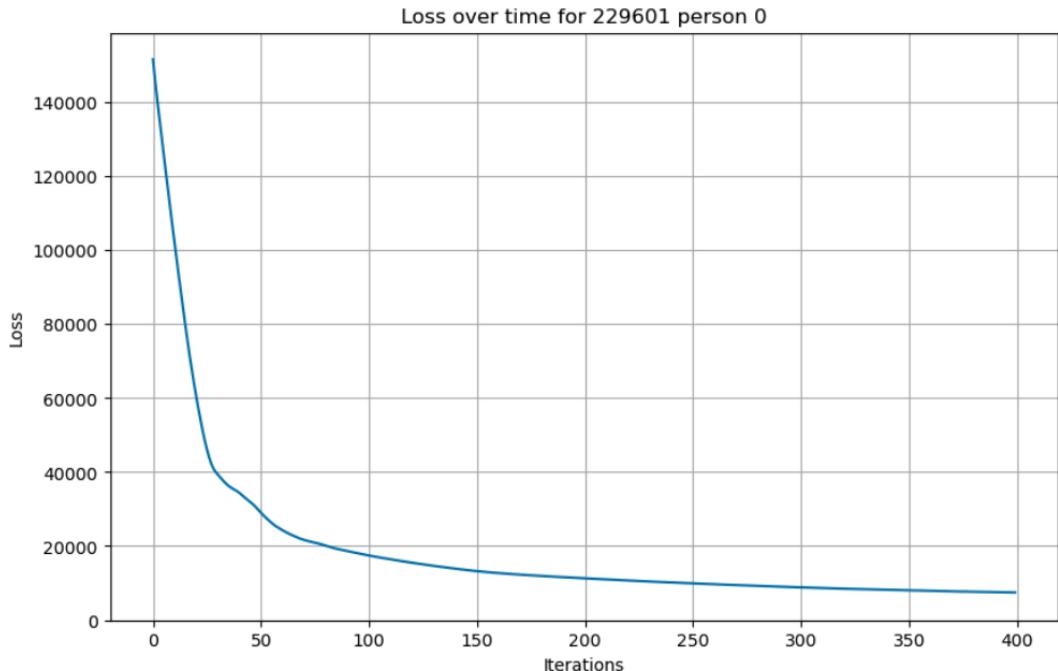
Obrázek 2.1: Příklad inicializace

vyberu ten s nejmenší hloubkou (nejbližší kameře). Původně jsem chtěl využít barycentrické váhy a embedding interpolovat uvnitř trojúhelníku tak, jak je to v návrhu metody. V prototypu jsem ale nakonec zvolil jednodušší kompromis: z viditelného trojúhelníku vybírám jeden reprezentativní vrchol (nejbližší bodu průsečíku) a embedding beru přímo z jeho předpočítané tabulky. Tím se ztrácí část „plynulosti“ vůči parametrům, ale implementace je výrazně jednodušší a paměťově méně náročná.

Výpočet viditelnosti jsem nejdříve napsal na CPU v NumPy, ale rychle se ukázalo, že je to pro reálné rozlišení příliš pomalé. Přepsání do PyTorch mi umožnilo využít GPU a hlavně vektorizovat celé bloky výpočtu: místo smyček nad pixely se pracuje s tenzory a maskami (typicky tabulka dotazovaných pixelů vůči kandidátním trojúhelníkům), což je v `common/utils.py` vidět i podle značení tvarů v komentářích. Parametry SMPL pak optimalizují gradientně pomocí Adam, stejně jako v `projects/euclidean_fitter.py`. Optimalizované proměnné jsou globální orientace, posun, póza  $\theta$  a tvar  $\beta$ ; v každé iteraci přepočtu síťovinu, projekci, viditelnost a následně minimalizují ztrátu založenou na podobnosti embeddingů na množině pixelů  $p \in S$ . Počet iterací jsem určil experimentálně na 400. Při větším počtu iterací již nepozorují žádnou změnu (viz 2.2) hodnoty ztrátové funkce a zbytečně bych prodlužoval čas čekání na výstup.

### 2.1.3 Hodnocení

Metoda mi na papíře dávala smysl, protože kombinuje dva silné signály: DensePose poskytuje husté korespondence v obraze a SMPL vynucuje globálně konzistentní lidskou geometrii. V praxi se ale ukázalo, že právě část „pro každý pixel spočti správný bod na povrchu“ je výpočetně i numericky citlivá.



Obrázek 2.2: Závislost ztráty na iteraci pro daný případ.

Prvním limitem byla náročnost. Ray tracing nad trojúhelníkovou sítí pro velké množství pixelů se rychle stává úzkým hrdlem a i po přepsání do GPU verze bylo ladění pomalé (zejména pokud jsem chtěl testovat více obrázků nebo vyšší rozlišení masky). Druhý, důležitější problém byla stabilita optimalizace. Opakovaně se mi stávalo, že SMPL vycházel posunutý vůči detekci, případně se optimalizátor „chytil“ špatného lokálního minima a pôza se začala hroutit do nelidských tvarů. V této fázi jsem nedokázal jednoznačně izolovat jedinou příčinu, ale prakticky se ukázalo, že metoda je velmi citlivá na inicializaci (hlavně posun a znaménko/škálování hloubky) a na to, jak přesně se ošetří projekce bodů za kamerou. Navíc je zde nepříjemná vlastnost samotné viditelnosti: i malé změny parametrů mohou skokově změnit, který trojúhelník je „první průsečík“, a tím se zhoršuje chování gradientní optimalizace. Z hlediska cíle maturitní práce pro mě bylo klíčové mít metodu, se kterou lze rychle iterovat a která dává interpretovatelné výsledky i na hraničních případech.

Proto jsem tuto variantu vyhodnotil jako slepu uličku a přešel jsem k jednoduššímu měření ztráty v obraze pomocí euklidovské vzdálenosti, které sice obětuje část původní elegance, ale výrazně zlepšuje stabilitu experimentů.

Následující metoda **Euklidovský fitter** vznikla jako zjednodušená varianta předchozího prototypu. Kompletní implementace je v souboru `projects/euclidean_fitter.py`; v `common/utils.py` pak zůstaly pomocné funkce pro práci s projekcí a výpočtem ztráty.

## 2.2 Euklidovský fitter

### 2.2.1 Návrh metody

V předchozí metodě jsem porovnával DensePose embeddingy se „stejnými“ embeddingy na povrchu SMPL, což si ale vyžádalo explicitní řešení viditelnosti (ray tracing) a vedlo to k nestabilní

optimalizaci. Zde proto volím jednodušší pohled: DensePose použije pouze k vytvoření 2D korespondencí mezi pixely a vrcholy SMPL a samotnou ztrátu budu měřit přímo v obrazové rovině euklidovskou vzdáleností.

DensePose v2 (CSE) mi pro každý pixel postavy dává embedding  $\mathbf{e}_p$  a zároveň masku  $S$  (pixely, které patří tělu). Protože pro SMPL vrcholy existují předpočítané embeddingy  $\mathbf{v}_i$ , mohu pro každý pixel  $p \in S$  vybrat nejbližší vrchol (podle kosinové podobnosti, tj. skalárního součinu normalizovaných embeddingů)

$$I(p) = \arg \max \langle \hat{\mathbf{e}}_p, \hat{\mathbf{v}}_i \rangle,$$

kde  $I(p)$  je index vrcholu SMPL, který DensePose pixelu  $p$  přiřazuje. Tím se z DensePose výstupu stane množina 2D bodů s identitou vrcholu: pro každý pixel známe jeho souřadnice  $\mathbf{u}_p \in \mathbb{R}^2$  (v souřadnicích bounding boxu) a příslušný index  $I(p)$ .

Na druhé straně mám v každé iteraci optimalizace aktuální SMPL síť ovinu. Pro vybranou množinu vrcholů  $V$  jejich 3D souřadnice promítu do obrazu a dostanu 2D body  $\mathbf{x}_i \in \mathbb{R}^2$ .

Tady pracuji s proběmem toho, k jakému pixelu mám vzdálenost od SMPL vrcholu počítat. Může se totíž stát, že takový pixel bude

- právě jeden,
- bude jich mnoho, nebo
- nebude viditelný žádný

Ztrátovou funkci pak definuji jako průměrnou vzdálenost mezi projekcí vrcholu a nejbližším DensePose pixelem.

$$\mathcal{L}_{EUC} = \frac{1}{|V^*|} \sum_{i \in V^*} \min_{p \in S: I(p)=i} \|\mathbf{x}_i - \mathbf{u}_p\|_2,$$

kde  $V^* = \{i \in V \mid \exists p \in S : I(p) = i\}$  je množina vrcholů, které se ve výstupu DensePose vůbec objeví. Pro takové vrcholy měří euklidovská vzdálenost  $\|\mathbf{x}_i - \mathbf{u}_p\|_2$  v pixelech, jak „daleko“ je SMPL projekce od DensePose korespondence.

Metoda se tedy shoduje s předchozím přístupem v tom, že stále využívá DensePose jako hustý signál a SMPL jako globální humanoidní omezení. Liší se ale tím, že se vyhýbá výpočtu viditelnosti na síť ovině a porovnávání embeddingů; místo toho optimalizuje přímo jednoduchou geometrickou chybu v obraze. Očekával jsem proto stabilnější chování gradientní optimalizace a výrazně rychlejší iterování experimentů.

### 2.2.2 Implementace

Implementace v `projects/euclidean_fitter.py` zachovává stejnou kostru jako předchozí metoda: z detekce DensePose vezmu bounding box, masku  $S$  a embeddingy, SMPL inicializuju v neutrální pozě a optimalizuju proměnné `global_orient`, `transl`, `pose` a `betas` pomocí Adam. Zásadní rozdíl je pouze v tom, jakým způsobem z těchto dat vytvořím ztrátu.

Nejdříve pro každou detekovanou osobu převedu DensePose embeddingy na mapu indexů vrcholů  $I(p)$ . To zajišťuje funkce `get_closest_vertices_mask_from_ES`, která pro každý pixel v masce

vybere vrchol s nejvyšší podobností embeddingu. V kódu tuto mapu počítám ve dvou rozlišeních: v původním rozlišení bounding boxu pro vizualizaci a v menším rozlišení pro samotný výpočet ztráty. Rozlišení pro loss volím tak, aby mělo přibližně konstantní plochu (proměnná LOSS\_AREA); tím udržuji počet bodů v  $S$  a výpočetní náročnost přibližně stejnou napříč různě velkými detekcemi.

Z mapy  $I(p)$  si pak připravím dvě pole:

- **E\_indices** – seznam vrcholových indexů  $I(p)$  pro všechny pixely  $p \in S$ ,
- **E\_coordinates** – odpovídající 2D souřadnice  $\mathbf{u}_p$  v souřadnicích zmenšeného bounding boxu.

Volitelně mohu omezit optimalizaci pouze na torso (TORSO\_MASK), čímž snížím vliv typicky nejhůře predikovaných končetin a zároveň zmenším množinu vrcholů, se kterou loss pracuje.

V každé iteraci vyrenderuji (tj. spočtu) aktuální SMPL vrcholy, promítu je do obrazu pomocí intrinsics matice  $K$  z `get_camera_intrinsics` a převedu je do stejné souřadnicové soustavy jako DensePose body: odečtu levý horní roh bounding boxu a aplikuj stejné škálování. Výsledkem je pole `SMPL_coordinates` s body  $\mathbf{x}_i$ .

Samotnou ztrátu počítá funkce `euclid_loss_gpu` v `common/utils.py`. Naivní CPU varianta by pro každý vrchol  $i$  musela prohledat všechny DensePose pixely a najít ty, které mají  $I(p) = i$ , což vede k pomalým vnořeným smyčkám. GPU verzi jsem proto napsal v PyTorch tak, aby byla plně vektorizovaná: vytvořím tabulku shod indexů (masku tvaru  $E \times S$ ), pomocí broadcastingu spočtu euklidovské vzdálenosti pouze pro shodné dvojice a pro každý vrchol vezmu minimum. Tato část už je analogická postupu z předchozí sekce, kde jsem kvůli výkonu také přecházel od smyček k tenzorovým operacím na GPU.

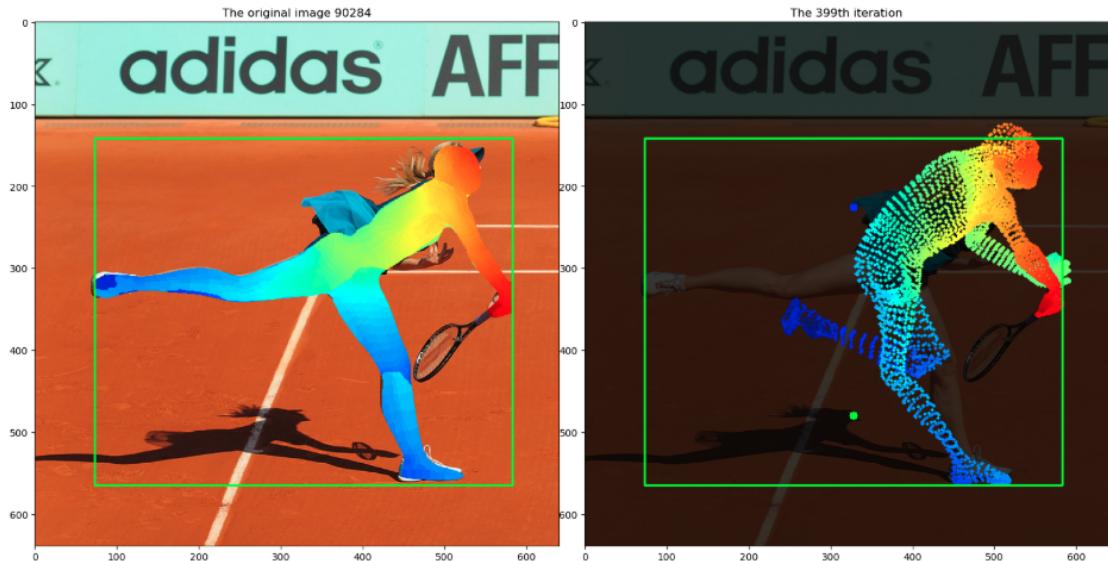
### 2.2.3 Hodnocení a výsledky

Z hlediska ladění se tato varianta ukázala jako výrazně příjemnější než předchozí metoda s ray tracingem: ztráta je definovaná čistě v obrazové rovině a výpočet neobsahuje skokové změny viditelnosti trojúhelníků. To ale neznamená, že by metoda automaticky opravovala všechny chyby DensePose.

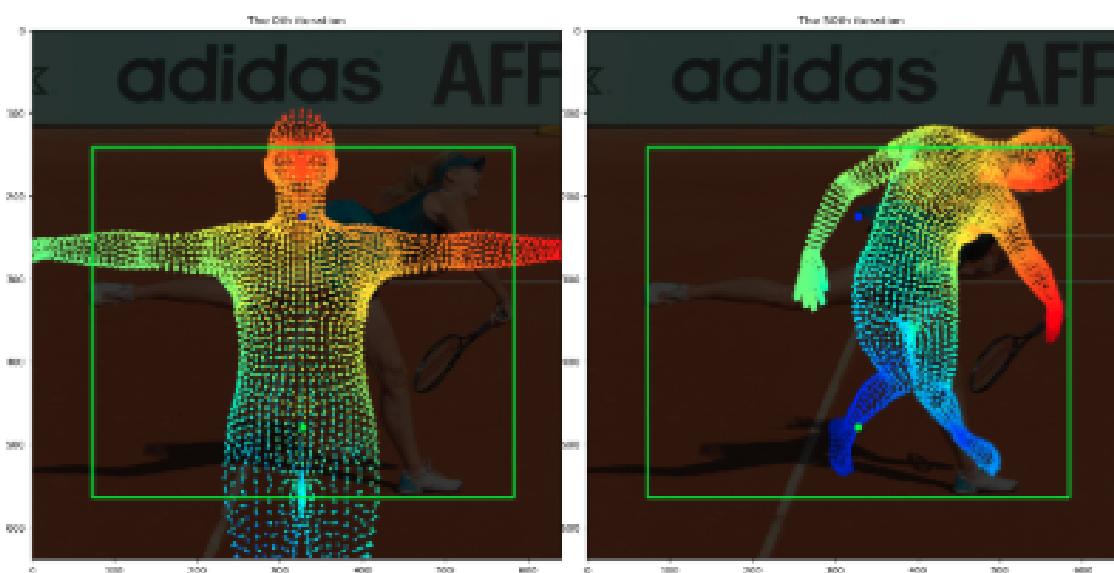
V případech, kde se DensePose výrazně mýlí (například prohodí končetiny), je pro optimalizaci těžké najít smysluplné řešení, protože korespondence  $I(p)$  je už na vstupu nekonzistentní. Typický příklad je na Obrázku 2.3, kde špatně označené části těla vedou SMPL do řešení, které sice lokálně snižuje ztrátu, ale neodpovídá skutečné póze.

Další praktický problém je inicializace. U složitějších scén se optimalizátor často „chytí“ tvarových parametrů  $\beta$  nebo lokálních kloubů dříve, než se podaří správně nastavit globální orientaci a posun. To může vést k deformacím, ze kterých se už optimalizace nevrátí. Typický se to stává u lidí stojících zády ke kameře (model se musí nejprve celý otočit) nebo u ležících postav. Obrázek 2.4 ukazuje případ, kdy se při použité inicializaci začne tělo deformovat dříve, než se stihne správně zorientovat.

Na druhou stranu se ve velké části případů podaří ztrátu výrazně snížit: pro všechny testované obrázky optimalizace našla konfiguraci SMPL s chybou pouze v jednotkách promile vůči původní heuristické inicializaci (viz Obrázek 2.5). To naznačuje, že zvolená loss funkce je „optimalizovatelná“ a umí SMPL do DensePose dat dotáhnout, pokud se algoritmus nezamotá v prvních iteracích. Zlepšená inicializace (případně silnější regularizace a omezení tvarových parametrů) by proto měla

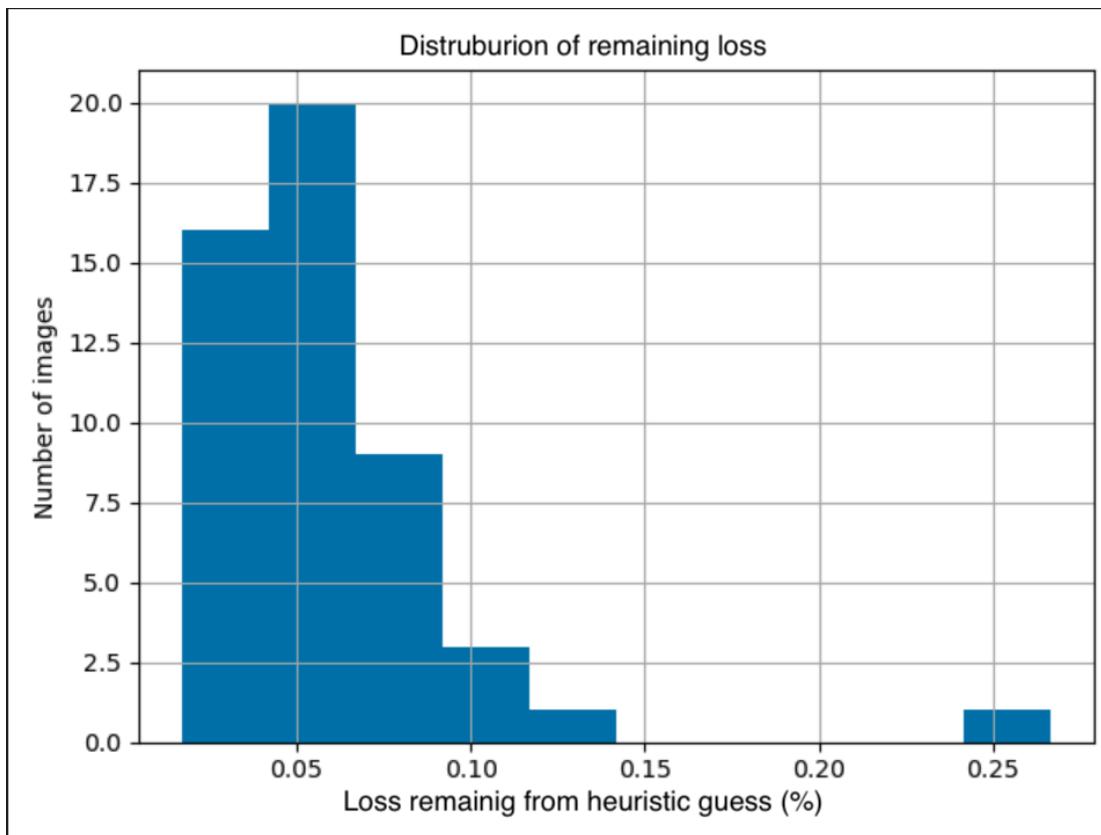


Obrázek 2.3: Výstup DensePose (vlevo) a výstup metody po předposlední (399.) iteraci (vpravo).



Obrázek 2.4: Inicializace (vlevo) a výstup metody po 50. iteraci (vpravo).

zvýšit počet úspěšných případů.

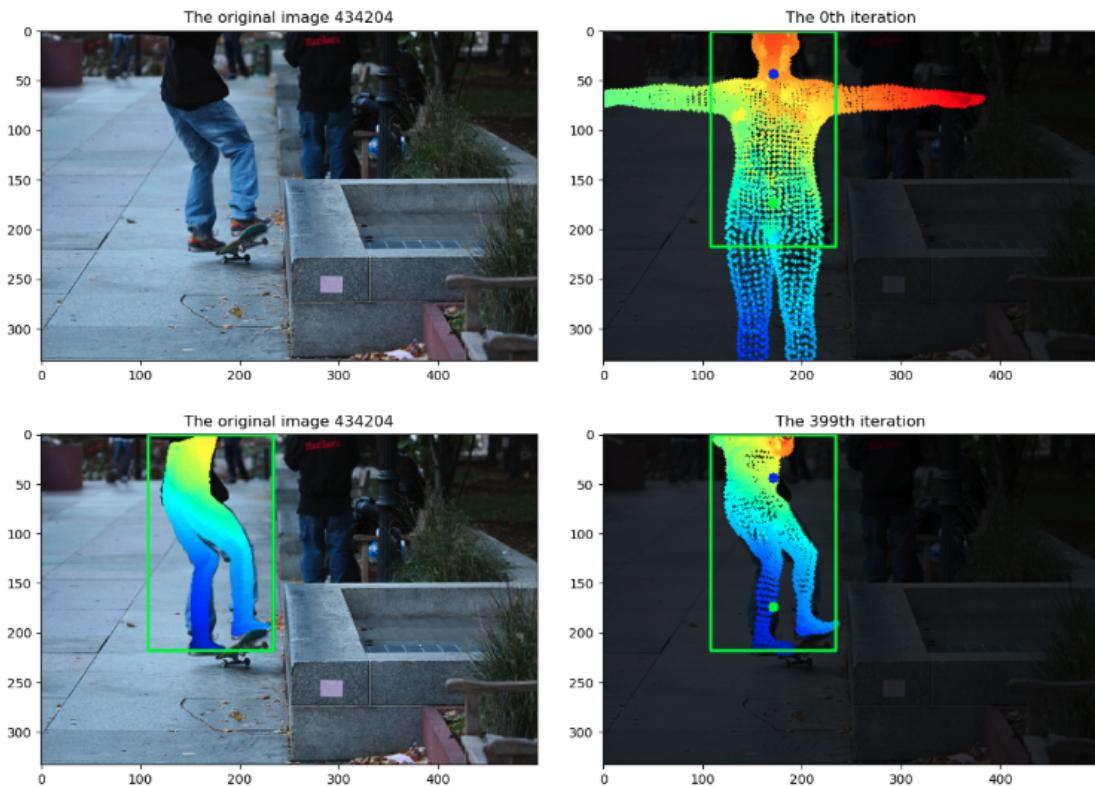


Obrázek 2.5: Počet obrázků podle zbývající chyby z původního heuristického odhadu v procentech.

I za zkoušených podmínek se přibližně u třetiny datasetu podařilo dostat 2D překryv na úroveň původních dat DensePose, s výhodou navíc v podobě explicitního 3D objemu. Zároveň se objevily příklady, kde fitting vede přímo ke zlepšení tvaru a proporcí oproti surovému DensePose výstupu (Obrázek 2.6).

Celkově tato metoda splnila hlavní cíl, který jsem od ní očekával: oproti předchozí variantě je výrazně jednodušší, stabilnější na implementaci a umožňuje rychleji provádět experimenty. Její limity jsou ale zřetelné — kvalita výsledku je stále silně závislá na kvalitě DensePose korespondencí a na inicializaci globální transformace. Pro další práci proto dává smysl soustředit se právě na tyto dvě části (lepší inicializace a robustnější práce s chybami korespondencemi), protože samotná ztráta založená na euklidovské vzdálenosti se v praxi chová předvídatelně a je dobře optimalizovatelná.

\*\*\*\*\* musim napsat o případech kde je jeden nebo zadny nebo mnoho body a PROC jsem vybral ten nejblízsi



Obrázek 2.6: Příklad vylepšení lidských proporcí

## 2.3 Zaměření na přesnost

Toto je přístup pokoušející se řešit první nedostatek předchozí metody a tedy případy, kde se DensePose mylil natolik, že například označil obě nohy jako levé a náš program "neví" kam má napasovat nohu SMPL modelu. Vychází z hypotézy, že body s indexem  $i$ , které budou mít pro densepose předpověď po vyrendrování do mřížky nejmenší rozptyl jsou nejpřesnější. Např. na Obrázku 2.3 by obě

### 2.3.1 Návrh metody

### 2.3.2 Implementace

### 2.3.3 Výsledky



# 3. Technická dokumentace

Poslední kapitola obsahuje informace o tom, jak projekt, který v rámci maturitní práce vznikl, nainstalovat, spustit a používat.

## 3.1 Struktura repozitáře

```
simonpose/
└── common/
└── docs/
└── documentation/
└── projects/
└── visualisations/
```

Obrázek 3.1: Hlavní struktura repozitáře

Cesty jsou v souboru config.py. Ty se dostavají do repozitáře pomocí knihovny ...

## 3.2 Instalace

## 3.3 Spuštění



# Závěr

Závěr obsahuje shrnutí práce a vyjadřuje se k míře splnění jejího zadání. Dále by se zde mělo objevit sebehodnocení studenta a informace o tom, co nového se naučil a jak vnímal svou práci na projektu.



# Seznam použité literatury

- [MM16] Dmytro Mishkin a Jiří Matas. *All you need is a good init.* 2016. doi: 10.48550/arXiv.1511.06422. arXiv: 1511.06422 [cs.LG]. URL: <https://arxiv.org/pdf/1511.06422.pdf> (cit. 27.02.2026).
- [Mor25] Šárka Morávková. *Title of the document.* 2025. URL: <https://drive.google.com/file/d/17Mhfoe3TpJCVXy5ry61DvdaYFdgLFuhM/view> (cit. 27.02.2026).



# Seznam obrázků

1.1	SMPL – stage 1 . . . . .	9
1.2	SMPL – stage 2 . . . . .	9
1.3	SMPL – stage 3 . . . . .	9
1.4	DensePose – DenseAnnoFigue, Zobrazuje korespondenci 2D pixelu s vybraným bodem na kanonické 3D síti lidského těla. . . . .	10
1.5	DensePose – Showcase, Použili jsme barevnou mapu (colormap) Jet pro ilustraci. Obě nohy bledě modrou a tedy detekovány jako pravé. Útržek tmavě modré a tedy jiné končetiny je vidět vlevo a odporuje spojitosti těla. Vpravo na obrázku je zase vidět problém s "lupínky". . . . .	13
2.1	Příklad inicializace . . . . .	17
2.2	Závislost ztráty na iteraci pro daný případ. . . . .	18
2.3	Výstup DensePose (vlevo) a výstup metody po předposlední (399.) iteraci (vpravo). . . . .	21
2.4	Inicializace (vlevo) a výstup metody po 50. iteraci (vpravo). . . . .	21
2.5	Počet obrázků podle zbývající chyby z původního heuristického odhadu v procentech. .	22
2.6	Příklad vylepšení lidských proporcí . . . . .	23
3.1	Hlavní struktura repozitáře . . . . .	25

# **Seznam tabulek**