VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

ALGORITMUS POROVNÁVÁNÍ HLAV OSOB V NE-STANDARDNÍCH POHLEDECH

ALGORITHM FOR HEAD COMPARISON IN NON-STANDARD VIEWS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JAROMÍR WYSOGLAD**
AUTHOR

VEDOUCÍ PRÁCE **Prof. Ing., Dipl.-Ing. MARTIN DRAHANSKÝ, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Wysoglad Jaromír**
Program: Informační technologie
Název: **Algoritmus porovnávání hlav osob v nestandardních pohledech**
Algorithm for Head Comparison in Non-Standard Views
Kategorie: Umělá inteligence

Zadání:

1. Prostudujte literaturu týkající se detekce a rozpoznávání obličeje a obecného porovnávání 2D objektů ve scéně, zaměřte se na řešení bez použití neuronových sítí.
2. Navrhněte algoritmický postup porovnání snímků hlavy osob ve stejných situačních podmínkách (natočení, sklon a náklon) - tedy porovnání dvou objektů zájmu.
3. Výše navržený algoritmus implementujte a otestujte na dostupných datech.
4. Shrňte dosažené výsledky a diskutujte možná rozšíření.

Literatura:

- LIAO, Zhouyingcheng, et al. Uniface: A Unified Network for Face Detection and Recognition. In: *2018 24th International Conference on Pattern Recognition (ICPR)*. IEEE, 2018. p. 3531-3536.
- YANG, Chengzhuan; WEI, Hui; YU, Qian. A novel method for 2D nonrigid partial shape matching. *Neurocomputing*, 2018, 275: 1160-1176.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2.

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

Vedoucí práce: **Drahanský Martin, prof. Ing., Dipl.-Ing., Ph.D.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 31. července 2020
Datum schválení: 31. října 2019

Abstrakt

Cílem této práce je vytvořit algoritmus pro porovnávání lidských hlav na obrázcích. Algoritmus je schopen porovnávat hlavy v mnoha různých pozicích, avšak pro porovnání musí vždy hlavy být ve stejné pozici. Algoritmus nejprve využije několik volně dostupných detektorů pro detekci hlavy a jejích částí. Následně je pro každou část hlavy vypočítán histogram orientovaných gradientů a jejich porovnáním algoritmus zjistí odlišnost hlav. Z testovací sady 30 obrázků se 26 z nich úspěšně podařilo detekovat. Při porovnávání se obrázek každé osoby porovnával s 5 jinými obrázky, z nichž jeden byl obrázek stejné osoby. Pokud nepočítám 4 obrázky, kde detekce nebyla úspěšná a 2 obrázky, které měly k těmto obrázkům být přiřazeny, tak algoritmus u 18 obrázků úspěšně určil jako nejpodobnější obrázek stejné osoby, u 5 obrázků byl tento obrázek 3. nejpodobnější a u jednoho obrázku zcela selhal a obrázek stejné osoby označil za nejodlišnější. Algoritmus úspěšně dokáže porovnat hlavy osob v různých polohách u většiny fotografií.

Abstract

The goal of this work is to create an algorithm for human head comparison. The algorithm is able to compare heads in a lot of different positions, but the heads, that are being compared, must be in the same position. At first the algorithm uses some freely available detectors for detecting heads and head parts. Then a histogram of oriented gradients is computed for each part of each head and by comparing them the algorithm finds out the dissimilarity of the heads. From the testing set of 30 pictures the algorithm is able to successfully detect heads on 26 pictures. Every picture was compared with 5 other pictures, with one of them containing a head of the same person. If I don't count the 4 pictures, where the algorithm wasn't able to detect the head and 2 pictures, which should have been assigned to these pictures. The algorithm successfully determined, that the head of the same person is the most similar on 18 pictures. On 5 pictures the head of the same person was determined as the 3rd most similar and on one of the pictures the algorithm failed completely and determined the head of the same person to be the least similar. The algorithm is successful with head comparison in different positions on most of the pictures.

Klíčová slova

Počítačové vidění, detekce tváří, detekce hlav, porovnávání tváří, porovnávání hlav

Keywords

Computer vision, face detection, head detection, face comparison, head comparison

Citace

WYSOGLAD, Jaromír. *Algoritmus porovnávání hlav osob v nestandardních pohledech.* Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Prof. Ing., Dipl.-Ing. Martin Drahanský, Ph.D.

Algoritmus porovnávání hlav osob v nestandardních pohledech

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Prof.Ing., Dipl.-Ing. Martina Drahanského Ph.D. Další informace mi poskytli Ing. Tomáš Goldmann Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Jaromír Wysoglad
31. července 2020

Poděkování

Děkuji panu Prof.Ing., Dipl.-Ing. Martinu Drahanskému Ph.D. za vedení práce, panu Ing. Tomáši Goldmannovi za některé rady týkající se počítačového vidění a mé rodině za umožnění nafocení jejich hlav pro účely testování projektu.

Obsah

Kapitola 1

Úvod

Oblast počítačového vidění je v poslední době velice aktivní oblastí výzkumu v informačních technologiích. Velice významnou částí tohoto výzkumu je detekce a porovnávání lidských tváří. Tato oblast je důležitá hned z několika důvodů. Sken tváře může sloužit k identifikaci člověka a tím pádem může být použit ke snadnější a rychlejší autentizaci, než kdyby mělo být vyžadováno zadání hesla. Dalším využitím může být vyhledávání osob. Při spáchání trestného činu se může povést získat fotografii pachatele a ten může být následně nalezen pomocí automatické analýzy záznamů z bezpečnostních kamer.

Většina nynějších algoritmů pro identifikaci osob však ke správnému fungování potřebuje obličej snímané osoby. Mnohdy je potřeba, aby byl tento obličej natočen přímo na kameru a aby kvalita snímku byla značně vysoká. Tohle je bohužel velice nepravděpodobné pro již zmiňovaného pachatele zločinu snímaného bezpečnostní kamerou. Je tedy potřeba identifikaci založit na snímku hlavy s pouhou částí obličeje na rozdíl od obličeje celého.

Na začátku práce (kapitola 2) se čtenář seznámí s různými současnými metodami pro detekci a porovnání tváří a různých jiných tvarů. Tato sekce je zaměřena hlavně na algoritmy, které nevyužívají neuronové sítě z důvodu časové náročnosti jejich tréninku. Práce dále pokračuje (kapitola 3) návrhem řešení a stanovením nejhorších, ale stále ještě přijatelných výsledků práce. V kapitole 4 se čtenář dozví veškeré detaily související s implementací práce, včetně použitých jazyků a knihoven. Nakonec v kapitole 5 budou představeny experimenty a jejich výsledky včetně vysvětlení některých nečekaných, nebo zajímavých výsledků.

Kapitola 2

Současný stav

V současnosti probíhá rozsáhlý výzkum v oblasti počítačového vidění. Významnou částí tohoto výzkumu je detekce a porovnání různých tvarů. Metody pro práci s tvary jsou pak dále využívány a specializovány pro detekci a porovnávání obličejů. To pak počítači umožňuje provádět identifikaci lidí na základě fotografie jejich tváře. Na poli detekce a porovnávání celých hlav v různém natočení (ne pouze tváří) však zatím moc výzkumu neproběhlo.

Tato kapitola se zabývá přehledem některých používaných metod pro popis, detekci a porovnání různých tvarů a obličejů.

2.1 Detekce a porovnávání tvarů

Tato sekce nabízí přehled některých algoritmů používaných pro detekci a porovnávání tvarů. Mnohé tyto algoritmy lze využít i k porovnání obličejů (sekce 2.2).

Algoritmy pro porovnávání tvarů jsou pro tuto práci důležité, protože na rozdíl od algoritmů pro porovnávání tváří umožňují porovnávat různé tvary (ne pouze tváře).

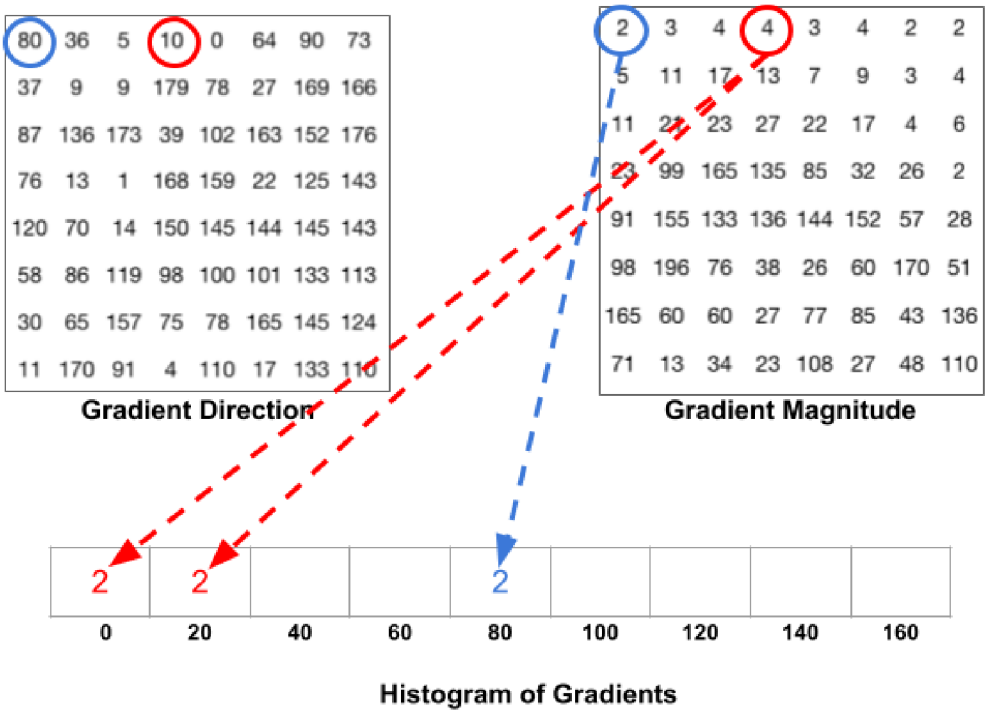
2.1.1 Histogram of Oriented Gradients

HoG (*Histogram of Oriented Gradients* - Histogram orientovaných gradientů) [4] je deskriptor obrázku založený na sledování intenzity a směru gradientů.

Pro výpočet HoG deskriptoru je obrázek nejprve profiltrován filtry pro hledání gradientů. Poté se pro každý pixel pomocí následující rovnice zjistí velikost a směr gradientu.

$$\begin{cases} g = \sqrt{g_x^2 + g_y^2} \\ \theta = arctan\frac{g_y}{g_x} \end{cases} \tag{2.1}$$

Kde g_x je výsledek filtrování filtrem pro hledání vertikálních gradientů, g_y je výsledek filtrování filtrem pro hledání horizontálních gradientů, g je velikost gradientu a θ je směr gradientu. Následně se obrázek rozdělí na několik stejně velkých dílů a pro každý díl obrázku se vytvoří histogram. Histogram se tvoří pomocí zápisu velikostí gradientů v každém pixelu dílku do vektoru o velikosti 9. Vektor je rozdělen rovnoměrně pro všechny směry gradientu a pozice zápisu do vektoru závisí na směru gradientu v daném pixelu. Zápis do vektoru je ilustrován na obrázku 2.1 Nakonec jsou vektory normalizovány a konkatenovány do jednoho



Obrázek 2.1: **Výpočet histogramu z údajů o velikosti a směru gradientů** Modrou barvou je do vektoru histogramu přidáván gradient se směrem přesně 80°; Červenou barvou je vyznačeno přidání gradientu se směrem 10°, což je mezi hodnotami směru 0°a 20°a hodnota je tedy rovnoměrně rozdělena mezi tyto dva směry. [2]

vektoru, který popisuje celý obrázek. Tento vektor lze následně porovnávat s vektory z jiných obrázků a tím mezi obrázky hledat podobnosti.

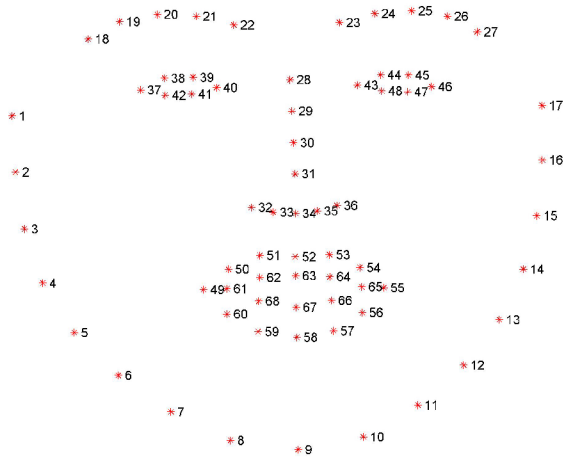
2.1.2 Active shape models

ASM (*Active Shape Models* - Aktivní modely tvarů) [3] je model, který se v obrázku snaží nalézt důležité body tvaru (příklad těchto bodů pro lidský obličej lze vidět na obrázku 2.2).

Pro učení modelu jsou potřeba obrázky tvaru s ručně vyznačenými lokacemi těchto důležitých bodů. Při detekci tvaru v novém obrázku se nejprve pro každý bod spočítá jeho pravděpodobná lokace (uprostřed mezi pozicemi tohoto bodu ve všech naučených variantách tvaru). Poté se nalezne nejbližší hrana v obrázku a model se upraví tak, aby se každý z bodů přiblížil nalezené hraně. V momentě kdy model odpovídá obrázku a není potřeba jej dále upravovat je tvar nalezen a jsou známy všechny jeho důležité body.

Pro ASM existuje mnoho rozšíření, které se pokoušejí o lepší popis okolí bodů. Například Combined-ASM [20], který pro popis některých bodů využívá deskriptor SIFT [11], nebo EP-LBP [13], který je blíže popsán v sekci 2.1.4 a pro popis okolí bodů využívá operátor LBP viz sekce 2.1.3

[2] Převzato z https://www.learnopencv.com/wp-content/uploads/2016/12/hog-histogram-1.png

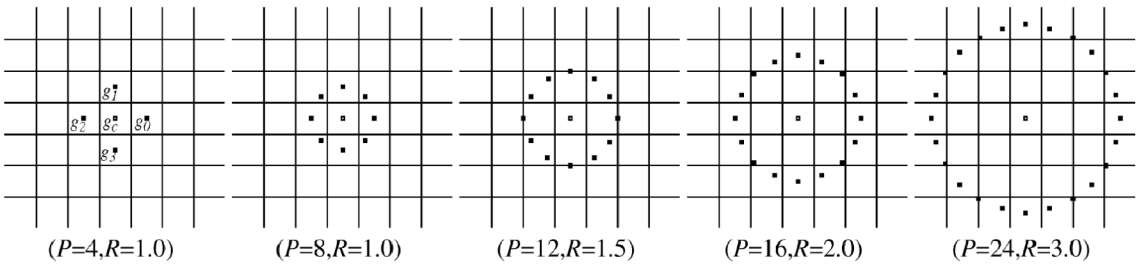


Obrázek 2.2: **Důležité body pro detekci obličeje**[3]

2.1.3 Local binary patterns

LBP (*Local Binary Patterns* - Lokální binární vzory) operátor je deskriptor textury popsaný v [14]. Pro popsání obrázku pomocí LBP se obrázek prvně převede na stupně šedi a následně se v určitém okolí každého pixelu vybere předem daný počet bodů viz obrázek 2.3. Od hodnoty intenzity barvy každého z bodů se odečte hodnota intenzity barvy pixelu, pro který je LBP počítán (hodnotu intenzity barvy bodu, který se nenachází uprostřed pixelu, lze zjistit pomocí interpolace z intenzity barev okolních pixelů). Zjištěné hodnoty jsou poté podle znaménka převedeny na 0 a 1 pomocí funkce (2.2) a jsou poskládány za sebe tak, aby vytvořily jedno binární číslo.

Toto umožňuje nalezení vždy stejných lokálních vzorů nezávisle na změně celkové světlosti obrázku. Pro docílení neměnnosti detekovaných vzorů i po rotaci celého obrázku lze zjištěné binární čísla binárně rotovat dokud není naleznena největší možná hodnota.



Obrázek 2.3: **Ukázka rozmístění bodů použitých pro výpočet hodnoty LBP jednoho pixelu obrázku** (P - počet bodů, R - vzdálenost bodů od středu v pixelech) [14]

$$s(x) = \begin{cases} 1, x \geq 0 \\ 0, x < 0 \end{cases} \tag{2.2}$$

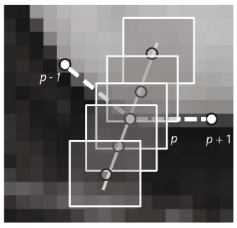
Díky tomuto operátoru je možné zakódovat okolí každého pixelu do podoby binárního čísla, které je schopno zachytit různé vzory v okolí pixelu (například: hrany, samostatné

[3]Převzato z www.pyimagesearch.com/wp-content/uploads/2017/04/facial_landmarks_68markup.jpg

body, gradienty, ...). Vzory lze následně snadno vyhledávat a porovnávat. LBP jde tedy snadno použít například pro detekci obličeje, nebo jiného tvaru v obrázku.

2.1.4 EP-LBP

EP-LBP (*Extended Profile Local Binary Patterns* - Rozšířené lokální binární vzory) [13] je rozšířením ASM ze sekce 2.1.2. EP-LBP využívá pro popis každého klíčového bodu p modelu ASM k bodů umístěných na normále k p se středem v p ve vzdálenosti d pixelů od sebe. Tento profil klíčového bodu pak nazýváme $g(p, k, d)$ a jeden profil klíčového bodu $g(p, 5, 2)$ (5 bodů na normále okolo bodu p ve vzdálenosti 2 pixely od sebe) si lze prohlédnout na obrázku 2.4. Pro zápis okolí každého bodu z profilu g je pak využito operátoru LBP [14] ze sekce 2.1.3. EP-LBP deskriptor klíčového bodu p je tedy sestaven konkatenací LBP deskriptorů všech k bodů profilu g



Obrázek 2.4: **Ukázka popisu jednoho klíčového bodu p pomocí profilu $g(p, 5, 2)$ modelu EP-LBP**

2.1.5 Triangular centroid distances

TCDs (*Triangular Centroid Distances* - Trojúhelníkové středové vzdálenosti) [19] je deskriptor tvaru. Nemění se při posunu, rotaci, změně velikosti ani při změně počátečního bodu popisu tvaru.

Pro výpočet TCDs je potřeba tvar popsat pomocí křivky podobně jako na obrázku 2.5, poté se křivka rozdělí na N bodů P_i ve stejné vzdálenosti od sebe, kdy $i = 1, 2, ..., N$ a P_1 je počáteční bod popisu tvaru. Po získání všech bodů se vypočítá střed tvaru $G = (x_g, y_g)$ pomocí:

$$\begin{cases} x_g = \frac{1}{N} \sum_{i=1}^{N} x_i \\ y_g = \frac{1}{N} \sum_{i=1}^{N} y_i \end{cases} \tag{2.3}$$

Pro každý bod $P_i = (x_i, y_i)(i = 1, 2, ..., N)$ jsou nalezeny všechny další body $P_j = (x_j, y_j)(j = 1, 2, ..., N, j \neq i)$. Tyto body pak se středovým bodem G tvoří trojúhelníky $\triangle P_i G P_j$ u každého trojúhelníku je pak vypočten jeho střed $g_{ij} = (x_{g_{ij}}, y_{g_{ij}})$ pomocí:

$$\begin{cases} x_{g_{ij}} = (x_i + x_G, +x_j)/3 \\ y_{g_{ij}} = (y_i + y_G, +y_j)/3 \end{cases} \tag{2.4}$$

Nyní je potřeba vypočítat deskriptory $TCDs(P_i)$ pro všechny body P_i podle následující rovnice:

$$TCDs(P_i) = |P_i g_{i1}|, ..., |P_i g_{ij}|, ..., |P_i g_{iN}| (i \in < 1, N >, j \in < 1, N > \wedge i \neq j) \tag{2.5}$$

Obrázek 2.5: **Hlava člověka popsaná pomocí křivky pro následné porovnávání pomocí algoritmu pro porovnávání tvarů**

Kde $|P_i g_{ij}| = \sqrt{(x_i - x_{g_{ij}})^2 + (x_i - x_{g_{ij}})^2}$ a (x_i, y_i), $(x_{g_{ij}}, y_{g_{ij}})$ jsou bod křivky P_i a střed trojúhelníku g_{ij}. Díky tomu lze získat TCDs deskriptor celého tvaru S následovně:

$$\begin{aligned} TCDs(S) &= (TCDs(P_1), ..., TCDs(P_N)) \\ &= \begin{pmatrix} |P_1 g_{12}| & \cdots & |P_N g_{N1}| \\ \vdots & \ddots & \vdots \\ |P_1 g_{1N}| & \cdots & |P_N g_{NN-1}| \end{pmatrix} \end{aligned} \tag{2.6}$$

Výsledkem je matice popisující tvar, která se při posunu tvaru v obrázku nemění, změna velikosti tvaru je však stále problém. Pro zajištění neměnnosti deskriptoru pro všechny velikosti tvaru je potřeba každý řádek matice znormalizovat pomocí následující rovnice:

$$|P_i g_{ij}| = \frac{|P_i g_{ij}|}{max_{i=1}^{N}\{|P_i g_{ij}|\}} \tag{2.7}$$

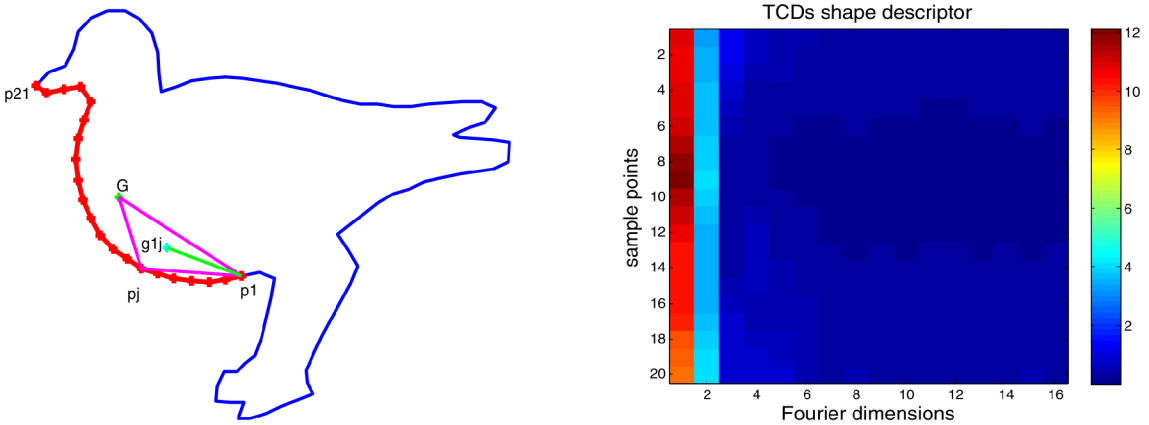
Rovnice (2.7) jednotlivé hodnoty deskriptoru převedla do intervalu $< 0, 1 >$. Tyto hodnoty jsou stejné pro všechny velikosti tvaru.

Po použití Fourierovy transformace na každý řádek matice deskriptoru a zanedbání fáze se deskriptor stává neměnný vůči rotaci tvaru. Pro to slouží následující 2 rovnice, kde $d_t, t \in < 1, N-1 >$ jsou jednotlivé řádky deskriptoru:

$$FD_t(i) = \frac{1}{N}\sum_{u=1}^{N} d_t(u) exp(\frac{-j2\pi(u-1)i}{N}), i = 1, 2, ..., N \tag{2.8}$$

$$TCDs = |FD_t(v)|, t = 1, 2, ..., N-1; v = 1, 2, ...M; M \ll N. \tag{2.9}$$

Výsledný deskriptor je neměnný vůči posunu, rotaci, změny velikosti a změny počátečního bodu popisu tvaru. Přiklad křivky a z ní vypočítaného deskriptoru si lze prohlédnout na obrázku 2.6



(a) Otevřená křivka (červeně)

(b) TCDs deskriptor otevřené křivky

Obrázek 2.6: **Příklad TCDs deskriptoru vypočteného z otevřené křivky (červená křivka vlevo)** [19]

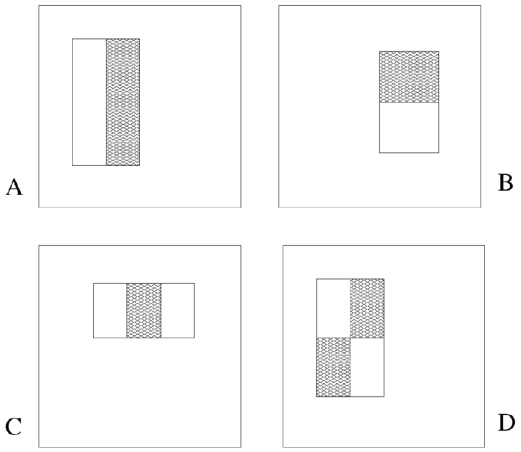
TCDs je velice dobrým deskriptorem pro porovnávání tvarů v různých situačních podmínkách a je také odolný vůči mnohým nepřesnostem popisu tvaru a šumu. Jeho použití však stěžuje nutnost tvar nejprve detekovat pomocí nějakého jiného algoritmu a následně jej co nejpřesněji popsat pomocí křivky.

2.1.6 Viola-Jones algoritmus

Viola-Jones je algoritmus od pánů Paul Viola a Michael J. Jones, který se používá k detekci objektů v obrázcích, s tím že při jeho vývoji byla hlavním cílem detekce obličejů hledících přímo do kamery.

Algoritmus funguje tak, že si původní obrázek převede do odstínů šedi. Dále se pro různé velikosti obrázku použije okno, které se po obrázku přesouvá a provádí se v něm detekce objektu. Detekce probíhá tak, že se z okna extrahují takzvané rysy. Extrakci si lze představit tak, že se do okna kladou obdélníky z obrázku 2.7 a od součtu hodnot pixelů pod černou barvou se odečtou hodnoty pod bílou barvou. Detektor je předem vytrénovaný, takže porovná vypočtenou hodnotu s hodnotou z tréninku a tím pro daný rys určí, zda odpovídá hledanému objektu či nikoliv. Pro urychlení se detekce provádí v etapách. V

prvotních etapách se extrahují relativně jednoduché rysy, které mají za úkol určit okna, kde se hledaný objekt nenachází, takže se jím další etapy nemusí vůbec zabývat. Okno, které úspěšně projde všemi etapami, se považuje za okno obsahující hledaný objekt.



Obrázek 2.7: **Příklad obdélníkových rysů zobrazených relativně k detekčnímu oknu.** Součet hodnot pixelů ležících pod bílými obdélníky je odečten od součtu hodnot pixelů ležících pod černými obdélníky. [18]

V souvislosti s tímto algoritmem hovoříme o takzvaných haar kaskádách, což je posloupnost fází sloužící pro detekci objektu. [18]

Integrovaná reprezentace obrázku

Jelikož by sčítání a odčítání všech hodnot pixelů v daném okně zabrala příliš mnoho času, je obrázek nejprve převeden do takzvané integrované reprezentace. Tato reprezentace spočívá v nahrazení hodnoty každého pixelu součtem hodnot všech pixelů vlevo a nad tímto pixelem. Díky tomu se sčítání provádí pouze jednou a zjištění součtu pixelů při extrakci rysů zahrnuje pouze několik operací. Tyto operace jsou popsány u obrázku 2.8

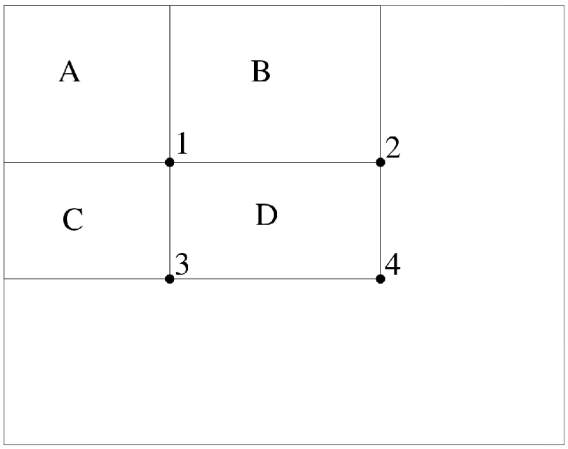
2.2 Detekce a porovnávání obličejů

Tato sekce nabízí přehled některých algoritmů, které se specializují na detekci a porovnávání obličejů. Většina z nich nějakým způsobem využívá algoritmy pro detekci a porovnání tvarů (sekce 2.1).

Pro většinu algoritmů je důležité obličej nejprve na fotografii detekovat. Následně jsou detekovány důležitá místa obličeje (oči, ústa, nos, obočí). Pomocí rozmístění a tvaru těchto míst jsou algoritmy schopny určit podobnost obličejů mezi obrázky, díky tomu lze najít nejpodobnější obličej a tím identifikovat člověka, kterému obličej patří. Některé algoritmy se na základě těchto míst pokoušejí určit i pohlaví, stáří, nebo náladu vyfoceného člověka.

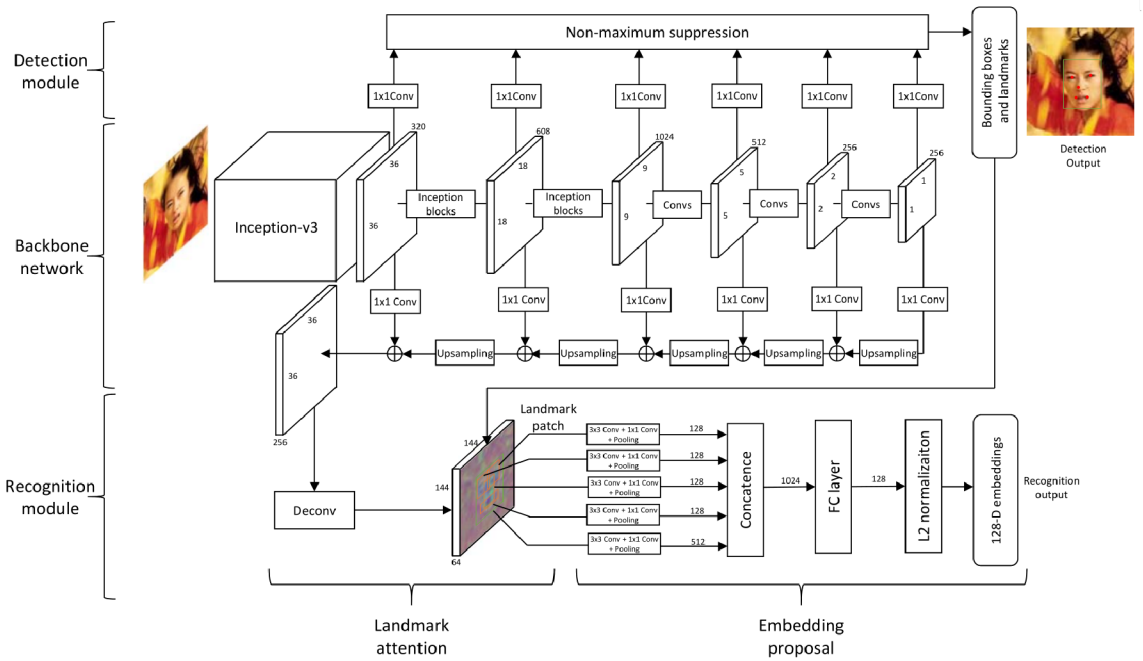
2.2.1 Uniface

Uniface (*A Unified Network for Face Detection and Recognition*) [9] je algoritmus využívající konvoluční neuronovou síť pro detekci a rozpoznání tváře zároveň (schéma sítě si lze



Obrázek 2.8: **Integrovaná reprezentace obrázku** Pro vypočítání součtu pixelů v oblasti D je potřeba provést pouze následující výpočet: 4 - 2 - 3 + 1. [18]

prohlédnout na obrázku 2.9). Díky provádění detekce i rozpoznání zároveň je algoritmus přesnější, protože je schopen zachytit některé důležité detaily tváře, které se při samostatné detekci a následném rozpoznání ztrácí. Další výhodou spojení těchto operací je vyšší rychlost zpracování obrázků s více tvářemi. Uniface je totiž spouštěn pro každý obrázek pouze jednou, zatímco většina jiných algoritmů musí provést rozpoznávání tváře pro každou tvář zvlášť.



Obrázek 2.9: **Schéma konvoluční neuronové sítě Uniface.** [9]

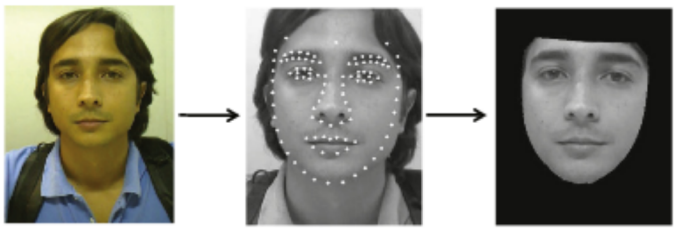
Detekce Podobá se algoritmu SSD [10]. Pro detekci tváře se využívá cesta sítí z obrázku 2.9 směrem nahoru. Pomocí aplikace konvolučních filtrů na různá rozlišení obrázku v něm algoritmus nalezne tváře i všechna jejich důležitá místa (ústa, nos, oči, ...).

Rozpoznávání Zároveň je použita cesta neuronovou sítí z obrázku 2.9 směrem dolů a z těchto detekovaných míst je vypočítán 128D vektor. Tento vektor lze následně porovnávat s vektory jiných tváří a tím nalézt obrázky tváří s nejlepší shodou.

2.2.2 Použití nepravidelností obličeje pro zpřesnění rozpoznávání

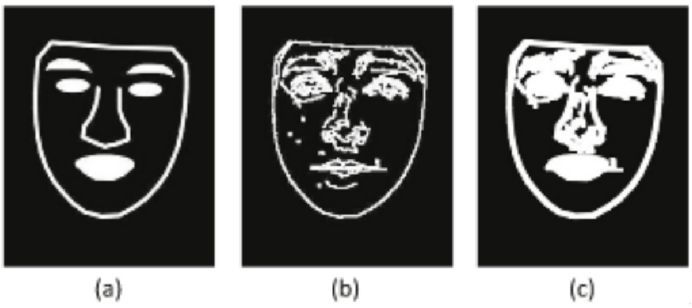
Podle [1] je možné použít nepravidelnosti obličeje (znaménka, jizvy, bradavice, tetování, pigmentové skvrny, ...) pro zpřesnění porovnání obličeje.

V originální práci je nejprve detekován obličej spolu s hlavními klíčovými body (oči, nos, obočí, ústa) pomocí EP-LBP modelu viz sekce 2.1.4. Takto detekovaná tvář je poté z fotografie vyříznuta, jak lze vidět na obrázku 2.10. Následně je zkonstruována maska, která



Obrázek 2.10: **Výřez oblasti obličeje detekovaného pomocí EP-LBP** [1]

slouží pro překrytí očí, úst, nosu, obočí a případných vousů, protože v těchto oblastech tváře je obtížné spolehlivě detekovat nepravidelnosti obličeje uvedené výše. Ke konstrukci masky jsou použity již dříve detekované klíčové body, ke kterým jsou přidány všechny detekované hrany dotýkající se těchto oblastí, díky čemuž jsou zamaskovány i například vousy. Průběh konstrukce masky je znázorněn na obrázku 2.11. Výsledná maska je k vidění na obrázku (c). Po konstrukci masky jsou v obrázku pomocí Laplacian-of-Gaussian (LoG) filtru [12]

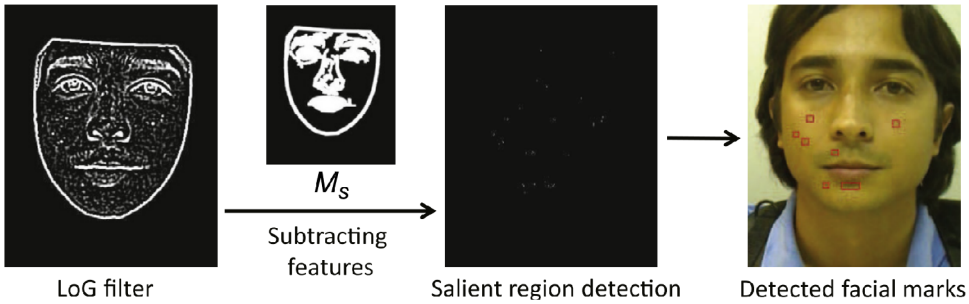


Obrázek 2.11: **Konstrukce masky okolo příliš rušných oblastí obličeje** (a) - maska zkonstruována z oblastí detekovaných pomocí EP-LBP;(b) - detekované hrany v obličeji; (c) - maska z (a) zkombinovaná s hranami z (b), které se masky dotýkají;[1]

detekovány nerovnosti obličeje. Nakonec je od detekovaných nerovností odečtena před chvílí zkonstruovaná maska. 8 x 8 pixelů velké okolí kolem každé detekované nerovnosti mající alespoň 2 pixely na šířku i na výšku je následně zakódováno pomocí HoG (Histogram

of Oriented Gradients - Histogram orientovaných gradientů) deskriptoru [4] popsaného v sekci 2.1.1. Samotný HoG deskriptor by však nestačil, protože se může například objevit podobně vypadající znaménko na bradě i na čele. Proto je k HoG deskriptoru přidán také normalizovaný údaj o poloze nerovnosti v rámci obličeje.

Použití filtru a masky spolu s výsledkem detekce nepravidelností si lze prohlédnout na obrázku 2.12.



Obrázek 2.12: **Průběh samotné detekce nepravidelností v obličeji a demonstrace výsledku.** Zleva lze vidět nejprve výsledek filtrování pomocí Laplacian-of-Gaussian filtru, poté zkonstruovanou masku, dále výsledné detekované nepravidelnosti a vpravo vyznačení nepravidelností na vstupní fotografii. [1]

Podle [1] jsou tyto nepravidelnosti samotatně pro porovnávání obličejů nedostatečné. Pokud se však porovnávání nerovností obličeje připojí k nějakému jinému algoritmu pro porovnávání tváří, může dojít k jeho značnému zpřesnění.

V případě této práce, pokud se podaří na fotografii najít celý obličej, pravděpodobně z něj bude možné extrahovat dostatek informací pro spolehlivé porovnání a tyto nerovnosti tím pádem mají pouze malý význam. Pokud ale půjde o fotografii hlavy na které bude vidět pouze část obličeje, detekce podobných nerovností se může ukázat jako zásadní zdroj informací potřebných pro porovnání.

Kapitola 3

Návrh implementace

Momentálně se většina projektů soustředí na detekci a následné porovnání pouze tváří. V případě kdy se tedy osoba na obrázku dívá jiným směrem než ke kameře, tak se detekce a tudíž i následné porovnání nemusí vůbec zdařit. Často je také nutné mít k dispozici databázi obličejů, kterou se algoritmus musí nejprve naučit a až poté je v ní schopen dotazovaný obličej vyhledat. Při přidání nového obličeje je tedy často nutné algoritmus znovu učit na nové rozšířené databázi. V této práci bych chtěl vytvořit algoritmus, který bude porovnávat vždy pouze dva obrázky s hlavami a vypíše jejich rozdílnost. Není tedy nutné algoritmus cokoliv předem učit. Algoritmus bude schopen detekovat a porovnávat hlavy s natočením až 90 stupňů doleva či doprava a s náklonem až 45 stupňů nahoru nebo dolů a libovolné kombinace tohoto natočení a náklonu. Detekce bude z větší části vycházet z detektorů pro detekci tváří hledících přímo na kameru, tudíž se kvalita detekce a následného porovnání bude s přibývající odchylkou pohledu od kamery snižovat. Chci však dosáhnout alespoň 80 procentní celkové úspěšnosti detekce hlavy ve výše stanoveném rozmezí její polohy a detekce alespoň poloviny jejich částí (při fotografii z profilu lze vidět jedno ucho, jedno oko, část úst a nos a mým cílem je detekovat alespoň libovolné dvě z těchto částí). Jelikož kvalita detekce s přibývající odchylkou bude klesat, bude nutně klesat i kvalita porovnání, protože méně detekovaných částí znamená méně věcí, které se dají porovnat. Výsledek algoritmu tedy se stoupající odchylkou může být více a více zavádějící. Měl by však vždy být natolik dobrý, aby i v nejobtížnější situaci při porovnávání fotografie jednoho člověka s mnoha jinými fotografiemi, mezi kterými je také další fotografie tohoto člověka byla tato fotografie mezi 50 procenty nejpodobnějších fotografií. Algoritmus by tak mělo jít použít pro seřazení fotografií podle podobnosti hlav a tím urychlit například následné ruční procházení těchto fotografií.

Pro porovnání bych chtěl vyjít z algoritmu pro zpřesnění rozpoznávání obličeje pomocí nepravidelností obličeje, který je blíže popsaný v sekci 2.2.2. Avšak místo použití znamének, tetování, jizev a podobných nepravidelností do porovnávání zahrnu části jako jsou oči, uši, nos a nebo ústa.

Na vstupu do algoritmu obrázky nemusí být zcela kvalitní, proto jako první na řadu přijde nějaké předzpracování jako je například úprava rozlišení nebo odstranění šumu. Následně je nutné provést detekci samotné hlavy, ta je velice důležitá pro správnou funkci zbytku algoritmu, proto chci poskytnout více způsobů detekce, aby v případě nezdaru při detekci jedním způsobem šlo použít jiný způsob, který snad bude mít větší úspěch. Poskytnutí více způsobů detekce také umožňuje použít různé způsoby detekce pro různé pozice hlavy, například je možné mít odlišný detektor pro hlavu natočenou přímo na kameru a pro hlavu natočenou z profilu. Po detekci hlavy se bude pokračovat detekcí různých částí

obličeje, jako jsou oči, uši, nos a ústa. Na závěr se takto detekované hlavy a jejich části popíšou pomocí histogramů orientovaných gradientů, tyto histogramy se zapíší jako vektory, které se porovnají a vypíše se hodnota rozdílnosti hlav.

3.1 Detekce

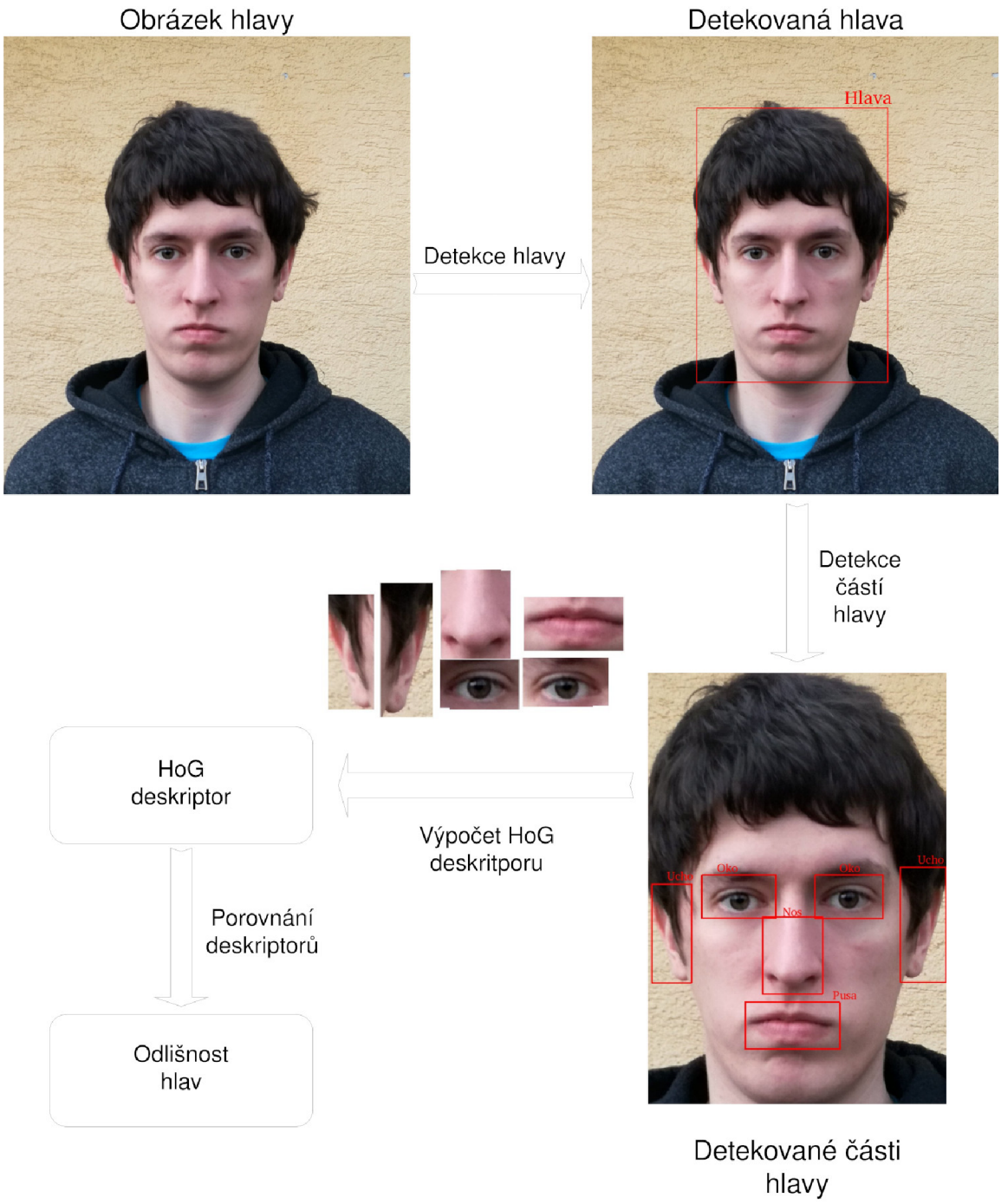
Tato práce se nezabývá přímo detekcí hlavy ani žádné její části. Pro detekci hlavy, obličeje a jejich částí bych tedy chtěl použít již existující detektory.

Detekce hlavy je velice důležitá, proto pro ni chci použít několik různých způsobů, aby v případě nefunkčnosti detekce jedním způsobem mohl být použitý jiný způsob detekce. Způsoby by měly zahrnovat použití haar kaskád 2.1.6, neuronové sítě, které podle článku [6] mají velice dobrou úspěšnost detekce. Detekce hlavy by také měla jít vynechat, což uživateli dává možnost obrázek oříznout ručně a algoritmus by poté předpokládal, že celý obrázek je pouze hlava.

Pro detekci očí, nosu, úst a uší bych chtěl použít detektory na bázi haar kaskád 2.1.6, LBP 2.1.3 nebo histogramu orientovaných gradientů 2.1.1.

3.2 Porovnání

Pro porovnání bude vypočítán histogram orientovaných gradientů 2.1.1 pro každou část hlavy (uši, oči, nos a ústa), čímž se získá vektor popisující tvar každé části. Vektory každé části se mezi hlavami porovnají a vypočítá se průměrná odlišnost tvarů všech částí. Uživatel bude také mít na výběr, zda chce zohlednit také to, že některé části byly detekovány pouze na jedné z hlav případně, že pozice některé části je na jedné hlavě příliš odlišná než na druhé.



Obrázek 3.1: **Schéma návrhu algoritmu** Navrhovaný postup detekce a následného porovnání hlav.

Kapitola 4

Implementace

V této kapitole popíšu použité programovací jazyky, použité knihovny a postup implementace práce.

4.1 Použité programovací jazyky

Pro implementaci práce jsem použil několik jazyků. Python je využit pro celou implementaci algoritmu pro detekci a porovnání hlav. Krátký shell skript je využit pro usnadnění testování programu.

4.1.1 Python

Python je vyšší, dynamicky tipovaný, objektově orientovaný, interpretovaný programovací jazyk. Jeho jednoduchost umožňuje snadno a rychle naprogramovat i rozsáhlé programy, které jsou díky jeho přenositelnosti spustitelné na spoustě různých platformách. [16]

Při výběru jazyku pro implementaci práce jsem vycházel z jazyků, pro které je dostupná knihovna OpenCV, která je pro počítačové vidění v současnosti téměř nutností. Těmito jazyky jsou C++, Python a Java. S Javou mám pouze velice málo zkušeností a C++ je v porovnání s Pythonem pro tuto práci příliš složité.

Celý projekt pro porovnání dvou hlav je napsán právě v pythonu.

4.1.2 Shell

Jako shell se v unixových systémech označuje příkazový interpret a programovací jazyk. Jako příkazový interpret lze shell použít pro zadávání příkazů, které lze využít například pro pohyb v souborovém systému, zápis do souboru, čtení do souboru atd. Jako programovací jazyk shell umožňuje příkazy kombinovat. Takto zkombinované příkazy lze uložit do souboru, kterému se říká shell skript a ten lze následně použít jako nový příkaz. [5]

V této práci je jeden skript použit pro usnadnění testování. Na vstupu skriptu se zadají dva adresáře obsahující obrázky hlav. Skript porovná každou hlavu z prvního adresáře s každou hlavou z druhého adresáře a výsledky zapíše do csv souboru.

4.2 Použité knihovny

Pro usnadnění implementace jsem v projektu při implementaci použil několik knihoven.

4.2.1 OpenCV

OpenCV je otevřená knihovna, která se používá pro počítačové vidění a počítačové učení. Knihovna obsahuje přes 2500 algoritmů pro detekci a rozpoznání tváří, identifikaci objektů a úpravu obrázků. [15]

OpenCV také obsahuje řadu nástrojů pro usnadnění tréninku nových detektorů. Tyto nástroje umožňují snadné vyznačení hledaného objektu na obrázcích. Další nástroj je schopen pomocí malých změn velikosti a rotací obrázku generovat mnoho dalších obrázků pro učení. Některé nástroje dále umožňují vyobrazení stavu detektoru v různých stádiích učení. Nejdůlezitějším nástrojem je však nástroj, který z mnoha obrázků obsahujících vyznačený hledaný objekt, vytrénuje detektor na bázi haar kaskád, LBP nebo histogramů orientovaných gradientů. Všechny z těchto nástrojů jsem při tréninku detektoru využil.

V této práci je OpenCV použita pro načítání a úpravu obrázků, k načtení a použití detektorů na bázi haar kaskád, LBP a histogramů orientovaných gradientů, které jsou použity pro detekci hlavy a jejích částí. Dalším důležitým využitím této knihovny v práci je výpočet histogramů orientovaných gradientů hlav a jejich částí, které jsou poté použity pro výpočet rozdílnosti hlav. V práci jsou využity také funkce knihovny pro úpravu obrázků, jako je zvětšení nebo zmenšení obrázků, rozmazání, ořezání, kreslení do obrázků pro vyznačení detekovaných oblastí, převod obrázků do odstínů šedé a zrdcadlení obrázků v situacích, kdy je detekce schopna detekovat obličej pouze z jedné strany.

4.2.2 Dlib

Dlib je moderní otevřená knihovna, obsahující algoritmy pro počítačové učení, zpracování obrázků a mnoho dalších. Knihovna skvěle doplňuje knihovnu OpenCV. Mezi její oblasti využití patří například robotika, vestavěné systémy, mobilní telefony. Hojně se využívá v průmyslu i pro akademické účely. [8]

V této práci je Dlib využita hlavně pro detekci hlavy. Důležitým algoritmem využitým v této práci je detektor pro detekci hlavy na bázi histogramu orientovaných gradientů. V situaci, kdy je hlava natočena přímo, nebo téměř přímo na kameru, se zdá tento algoritmus nejlepší, je nejrychlejší a kvalita detekce je nejvyšší. Další využitou funkcí této knihovny je možnost načíst a použít předtrénovaný detektor na bázi neuronových sítí. Tento detektor je schopen detekovat hlavy v nejvíce pozicích, jeho nevýhodou však je jeho velmi malá rychlost. V situaci, kdy je k dispozici pouze procesor a ne dedikovaná grafická karta může detekce zabrat i několik minut, zatímco jakýmkoliv jiným algoritmem využívaným v této práci by detekce trvala nejdéle několik sekund. Poslední věcí z této knihovny využitou v této práci je třída rect. Tato třída je využitá pro uložení všech detekovaných oblastí.

4.2.3 NumPy

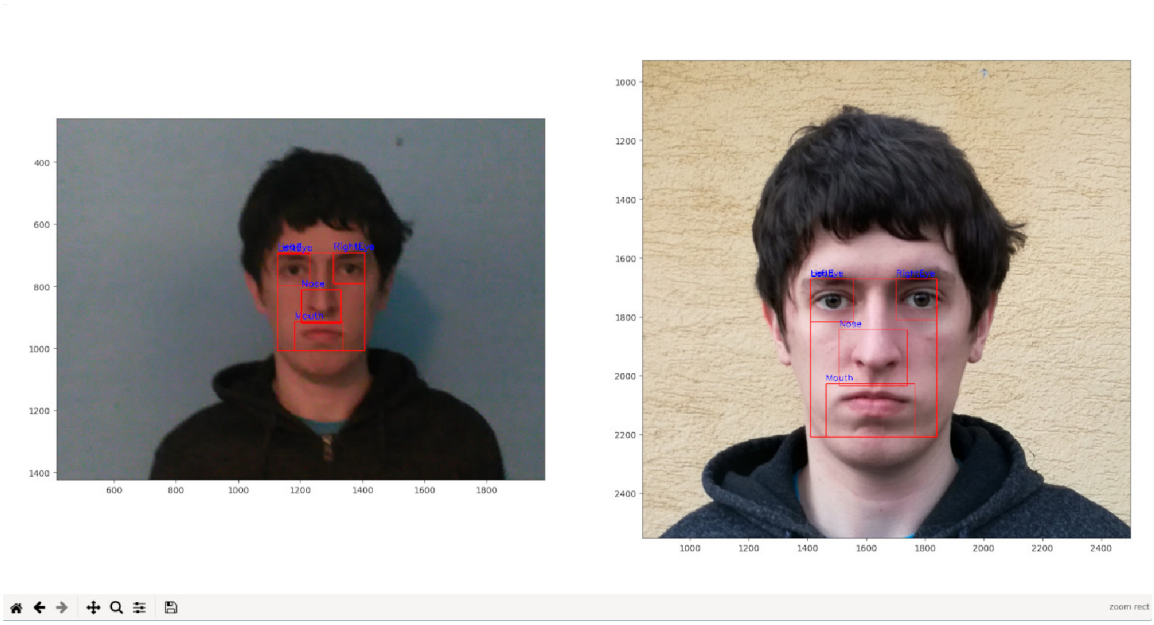
NumPy je otevřená matematická knihovna, která je významná pro vědecké výpočty v jazyce Python. Tato knihovna umožňuje výpočty nad velkými vícedimenzionálními polemi a maticemi, které jsou díky NumPy několikanásobně rychlejší. [17]

V této práci NumPy umožnilo rychlé a snadné odečtení dvou vektorů získaných výpočtem histogramu orientovaných gradientů. Tento rozdíl vektorů poté určuje rozdílnost hlav.

4.2.4 Matplotlib

Matplotlib je rozsáhlá knihovna sloužící pro tvoření statických, animovaných a interaktivních vizualizací. [7]

Tato knihovna je v práci využita pro zobrazení obrázků po provedení detekce při zadání parametru show při spuštění skriptu.



Obrázek 4.1: **Výsledek detekce hlav zobrazený pomocí knihovny matplotlib** Díky této knihovny je možné zobrazené obrázky libovolně přibližovat a posouvat s nimi. Umožňuje tedy detailní zkoumání provedené detekce hlavy a jejích částí.

4.3 Postup implementace práce

Při implementaci jsem nejprve využil Dlib a OpenCV pro načtení předtrénovaných detektorů a detekci hlav. Následně jsem pomocí několika haar kaskád detekoval části hlavy (oči, uši, ústa a nos) a pomocí OpenCV a Numpy jsem vypočítal histogramy orientovaných gradientů hlav a jejich rozdíl, který udává rozdílnost hlav.

4.3.1 Detekce

Detekce hlavy a jejích částí je stěžejní pro správné fungování zbytku práce.

Úplně nejdůležitějším krokem je správná detekce hlavy, která se provádí jako první. Jelikož je tento krok tak důležitý a jelikož většina detektorů funguje pouze pro malé rozmezí natočení hlavy, je použito více různých přístupů k detekci. Používá se vnitřní algoritmus knihovny dlib, několik haar kaskád, LBP kaskáda a neuronová síť. Každý z algoritmů nabízí jinou přesnost a spolehlivost detekce a jiné rozmezí natočení hlavy a proto se skvěle doplňují (při nefunkčnosti jednoho je šance, že jiný bude fungovat).

Po detekci hlavy následuje detekce jejích jednotlivých částí jako jsou nos, oči, uši a ústa pomocí haar kaskád a následného ověření, že rozmístění takto detekovaných částí opravdu odpovídá možné hlavě.

Detekce hlavy

Pro detekci hlavy je možné použít hned několik různých algoritmů. Uživatel si tak buď může zvolit, který algoritmus chce použít, nebo nechá skript automaticky postupně použít všechny algoritmy, až dokud nenalezne alespoň nějakou hlavu.

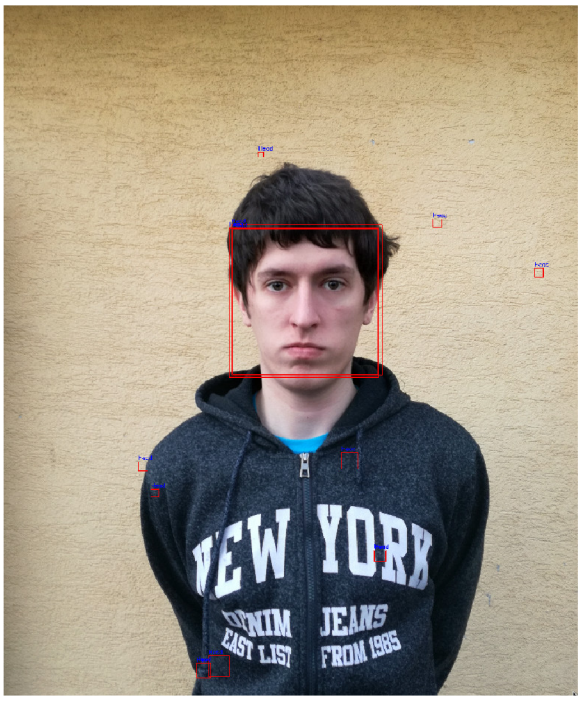
Prvním algoritmem je algoritmus pro detekci obličejů z knihovny dlib. Je schopen relativně rychle a přesně detekovat obličeje dívající se přímo do kamery. Lze použít i pro detekci lehce otočených obličejů (asi 20 stupňů v každém směru).

Obrázek 4.2: **Detekce hlavy pomocí algoritmu z knihovny dlib.**

Při neúspěchu předchozího algoritmu je použito několik volně dostupných předtrénovaných haar a LBP kaskád určených pro detekci obličejů dívajících se do kamery a jedné haar a LBP kaskády pro detekci obličejů z levého profilu. Jelikož detekce z profilu funguje pouze na levý profil, tak je tato detekce vždy provedena i na zrdcadlově obráceném obrázku a při úspěšné detekci je detekovaná pozice hlavy přepočítána tak, aby odpovídala původnímu obrázku. Tento algoritmus je díky možnosti detekce hlavy z levého profilu schopen detekovat i hlavy, které nejsou natočeny přímo do kamery (asi 90 stupňů v každém směru). Oproti předchozímu algoritmu však vyzkoušení všech kaskád trvá o něco déle a produkuje mnohem více falešně pozitivních detekcí. Jelikož tyto falešně pozitivní detekce jsou většinou pouze malé nerovnosti na pozadí, tak jsou nejprve odstraněny všechny detekované hlavy, které se nacházejí uvnitř jiné hlavy a pak je ze všech detekovaných oblastí vybrána ta největší.

Při neúspěchu předchozích algoritmů je použita neuronová síť, o které se zmiňuje [6]. Tato detekce se zdá být velice přesná, spolehlivá a použitelná ve velkém množství natočení hlav. Její největší nevýhodou je však časová náročnost detekce při nedostupnosti dedikované grafické karty, kdy jeden obrázek může v závislosti na jeho velikosti a výkonnosti procesoru zabrat i více než 5 minut.

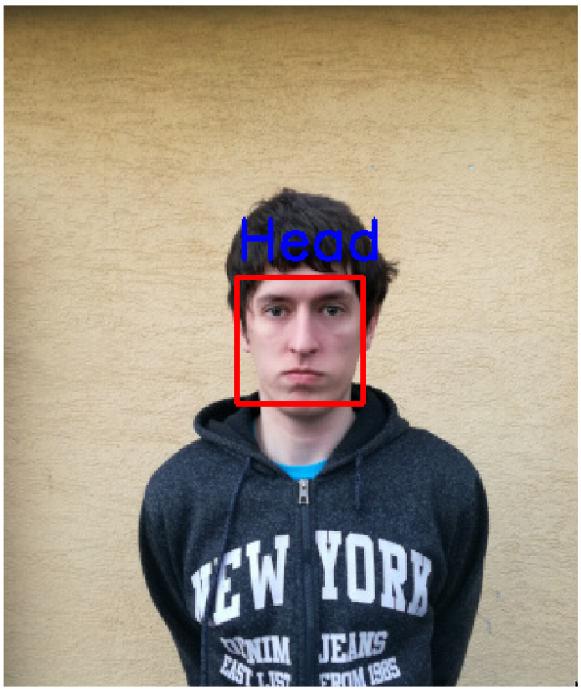
Pokud selžou všechny předchozí pokusy o detekci hlavy, tak se pro zbytek algoritmu předpokládá, že celý obrázek obsahuje pouze hlavu.



Obrázek 4.3: **Detekce hlavy pomocí haar a LBP kaskád.** Lze vidět, že hlava je detekována několikrát, to je způsobeno tím, že bylo použito několik různých kaskád a více z nich hlavu úspěšně nalezlo. Na obrázku lze ale také vidět řadu falešně pozitivních detekcí. Tyto oblasti jsou však téměř vždy relativně malé a proto jsou po výběru největší detekované oblasti zahozeny.

Někdy se může stát, že je na jednom obrázku detekováno více hlav. Důvodem může být nesprávná detekce hlavy případně se na obrázku skutečně může nacházet více hlav. Z pohledu algoritmu nedává přítomnost více hlav v jednom obrázku žádný smysl a z pohledu uživatele je velice složité určit kterou hlavu by měl algoritmus použít. V případě detekce více hlav ve stejném obrázku je proto dále použita pouze hlava s větší detekovanou plochou. V případě více hlav na obrázku je to tedy pravděpodobně hlava v popředí a v případě nesprávné detekce jsou tyto detekce většinou oproti skutečným hlavám velice malé, takže se tedy pravděpodobně vybere správná hlava.

Algoritmus detekce hlavy je blíže popsán python pseudokódem 4.1.



Obrázek 4.4: **Detekce hlavy pomocí neuronových sítí.** Na obrázku lze vidět hlavu detekovanou neuronovou sítí. Obrázek musel být kvůli dlouhému trvání detekce na původním obrázku s vysokým rozlišením 100 krát zmenšen. Detekce na původním obrázku pouze s použitím procesoru trvala déle než 5 minut. Detekce na zmenšeném obrázku trvala několik sekund a její rychlost tak byla srovnatelná s předchozími dvěma algoritmy.

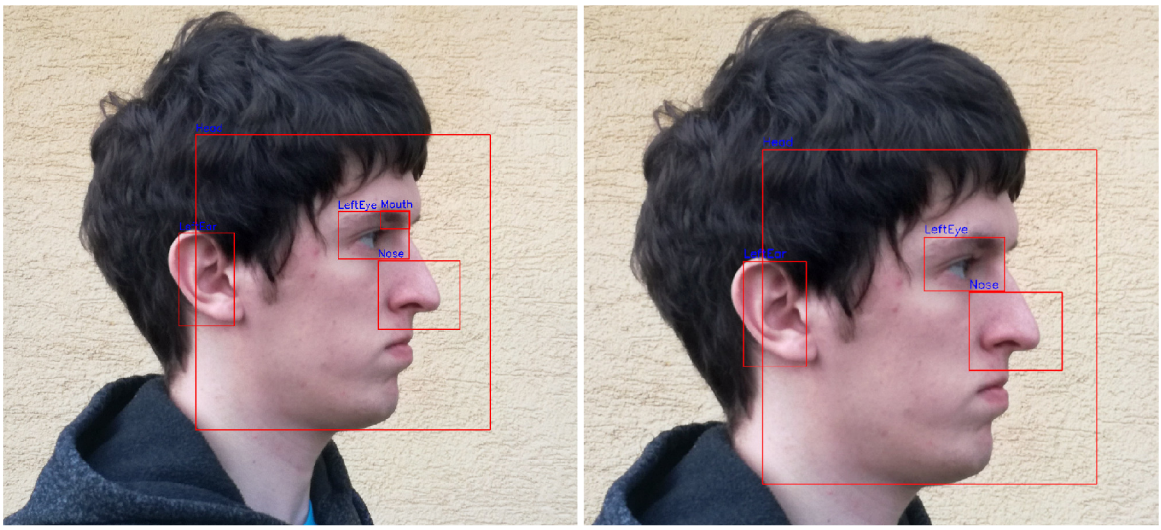
```
def detekce_hlavy(zadany_obrazek):
    for typ in typy_detekci:
        detekovane_hlavy.append(detektor(zadany_obrazek))

    for hlava in detekovane_hlavy:
        for hlava2 in detekovane_hlavy:
            if hlava == hlava2:
                continue
            if hlava.contains(hlava2):
                detekovane_hlavy.remove(hlava2)

    return nejvetsi_hlava(detekovane_hlavy)
```

Pseudokód 4.1: Detekce hlavy

Detekce částí hlavy

Před samotnou detekcí jednotlivých částí hlavy (uši, nos, ústa, oči) je oblast hlavy lehce rozmazána, aby se zamezilo nesprávným detekcím způsobených příliš rušným pozadím a různými útvary na kůži, jako jsou mateřská znaménka nebo jizvy.

Samotná detekce probíhá pomocí několika volně dostupným haar kaskádám. Jednou z nich je i kaskáda pro detekci úst vycvičená pro výzkum detekce tváří v reálném čase [2].



Obrázek 4.5: **Odstranění falešně pozitivních detekcí pomocí znalosti rozmístění jednotlivých částí hlavy** Na obrázku vlevo lze vidět, že byla nesprávně detekována ústa v oblasti oka. Tato detekce je následně na obrázku vpravo odstraněna, jelikož víme, že ústa se musí nacházet v dolních 50% hlavy.

Vždy po provedení detekce dochází k odstranění nesprávně detekovaných částí. Při detekci částí, z nichž je jedna uvnitř jiné, se vždy ponechá pouze vnější část. Dále jelikož známe přibližné rozmístění jednotlivých částí na hlavě, tak jsou odstraněny všechny nesmyslné části následovně (jejich detekce je provedena znovu):

1. Ucho musí být vždy nejlevější a nebo nejpravější část hlavy. Pokud tomu tak není, je ucho odstraněno
2. Oko musí být vždy v horních 50% hlavy. Pokud tomu tak není, je ucho odstraněno
3. Ústa musí být vždy v dolních 50% hlavy. Pokud tomu tak není, jsou ústa odstraněna.
4. Nos musí vždy být výškově mezi očima a ústy. Pokud tomu tak není, je nos odstraněn.
5. Pokud je hlava natočena přímo do kamery, tak musí být nos vždy stranově mezi oběma očima. Pokud tomu tak není, je nos odstraněn.

Při této kontrole je také zjištěna strana každého oka a ucha a je správně označena. Celý algoritmus detekce částí hlavy je zkráceně popsán python pseudokódem 4.2.

```
def detekuj_cast(hlava, vyloucene_casti, detektor):
    detekovane_casti = detektor(hlava)
    for cast in detekovane_casti:
        if !vyloucene_casti.contains(cast):
            return cast

def detekce_casti_hlavy(hlava):
    nos = detekuj_cast(hlava, None, detektor_pro_nos)
    usta = detekuj_cast(hlava, None, detektor_pro_usta)
    usi = list(detekuj_cast(hlava, None, detektor_pro_ucho))
    usi.append(detekuj_cast(hlava, usi[0], detektor_pro_ucho))
    oci = list(detekuj_cast(hlava, None, detektor_pro_oko))
    oci.append(detekuj_cast(hlava, oci[0], detektor_pro_oko))

    # Vyrazeni casti na zaklade znalosti jejich vzajemych pozic
    # Ucho je nejlevejsi, nebo nejpravejsi casti hlavy

    while neco bylo vylouceno:
        for index, ucho in usi:
            if !je_nejpravejsi(ucho, nos, usta, oci) and
               !je_nejlevejsi(ucho, nos, usta, oci):
                usi[index] = detekuj_cast(hlava, vyloucene_usi,
                    detektor_pro_ucho)
                vyloucene_usi.append(ucho)

    # Stejnym zpusobem zajistime i splneni vsech nasledujicich podminek:
    # Oko je v horni polovine hlavy
    # Usta jsou v dolni polovine hlavy
    # Nos je vyskove mezi ocima a usty a stranove mezi ocima,
    # pokud je hlava natocena do kamery

    return nos, usta, usi, oci
```

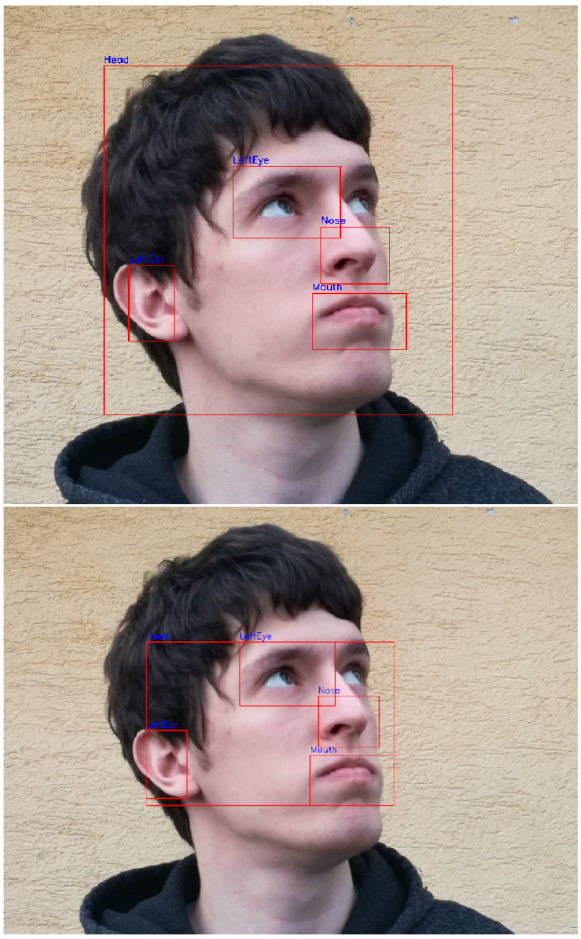
Pseudokód 4.2: Detekce částí hlavy

4.3.2 Porovnání

Porovnání dvou hlav je to, k čemu směřovala veškerá předchozí detekce hlav a jejich částí a to hlavní, čím se zabývá tato práce.

Před samotným porovnáváním je každá hlava ještě ořezána tak, aby obsahovala pouze oblasti s detekovanými částmi hlavy. Hlava se tak zbaví přebytečného pozadí, které by mohlo zkreslovat porovnání. Zároveň teď bude porovnávaná oblast tvořena hlavně očima, nosem, ústy, ušima a oblastmi mezi nimi, které mají mnohem vyšší vypovídací hodnotu pro porovnání, než třeba oblast krku, nebo vlasy, které by při změnách účesu mohly porovnání mást. Dále je pro snadnější porovnání změněna velikost na 175 x 175 px.

Uživatel má na výběr z několika porovnávacích algoritmů, které zohledňují různé faktory. Na pozadí každého algoritmu se však provádí výpočet histogramu orientovaných gradientů



Obrázek 4.6: **Odstranění nepotřebných částí z oblasti detekce hlavy pomocí ořezání k okrajům částí hlavy.** Na obrázku vlevo lze vidět vyznačenou detekovanou oblast hlavy a oblasti jejích částí před provedením ořezání. Na obrázku vpravo je oblast hlavy zařezána k okrajům oblastí s detekovanými částmi hlavy. Tím je z oblasti hlavy odstraněno nepotřebné pozadí vpravo a vlasy nahoře, které by v případě změny účesu mohly algoritmus mást. Zároveň je díky ořezání oblast hlavy soustředěna ta oblast obličeje, která má při porovnávání pro nás největší infomační hodnotu.

2.1.1, který každou hlavu případně její část popíše vektorem a rozdíl těchto vektorů určuje rozdílnost hlav nebo jejich částí.

Základní algoritmus porovnává histogram orientovaných gradientů z celých hlav (po ořezání tak, aby obsahovala pouze oblasti s detekovanými částmi hlavy).

Další algoritmus porovnává histogramy pro jednotlivé části hlavy a jeho výsledkem je průměr rozdílností jednotlivých částí hlav. Všechny další algoritmy vycházejí z tohoto algoritmu.

Při detekci částí hlav se může stát, že se některá z částí nepovede detekovat. Důvodem může být například neobvyklý tvar části, například následkem úrazu, špatná kvalita obrázku, nebo nedokonalost detektoru. Při pokusu o porovnání dvou různých obrázků hlavy stejného člověka s hlavou ve stejné pozici a stejným detektorem však lze předpokládat, že se povede na obou obrázcích detekovat stejné části hlavy. Tento algoritmus proto nejprve vypočítá průměrnou rozdílnost částí hlavy, které se podařilo detekovat na obou hlavách

(stejně jako předchozí algoritmus), ale poté za každou část detekovanou pouze na jedné z hlav k výsledné rozdílnosti přičte 5%.

Dalším rozdílem mezi hlavami různých lidí je pozice částí hlavy (někdo může mít oči blíže u sebe než někdo jiný). Proto je nejprve vypočítána maximální možná vzdálenost dvou bodů na hlavě, což je vzdálenost mezi dvěma nesousedícími rohy detekované oblasti hlavy. Pokud se pozice některé části hlavy liší o více než 3% této vzdálenosti, je podobně jako u předchozího algoritmu, k průměrné rozdílnosti částí hlavy přičteno 5% za každou takovou část.

Poslední algoritmus kombinuje oba předchozí algoritmy. K průměrné rozdílnosti částí hlav tedy přičítá 5% za každou část, která byla detekována pouze na jedné z hlav a za každou část, jejíž pozice se mezi hlavami příliš liší.

Poslední algoritmus je blíže popsán python pseudokódem 4.3.

```
def porovnani(hlava1, hlava2):
    resize_heads(hlava1, hlava2)
    odlisnosti = list()
    # Vypocet 3% vzdalenosti uhlopricky obdelnika ohranicujiciho hlavu.
    max_vzdalenost = sqrt(hlava1.sirka, hlava1.vyska) / 100 * 3
    pocet_vzdalenych_casti = 0
    for cast1 in hlava1.casti:
        for cast2 in hlava2.casti:
            # Pokud se jedna o stejnou cast (napriklad usta)
            if cast1.typ = cast2.typ:
                odlisnosti.append(vzdalenost(cast1.hog, cast2.hog))
                # relativni pozice je pozice casti v ramci hlavy.
                if relativni_pozice(cast1) - relativni_pozice(cast2) >
                    max_vzdalenost:
                    pocet_vzdalenych_casti += 1

    prumerna_odlisnost = avg(odlisnosti)
    zvyseni = prumerna_odlisnost / 20
    pocet_zvyseni = pocet_vzdalenych_casti + pocet_odlisnych_casti(hlava1,
        hlava2)
    vysledek = prumerna_odlisnost + zvyseni * pocet_zvyseni

    return vysledek
```

Pseudokód 4.3: Porovnání hlav

4.4 Trénink detektoru

Jelikož nalézt vhodné detektory pro všechny potřebné části ve všech pozicích není jednoduché, byl jsem nucen se pokusit vytrénovat vlastní detektor pro detekci nosu z profilu.

Pro správný trénink je potřeba zajistit nejlépe několik tisíc obrázků objektu, který chceme nalézt a objekt v každém z nich správně zaznačit. Dále je potřeba ještě mnohem více obrázků, které hledaný objekt neobsahují. Kvůli přílišné časové náročnosti zajištění takového množství obrázků jsem se pokusil pro účely demonstrace práce vytrénovat detektor pouze s použitím obrázků, které budou později použity k testování, s tím že jsem doufal, že právě na tyto obrázky bude detektor bez problémů fungovat. OpenCV má na takový

trénink spoustu nástrojů a nakonec se mi podařilo detektor na bázi LBP 2.1.3 vytrénovat. Detektor však detekoval spoustu falešně pozitivních nosů a navíc se mi nakonec podařilo nalézt jiný funkční detektor, proto jsem jej ve výsledném projektu nepoužil

Kapitola 5

Testování

V této kapitole představím provedené experimenty, jejich výsledky a případné odchylky od cílů vytyčených v kapitole 3.

5.1 Popis datové sady

Datová sada, na které jsou prováděny všechny testy se skládá z fotografií pěti členů mé rodiny. Pro zkrácení a anonymizaci jsou fotografie pojmenovány čísly. Význam čísel lze nalézt v tabulce 5.1.

Fotografie ostatních členů rodiny, kromě mě nechci v textu této práce zveřejňovat, musím však napsat, že jsme si s otcem a bratry velice podobní, což algoritmu stěžuje naše rozpoznání.

Každý člen rodiny je vyfocen ve třech různých pozicích. Pozice obsahují pohled přímo do kamery, pohled lehce vpravo vzhůru a pohled přímo doprava. Pro zkrácení budou dále tyto pozice pojmenovány: "čelní", "šikmo vzhůru"a "boční". Každý člen rodiny je v každé pozici vyfotografován dvakrát. Jednou ve vyšším rozlišení venku se žlutou omítkou na pozadí a jednou v nižším rozlišení uvnitř s modrou zdí na pozadí. Rozdíly v rozlišení, světelných podmínkách a pozadí chci nasimulovat rozdíly mezi fotografií a obrázkem vygenerovaným z 3D modelu. Porovnání fotografie s obrázkem hlavy vygenerovaným z 3D modelu by totiž mělo být finální využití této práce. Ukázku pozic a míst si lze prohlédnout na fotografiích 5.1. Pro identifikaci fotografií v grafech, tabulkách a následujícím textu budu používat následující notaci: místo-pozice-osoba, tedy například venku-čelní-1.

5.2 Popis experimentů

Pro všechny experimenty budou použity fotografie popsané výše. Jelikož je algoritmus navržen pro porovnání dvou hlav ve stejné poloze, budou vždy porovnávány všechny osoby v

Tabulka 5.1: Číslování osob na fotografiích

1	já
2	otec
3	matka
4	prostřední bratr
5	nejmladší bratr

Obrázek 5.1: **Ukázka všech pozic a míst použitých ve všech následujících experimentech** Fotografie na prvním řádku jsou všechny vyfoceny venku, na řádku druhém jsou fotografie vyfoceny uvnitř. Ve sloupci vlevo je ukázka pozice "čelní", uprostřed je pozice "šikmo vzhůru"a vpravo je pozice "boční".

určité pozici na určitém místě se všemi osobami ve stejné pozici ve druhém místě. Cílem je, aby algoritmus správně určil dvojice fotografií stejné osoby na různých místech, tedy aby vypočítaná rozdílnost hlav byla nejmenší pro dvě fotografie stejné osoby a rozdílnost fotografií různých osob byla větší.

Výstupem této práce je ceklem 5 algoritmů pro porovnání hlav, jelikož je však jejich popis příliš složitý, než aby se dal použít pro označení jednotlivých algoritmů v hlavičkách tabulek s výsledky, budou v následujících experimentech algoritmy označeny písmeny A - E. Algoritmus A porovnává pouze celé detekované oblasti hlav. Algoritmus B porovnává jednotlivé části hlav mezi sebou a poté vypočítá průměr jejich rozdílností, což je výsledkem tohoto algoritmu. Algoritmus C je stejný jako algoritmus B, ale pro každou část, která byla detekována pouze na jedné z hlav přičte dalších 5% k výsledné rozdílnosti. Algoritmus D je stejný, jako algoritmus B, ale pro každou část, u které se poloha jejího středu liší o více než 3% maximální možné vzdálenosti částí hlavy, přičte dalších 5% k výsledné rozdílnosti. Maximální možnou vzdáleností částí hlavy se rozumí délka úhlopříčky obdélníku ohraničujícího detekovanou oblast hlavy. Algoritmus E je kombinací algoritmu C a algoritmu D, tedy je vypočítána průměrná odlišnost částí hlav a k té je přičteno dalších 5% za každou část, která je detekovaná pouze na jedné z hlav a za každou část, jejíž poloha se mezi hlavami příliš liší.

Většina algoritmů pro detekci tváří a jejich následné porovnávání pracuje s tvářemi hledícími přímo do kamery. Tato práce se zaměřuje i na hlavy v jiných pozicích, avšak

Tabulka 5.2: **Experiment v pozici čelní**

fotografie	A	B	C	D	E
venku-čelní-1	5	4	4	3	3
venku-čelní-2	1	4	4	1	1
venku-čelní-3	4	1	1	1	1
venku-čelní-4	1	1	1	1	1
venku-čelní-5	1	1	1	1	1
uvnitř-čelní-1	4	1	1	1	1
uvnitř-čelní-2	1	1	1	1	1
uvnitř-čelní-3	5	5	5	5	5
uvnitř-čelní-4	1	1	1	1	1
uvnitř-čelní-5	1	1	1	1	1

většina použitých detektorů stále pracuje s hlavami hledícími přímo do kamery. Proto jako první provedu experiment v poloze "čelní", u kterého očekávám nejlepší výsledky. Následovat bude stejný experiment v poloze "šikmo vzhůru"a nakonec poloha "boční", u které může být výsledek o něco horší.

5.3 Experimenty

V této části budou ukázány výsledky provedených experimentů a budou detailněji rozebrány případné překvapivé výsledky. Čísla v tabulce uvádějí, kolikátá nejpodobnější je podle daného algoritmu fotografie stejného člověka ale v druhém místě. Nejlepším možným výsledkem je tedy číslo 1, které označuje vysokou podobnost hlav a nejhorším je číslo 5, které uvádí, že i přes to že je na obou fotografiích stejná osoba, algoritmus hlavy na těchto fotografiích označil jako nejodlišnější. Přesné hodnoty vypočítané algoritmem lze nalézt v příloze B.

Pozice čelní 5.2

U fotografie venku-čelní-1 a uvnitř-čelní-3 se porovnání nezdařilo. U fotografie venku-čelní-3 se sice nepodařila detekovat ústa, avšak to by nemělo způsobit tak špatný konečný výsledek. V těchto dvou případech algoritmus fungoval špatně.

Pozice šikmo vzhůru 5.3

V tomto experimentu se objevilo několik zajímavých výsledků. Algoritmus u osob 1 a 2 zcela selhal, protože u fotografií uvnitř-šikmo vzhůru-1 a uvnitř-šikmo vzhůru-2 se ani jednomu způsobu detekce nepodařilo správně detekovat hlavu a tak algoritmus pracoval s falešně pozitivní detekcí v oblasti oblečení. Proto u algoritmů B, C, D, E není doplněno pořadí. U algoritmu A pořadí sice je, ale s podobností hlav nemá nic společného.

U osoby 4 lze vidět postupné zhoršování mezi algoritmy B, C, D a E i přes to, že přidané prvky každého algoritmu by měly vést naopak spíše ke zlepšení porovnávání. Toto zhoršování je bohužel chybou špatně nafocených fotografií. Algoritmus totiž očekává hlavy vždy ve stejné pozici, ale i přes to že je pozice z pohledu lidského oka velice podobná, je z pohledu algoritmu velice rozdílná. Na fotografii uvnitř-šikmo vzhůru-4 algoritmus totiž úspěšně detekoval obě oči, ucho a nos, zatímco na stejné fotografii venku je hlava otočená o

Tabulka 5.3: **Experiment v pozici šikmo vzhůru**

fotografie	A	B	C	D	E
venku-šikmo vzhůru-1	5	0	0	0	0
venku-šikmo vzhůru-2	3	0	0	0	0
venku-šikmo vzhůru-3	1	1	1	1	1
venku-šikmo vzhůru-4	2	2	3	3	3
venku-šikmo vzhůru-5	1	1	1	1	1
uvnitř-šikmo vzhůru-1	1	0	0	0	0
uvnitř-šikmo vzhůru-2	5	0	0	0	0
uvnitř-šikmo vzhůru-3	1	1	1	1	1
uvnitř-šikmo vzhůru-4	3	1	2	2	3
uvnitř-šikmo vzhůru-5	1	2	3	2	3

Tabulka 5.4: **Experiment v pozici boční**

fotografie	A	B	C	D	E
venku-boční-1	4	3	3	3	3
venku-boční-2	1	0	0	0	0
venku-boční-3	1	1	1	1	1
venku-boční-4	1	1	1	1	1
venku-boční-5	4	1	1	1	1
uvnitř-boční-1	3	1	1	1	1
uvnitř-boční-2	1	0	0	0	0
uvnitř-boční-3	1	1	1	1	1
uvnitř-boční-4	2	1	1	1	1
uvnitř-boční-5	4	1	1	1	1

trochu jinak a nosem si zakrývá jedno oko. Z pohledu algoritmu je to tedy úplně jiná poloha. Pokud by byla jedna z fotografií vygenerována automaticky z 3D modelu osoby, tato chyba by se nemohla stát, protože právě pozice očí je často využívána pro určení polohy hlavy.

Pozice boční 5.4

U této pozice si algoritmus nečekaně vedl daleko lépe, než u předchozí pozice. Jediný problém je osoba 2, u níž se sice na obou fotografiích podařila detekovat hlava, avšak z obou hlav se podařila detekovat pouze jedna část a to oko na fotografii venku. Jelikož však uvnitř nebyla detekována žádná část, nebylo oko s čím porovnat a tudíž pro tuto osobu není k dispozici žádný výsledek, kromě algoritmu A.

5.4 Zhodnocení experimentů

Algoritmus si v experimentech vedl docela dobře. Nastalo sice pár problémů se správnou detekcí, ale jakmile byla hlava a její části správně detekována a zároveň nedošlo k chybě při fotografování, takže byla hlava na obou fotografiích opravdu ve stejné pozici, tak porovnání fungovalo většinou dobře. Z provedených experimentů jde vidět, že pouhé porovnání celé oblasti detekované hlavy (algoritmus A) k identifikaci osob nestačí. Avšak ostatní algoritmy, hlavně algoritmus E většinou dokázaly správně k sobě přiřadit dvě fotografie stejné osoby.

V kapitole 3 jsem uvedl, že chci dosáhnout alespoň 80% úspěšnosti detekce hlavy. Ze 30 fotografií se hlava podařila úspěšně detekovat na 28 fotografiích, což je úspěšnost více než 90%. Dále jsem chtěl vždy detekovat alespoň 50% viditelných částí hlavy, což se kromě osoby 2 v pozici boční také podařilo. Jako poslední kritérium bylo, aby při porovnávání vždy byla hledaná fotografie v 50% nejpodobnějších fotografií. Když pomineme ty 4 fotografie, na kterých detekce nefungovala správně a 2 fotografie, které měly být k těmto fotografiím přiřazeny, tak je v algoritmu E pouze jedna fotografie, která toto kritérium nesplnila a byla 5., dále je tady 5 fotografií, které při porovnávání skončily 3. Na druhou stranu ale všechny ostatní fotografie jsou 1. I přes mírné nesplnění jednoho z kritérií jsem tedy s výsledky experimentů spokojen.

Výsledky algoritmu by dále šly zlepšit použitím ještě lepších detektorů. Díky vyšší úspěšnosti detekce by algoritmus měl více částí hlavy k porovnání. Pokud by detektory pro stejného člověka ve stejné pozici byly více konzistentní v tom, které části hlavy jsou u něj schopny detekovat, mohlo by se v algoritmu C a následně také v algoritmu E přičítat více než 5% k odlišnosti obličejů a tím by se zpřesnilo porovnání. Pokud by detektory pro stejného člověka pro každou část detekovaly vždy stejnou oblast hlavy, mohla by se v algoritmu D a následně v algoritmu E snížit tolerance pro rozdíl pozic částí hlavy a navýšit množství navýšení odlišnosti.

Kapitola 6

Závěr

Účelem práce bylo vytvořit algoritmus, který bude schopen porovnávat hlavy lidí v různém náklonu, natočení a sklonu. Vstupem do algoritmu by měla být jedna fotografie z reálného světa, například z bezpečnostní kamery. Další na vstupu by měl být obrázek hlavy ve stejné pozici (stejný náklon, natočení a sklon) vygenerovaný z 3D modelu člověka. Algoritmus se měl z důvodu zdlouhavosti tréninku neuronové sítě zaměřit na řešení bez neuronových sítí.

V praxi by pak tento algoritmus při dostatečné databázi 3D modelů lidí mohl umožnit například identifikaci osob zachycených při kriminální činnosti bezpečnostní kamerou. Dalším použitím by mohlo být vyhledávání osob v případě, kdy by byl dostupný 3D model hledané osoby, bylo by možné jej použít pro neustálé porovnávání se záběry z bezpečnostních kamer a tím určit pozici hledané osoby v okamžiku, kdy ji zachytí některá z bezpečnostních kamer. Výhodou tohoto algoritmu měla být jeho schopnost porovnávat hlavy ve více pozicích, než většina dosavadních řešení pro identifikaci osob pomocí počítačového vidění, která se soustřeďují hlavně na porovnávání obličejů hledících téměř přímo do kamery.

Na začátku práce je představena řada v současnosti využívaných algoritmů určených pro detekci a porovnání tváří a objektů. Mezi významné algoritmy z této sekce, které jsou v práci využity patří výpočet a následné použití histogramu orientovaných gradientů, který lze poté využít k detekci objektů a také ke zjištění jejich rozdílnosti. V této práci je použit pro detekci hlav a je stěžejním i pro porovnávání hlav, protože právě histogramy orientovaných gradientů jsou využity ve všech porovnávacích algoritmech v této práci. Dalším důležitým algoritmem je Viola-Jones algoritmus, který slouží pro detekci objektů. V této práci je využit pro detekci obličeje hledícího přímo do kamery, pro detekci obličeje z profilu a pro detekci částí obličeje (oči, uši, nos, ústa). Posledním důležitým představeným algoritmem je "Použití nepravidelností obličeje pro zpřesnění rozpoznávání", který používá různé výrazné oblasti na kůži v obličeji, jako jsou znaménka, jizvy nebo tetování pro zpřesnění jiných metod rozpoznávání osob z fotografií. Porovnání v této práci je tímto algoritmem inspirováno, místo jizev, znamének nebo tetování jsou však využity větší části hlavy jako jsou uši, oči, nos nebo ústa.

Dále je podrobně popsán návrh řešení práce, očekávaná úskalí a jsou stanoveny minimální požadavky na funkčnost algoritmu. Následně je popsán postup implementace včetně zvoleného jazyka a všech knihoven.

Jako poslední jsou provedeny experimenty na fotografiích lidí z mé rodiny. Je vyhodnocena funkčnost výsledného řešení a je srovnána s požadavky z předchozích kapitol. Na závěr je vypsáno několik nápadů na další zlepšení výsledků algoritmu.

Práce mě naučila mnoho z oblasti počítačového vidění, ve které jsem před tím neměl žádné znalosti. Dala mi příležitost k prohloubení mých znalostí programování v jazyce

Python, se kterým jsem se do té doby bohužel setkal pouze několikrát. Nakonec mi práce ukázala, že napsat text o rozsahu bakalářské, nebo jiné podobné práce není vůbec tak jednoduché, jak se může zdát.

Literatura

[1] Becerra Riera, F., Morales González, A. a Méndez Vázquez, H. Facial marks for improving face recognition. *Pattern Recognition Letters*. Elsevier. 2018, sv. 113, s. 3–9.

[2] Castrillón Santana, M., Déniz Suárez, O., Hernández Tejera, M. a Guerra Artal, C. ENCARA2: Real-time Detection of Multiple Faces at Different Resolutions in Video Streams. *Journal of Visual Communication and Image Representation*. April 2007, s. 130–140.

[3] Cootes, T. F., Taylor, C. J., Cooper, D. H. a Graham, J. Active shape models-their training and application. *Computer vision and image understanding*. Elsevier. 1995, sv. 61, č. 1, s. 38–59.

[4] Dalal, N. a Triggs, B. Histograms of oriented gradients for human detection. In:. 2005.

[5] Free Software Foundation, Inc.. *What is a shell?* [online]. [cit. 30. července 2020]. Dostupné z: https://www.gnu.org/software/bash/manual/html_node/What-is-a-shell_003f.html#What-is-a-shell_003f.

[6] Gupta, V. Face Detection – OpenCV, Dlib and Deep Learning (C++ / Python). *Learn OpenCV* [online]. 2018. Aktualizováno 22. 10. 2018 [cit. 2. ledna 2020]. Dostupné z: https://www.learnopencv.com/face-detection-opencv-dlib-and-deep-learning-c-python/.

[7] Hunter, J. D. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*. IEEE COMPUTER SOC. 2007, sv. 9, č. 3, s. 90–95. DOI: 10.1109/MCSE.2007.55.

[8] King, D. E. Dlib-ml: A Machine Learning Toolkit. *Journal of Machine Learning Research*. 2009, sv. 10, s. 1755–1758.

[9] Liao, Z., Zhou, P., Wu, Q. a Ni, B. Uniface: A Unified Network for Face Detection and Recognition. In: IEEE. *2018 24th International Conference on Pattern Recognition (ICPR)*. 2018, s. 3531–3536.

[10] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S. et al. Ssd: Single shot multibox detector. In: Springer. *European conference on computer vision*. 2016, s. 21–37.

[11] Lowe, D. G. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*. Springer. 2004, sv. 60, č. 2, s. 91–110.

[12] Marr, D. a Hildreth, E. Theory of edge detection. *Proceedings of the Royal Society of London. Series B. Biological Sciences*. The Royal Society London. 1980, sv. 207, č. 1167, s. 187–217.

[13] Méndez, N., Chang, L., Plasencia Calana, Y. a Méndez Vázquez, H. Facial landmarks detection using extended profile lbp-based active shape models. In: Springer. *Iberoamerican Congress on Pattern Recognition*. 2013, s. 407–414.

[14] Ojala, T., Pietikäinen, M. a Mäenpää, T. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis & Machine Intelligence*. IEEE. 2002, č. 7, s. 971–987.

[15] OpenCV team. *About* [online]. 2020 [cit. 30. července 2020]. Dostupné z: https://opencv.org/about/.

[16] Python Software Foundation. *The Python Tutorial* [online]. 2020. Aktualizováno 30. 7. 2020 [cit. 30. července 2020]. Dostupné z: https://docs.python.org/3/tutorial/index.html.

[17] The SciPy community. *About NumPy* [online]. 2020. Aktualizováno 29. 6. 2020 [cit. 30. července 2020]. Dostupné z: https://numpy.org/doc/stable/about.html.

[18] Viola, P., Jones, M. et al. Rapid object detection using a boosted cascade of simple features. *CVPR (1)*. 2001, sv. 1, 511-518, s. 3.

[19] Yang, C., Wei, H. a Yu, Q. A novel method for 2D nonrigid partial shape matching. *Neurocomputing*. Elsevier. 2018, sv. 275, s. 1160–1176.

[20] Zhou, D., Petrovska Delacrétaz, D. a Dorizzi, B. Automatic landmark location with a combined active shape model. In: IEEE. *2009 IEEE 3rd International Conference on Biometrics: Theory, Applications, and Systems*. 2009, s. 1–7.

Příloha A

Úspěšnost detekce

V této příloze je uvedena tabulka, ve které jsou zaznamenány výsledky detekce bez využití neuronové sítě pro detekci hlavy. Jednotlivé fotografie jsou pojmenovány podle vzoru místo-pozice-osoba stejně jako v kapitole 5. Čísla v buňkách tabulky udávají počet detekovaných částí. Při falešně pozitivní detekci je v tabulce uvedeno číslo se znaménkem mínus.

Tabulka A.1: Úpěšnost detekce

Fotografie	Hlava	Oči	Uši	Nos	Ústa
venku-čelní-1	1	2	0	1	1
venku-čelní-2	1	2	0	1	1
venku-čelní-3	1	2	0	1	0
venku-čelní-4	1	2	0	1	1
venku-čelní-5	1	2	0	1	1
venku-šikmo vzhůru-1	1	1	1	1	1
venku-šikmo vzhůru-2	1	1	0	1	1
venku-šikmo vzhůru-3	1	1	1	0	0
venku-šikmo vzhůru-4	1	1	1	0	-1
venku-šikmo vzhůru-5	1	1	1	0	1
venku-boční-1	1	1	1	1	0
venku-boční-2	1	1	0	0	0
venku-boční-3	1	1	1	1	0
venku-boční-4	1	1	1	1	0
venku-boční-5	1	1	1	0	0
uvnitř-čelní-1	1	2	0	1	1
uvnitř-čelní-2	1	2	0	1	1
uvnitř-čelní-3	1	2	0	1	1
uvnitř-čelní-4	1	2	0	1	1
uvnitř-čelní-5	1	2	0	1	1
uvnitř-šikmo vzhůru-1	-1	0	0	0	0
uvnitř-šikmo vzhůru-2	-1	0	0	0	0
uvnitř-šikmo vzhůru-3	1	1	1	0	0
uvnitř-šikmo vzhůru-4	1	2	1	1	0
uvnitř-šikmo vzhůru-5	1	1	1	1	0
uvnitř-boční-1	1	1	1	0	0
uvnitř-boční-2	1	0	0	0	0
uvnitř-boční-3	1	1	1	0	0
uvnitř-boční-4	1	1	1	0	0
uvnitř-boční-5	1	1	1	1	1

Příloha B

Výsledky porovnání

Tato příloha obsahuje detailní výsledky porovnání všech fotografií. Tyto výsledky jsou shrnuty v kapitole 5. Pojmenování algoritmů v záhlaví je stejné jako v kapitole 5, tudíž nejdůležitější hodnotou je algoritmus E, který je kombinací předešlých algoritmů. Hodnoty znamenají rozdílnost hlav. Čím menší hodnota, tím podobnější hlavy na fotografiích jsou. Hodnota 0 znamená stejnou fotografii. Maximální možná hodnota pro toto porovnání nemá význam (lze ji vypočítat, avšak při porovnání 2 hlav se k ní nelze ani řádově přiblížit). Pojmenování fotografií je podle vzoru místo-pozice-osoba stejně jako v kapitole 5, v tabulkách je pro ušetření místa pozice vynechána (je vždy stejná pro celou sekci tabulek).

B.1 Pozice čelní

Tabulka B.1: **venku-čelní-1**

foto	A	B	C	D	E	počet částí detekovaných pouze na jedné hlavě	počet částí s příliš odlišnou pozicí
uvnitř-1	31,033	5,990	5,990	5,990	5,990	0	0
uvnitř-2	29,650	6,051	6,051	6,051	6,051	0	0
uvnitř-3	27,845	5,399	5,399	5,669	5,669	0	1
uvnitř-4	29,199	5,587	5,587	5,866	5,866	0	1
uvnitř-5	29,278	5,970	5,970	6,268	6,268	0	1

Tabulka B.2: **venku-čelní-2**

foto	A	B	C	D	E	počet částí detekovaných pouze na jedné hlavě	počet částí s příliš odlišnou pozicí
uvnitř-1	29,502	6,268	6,268	6,581	6,581	0	1
uvnitř-2	27,706	5,539	5,539	5,539	5,539	0	0
uvnitř-3	28,978	5,425	5,425	5,696	5,696	0	1
uvnitř-4	28,274	5,463	5,463	5,736	5,736	0	1
uvnitř-5	29,212	5,410	5,410	5,680	5,680	0	1

Tabulka B.3: **venku-čelní-3**

foto	A	B	C	D	E	počet částí detekovaných pouze na jedné hlavě	počet částí s příliš odlišnou pozicí
uvnitř-1	32,138	7,122	7,478	8,190	8,547	1	3
uvnitř-2	31,735	6,302	6,617	7,247	7,562	1	3
uvnitř-3	32,911	5,917	6,212	6,804	7,100	1	3
uvnitř-4	33,033	6,290	6,605	7,234	7,548	1	3
uvnitř-5	31,565	6,468	6,791	7,438	7,762	1	3

Tabulka B.4: **venku-čelní-4**

foto	A	B	C	D	E	počet částí detekovaných pouze na jedné hlavě	počet částí s příliš odlišnou pozicí
uvnitř-1	30,466	6,030	6,030	6,331	6,331	0	1
uvnitř-2	29,749	5,845	5,845	5,845	5,845	0	0
uvnitř-3	28,747	5,457	5,457	5,729	5,729	0	1
uvnitř-4	27,185	4,846	4,846	4,846	4,846	0	0
uvnitř-5	29,162	5,843	5,843	6,135	6,135	0	1

Tabulka B.5: **venku-čelní-5**

foto	A	B	C	D	E	počet částí detekovaných pouze na jedné hlavě	počet částí s příliš odlišnou pozicí
uvnitř-1	29,556	6,027	6,027	6,328	6,328	0	1
uvnitř-2	30,114	5,781	5,781	6,070	6,070	0	1
uvnitř-3	30,800	5,674	5,674	5,958	5,958	0	1
uvnitř-4	28,561	5,318	5,318	5,850	5,850	0	2
uvnitř-5	24,931	5,054	5,054	5,054	5,054	0	0

Tabulka B.6: **uvnitř-čelní-1**

foto	A	B	C	D	E	počet částí detekovaných pouze na jedné hlavě	počet částí s příliš odlišnou pozicí
venku-1	31,033	5,990	5,990	5,990	5,990	0	0
venku-2	29,502	6,268	6,268	6,581	6,581	0	1
venku-3	32,138	7,122	7,478	8,190	8,547	1	3
venku-4	30,466	6,030	6,030	6,331	6,331	0	1
venku-5	29,556	6,027	6,027	6,328	6,328	0	1

Tabulka B.7: **uvnitř-čelní-2**

foto	A	B	C	D	E	počet částí detekovaných pouze na jedné hlavě	počet částí s příliš odlišnou pozicí
venku-1	29,650	6,051	6,051	6,051	6,051	0	0
venku-2	27,706	5,539	5,539	5,539	5,539	0	0
venku-3	31,735	6,302	6,617	7,247	7,562	1	3
venku-4	29,749	5,845	5,845	5,845	5,845	0	0
venku-5	30,114	5,781	5,781	6,070	6,070	0	1

Tabulka B.8: **uvnitř-čelní-3**

foto	A	B	C	D	E	počet částí detekovaných pouze na jedné hlavě	počet částí s příliš odlišnou pozicí
venku-1	27,845	5,399	5,399	5,669	5,669	0	1
venku-2	28,978	5,425	5,425	5,696	5,696	0	1
venku-3	32,911	5,917	6,212	6,804	7,100	1	3
venku-4	28,747	5,457	5,457	5,729	5,729	0	1
venku-5	30,800	5,674	5,674	5,958	5,958	0	1

Tabulka B.9: **uvnitř-čelní-4**

foto	A	B	C	D	E	počet částí detekovaných pouze na jedné hlavě	počet částí s příliš odlišnou pozicí
venku-1	29,199	5,587	5,587	5,866	5,866	0	1
venku-2	28,274	5,463	5,463	5,736	5,736	0	1
venku-3	33,033	6,290	6,605	7,234	7,548	1	3
venku-4	27,185	4,846	4,846	4,846	4,846	0	0
venku-5	28,561	5,318	5,318	5,850	5,850	0	2

Tabulka B.10: **uvnitř-čelní-5**

foto	A	B	C	D	E	počet částí detekovaných pouze na jedné hlavě	počet částí s příliš odlišnou pozicí
venku-1	29,278	5,970	5,970	6,268	6,268	0	1
venku-2	29,212	5,410	5,410	5,680	5,680	0	1
venku-3	31,565	6,468	6,791	7,438	7,762	1	3
venku-4	29,162	5,843	5,843	6,135	6,135	0	1
venku-5	24,931	5,054	5,054	5,054	5,054	0	0

B.2 Pozice šikmo vzhůru

Tabulka B.11: **venku-šikmo vzhůru-1**

foto	A	B	C	D	E	počet částí detekovaných pouze na jedné hlavě	počet částí s příliš odlišnou pozicí
uvnitř-1	Detekce se nezdařila						
uvnitř-2	Detekce se nezdařila						
uvnitř-3	31,800	6,205	6,825	6,825	7,446	2	2
uvnitř-4	29,651	6,299	6,929	6,614	7,244	2	1
uvnitř-5	29,182	5,813	6,103	6,103	6,394	1	1

Tabulka B.12: **venku-šikmo vzhůru-2**

foto	A	B	C	D	E	počet částí detekovaných pouze na jedné hlavě	počet částí s příliš odlišnou pozicí
uvnitř-1	Detekce se nezdařila						
uvnitř-2	Detekce se nezdařila						
uvnitř-3	Detekce nedostatečná					5	
uvnitř-4	34,509	6,983	8,030	7,332	8,379	3	1
uvnitř-5	33,713	4,991	5,989	4,991	5,989	4	0

Tabulka B.13: **venku-šikmo vzhůru-3**

foto	A	B	C	D	E	počet částí detekovaných pouze na jedné hlavě	počet částí s příliš odlišnou pozicí
uvnitř-1	Detekce se nezdařila						
uvnitř-2	Detekce se nezdařila						
uvnitř-3	27,743	5,353	5,353	5,353	5,353	0	0
uvnitř-4	31,931	6,080	6,688	6,384	6,992	2	1
uvnitř-5	31,799	5,598	5,877	5,877	6,157	1	1

Tabulka B.14: **venku-šikmo vzhůru-4**

foto	A	B	C	D	E	počet částí detekovaných pouze na jedné hlavě	počet částí s příliš odlišnou pozicí
uvnitř-1	Detekce se nezdařila						
uvnitř-2	Detekce se nezdařila						
uvnitř-3	29,028	6,001	6,301	6,301	6,601	1	1
uvnitř-4	31,691	5,927	6,816	6,520	7,409	3	2
uvnitř-5	32,042	5,674	6,241	6,241	6,809	2	2

Tabulka B.15: **venku-šikmo vzhůru-5**

foto	A	B	C	D	E	počet částí detekovaných pouze na jedné hlavě	počet částí s příliš odlišnou pozicí
uvnitř-1	Detekce se nezdařila						
uvnitř-2	Detekce se nezdařila						
uvnitř-3	31,640	6,451	6,773	7,096	7,418	1	2
uvnitř-4	31,044	6,640	7,636	6,972	7,969	3	1
uvnitř-5	28,006	5,516	6,068	5,792	6,343	2	1

Tabulka B.16: **uvnitř-šikmo vzhůru-1**

foto	A	B	C	D	E	počet částí detekovaných pouze na jedné hlavě	počet částí s příliš odlišnou pozicí
venku-1 venku-2 venku-3 venku-4 venku-5	Detekce se nezdařila						

Tabulka B.17: **uvnitř-šikmo vzhůru-2**

foto	A	B	C	D	E	počet částí detekovaných pouze na jedné hlavě	počet částí s příliš odlišnou pozicí
venku-1 venku-2 venku-3 venku-4 venku-5	Detekce se nezdařila						

Tabulka B.18: **uvnitř-šikmo vzhůru-3**

foto	A	B	C	D	E	počet částí detekovaných pouze na jedné hlavě	počet částí s příliš odlišnou pozicí
venku-1	31,800	6,205	6,825	6,825	7,446	2	2
venku-2	Detekce nedostatečná					5	
venku-3	27,743	5,353	5,353	5,353	5,353	0	0
venku-4	29,028	6,001	6,301	6,301	6,601	1	1
venku-5	31,640	6,451	6,773	7,096	7,418	1	2

Tabulka B.19: **uvnitř-šikmo vzhůru-4**

foto	A	B	C	D	E	počet částí detekovaných pouze na jedné hlavě	počet částí s příliš odlišnou pozicí
venku-1	29,651	6,299	6,929	6,614	7,244	2	1
venku-2	34,509	6,983	8,030	7,332	8,379	3	1
venku-3	31,931	6,080	6,688	6,384	6,992	2	1
venku-4	31,691	5,927	6,816	6,520	7,409	3	2
venku-5	31,044	6,640	7,636	6,972	7,969	3	1

Tabulka B.20: **uvnitř-šikmo vzhůru-5**

foto	A	B	C	D	E	počet částí detekovaných pouze na jedné hlavě	počet částí s příliš odlišnou pozicí
venku-1	29,182	5,813	6,103	6,103	6,394	1	1
venku-2	33,713	4,991	5,989	4,991	5,989	4	0
venku-3	31,799	5,598	5,877	5,877	6,157	1	1
venku-4	32,042	5,674	6,241	6,241	6,809	2	2
venku-5	28,006	5,516	6,068	5,792	6,343	2	1

B.3 Pozice boční

Tabulka B.21: **venku-boční-1**

foto	A	B	C	D	E	počet částí detekovaných pouze na jedné hlavě	počet částí s příliš odlišnou pozicí
uvnitř-1	31,298	5,723	6,009	6,295	6,582	1	2
uvnitř-2	Detekce se nezdařila						
uvnitř-3	29,611	6,314	6,630	6,946	7,262	1	2
uvnitř-4	29,696	5,708	5,994	6,279	6,565	1	2
uvnitř-5	31,105	5,533	5,809	6,086	6,363	1	2

Tabulka B.22: **venku-boční-2**

foto	A	B	C	D	E	počet částí detekovaných pouze na jedné hlavě	počet částí s příliš odlišnou pozicí
uvnitř-1	Detekce se nezdařila						
uvnitř-2	Detekce se nezdařila						
uvnitř-3	Detekce se nezdařila						
uvnitř-4	Detekce se nezdařila						
uvnitř-5	33,587	6,281	7,224	6,281	7,224	3	0

Tabulka B.23: **venku-boční-3**

foto	A	B	C	D	E	počet částí detekovaných pouze na jedné hlavě	počet částí s příliš odlišnou pozicí
uvnitř-1	29,635	6,809	7,149	7,149	7,490	1	1
uvnitř-2	Detekce se nezdařila						
uvnitř-3	25,976	5,640	5,922	5,922	6,204	1	1
uvnitř-4	29,900	6,132	6,439	6,439	6,746	1	1
uvnitř-5	31,821	6,162	6,470	6,778	7,087	1	2

Tabulka B.24: **venku-boční-4**

foto	A	B	C	D	E	počet částí detekovaných pouze na jedné hlavě	počet částí s příliš odlišnou pozicí
uvnitř-1	31,301	6,560	6,888	6,888	7,216	1	1
uvnitř-2	Detekce se nezdařila						
uvnitř-3	30,257	6,483	6,807	6,807	7,131	1	1
uvnitř-4	29,621	5,273	5,537	5,537	5,800	1	1
uvnitř-5	30,828	5,708	5,993	5,993	6,278	1	1

Tabulka B.25: **venku-boční-5**

foto	A	B	C	D	E	počet částí detekovaných pouze na jedné hlavě	počet částí s příliš odlišnou pozicí
uvnitř-1	29,299	6,684	6,684	7,353	7,353	0	2
uvnitř-2	Detekce se nezdařila						
uvnitř-3	30,254	6,786	6,786	7,125	7,125	0	1
uvnitř-4	28,328	6,139	6,139	6,753	6,753	0	2
uvnitř-5	32,266	4,924	5,417	5,170	5,663	2	1

Tabulka B.26: **uvnitř-boční-1**

foto	A	B	C	D	E	počet částí detekovaných pouze na jedné hlavě	počet částí s příliš odlišnou pozicí
venku-1	31,298	5,723	6,009	6,295	6,582	1	2
venku-2	Detekce se nezdařila						
venku-3	29,635	6,809	7,149	7,149	7,490	1	1
venku-4	31,301	6,560	6,888	6,888	7,216	1	1
venku-5	29,299	6,684	6,684	7,353	7,353	0	2

Tabulka B.27: **uvnitř-boční-2**

foto	A	B	C	D	E	počet částí detekovaných pouze na jedné hlavě	počet částí s příliš odlišnou pozicí
venku-1 venku-2 venku-3 venku-4 venku-5	Detekce se nezdařila						

Tabulka B.28: **uvnitř-boční-3**

foto	A	B	C	D	E	počet částí detekovaných pouze na jedné hlavě	počet částí s příliš odlišnou pozicí
venku-1	29,611	6,314	6,630	6,946	7,262	1	2
venku-2	Detekce se nezdařila						
venku-3	25,976	5,640	5,922	5,922	6,204	1	1
venku-4	30,257	6,483	6,807	6,807	7,131	1	1
venku-5	30,254	6,786	6,786	7,125	7,125	0	1

Tabulka B.29: **uvnitř-boční-4**

foto	A	B	C	D	E	počet částí detekovaných pouze na jedné hlavě	počet částí s příliš odlišnou pozicí
venku-1	29,696	5,708	5,994	6,279	6,565	1	2
venku-2	Detekce se nezdařila						
venku-3	29,900	6,132	6,439	6,439	6,746	1	1
venku-4	29,621	5,273	5,537	5,537	5,800	1	1
venku-5	28,328	6,139	6,139	6,753	6,753	0	2

Tabulka B.30: **uvnitř-boční-5**

foto	A	B	C	D	E	počet částí detekovaných pouze na jedné hlavě	počet částí s příliš odlišnou pozicí
venku-1	31,105	5,533	5,809	6,086	6,363	1	2
venku-2	33,587	6,281	7,224	6,281	7,224	3	0
venku-3	31,821	6,162	6,470	6,778	7,087	1	2
venku-4	30,828	5,708	5,993	5,993	6,278	1	1
venku-5	32,266	4,924	5,417	5,170	5,663	2	1

Příloha C

Obsah přiloženého paměťového média

.
|– `projekt/` - složka s projektem k bakalářské práci.
| |– `cascades/` - složka obsahující kaskády použité v projektu.
| |– `data/` - složka obsahující fotografie použité v experimentech.
| |– `documentation/` - složka obsahující vygenerovanou programovou dokumentaci.
| |– `main.py` - hlavní soubor projektu.
| |– `README.txt` - soubor s detailním popisem instalace a spuštění projektu.
| |– `requirements.txt` - soubor s výpisem potřebných knihoven.
| |– `results/` - složka s výsledky experimentů.
| |– `src/` - složka s většinou zdrojových kódů projektu.
| |– `test.sh` - skript, který lze použít pro testování projektu.
| |– `training/` - data pro trénink detektoru včetně vytrénovaného detektoru.
|– `text/` - složka obsahující vše pro vygenerování textu této práce.
|– `xwysog00-algoritmus-porovnavani-hlav-web.pdf` - text práce. Barevné odkazy
|– `xwysog00-algoritmus-porovnavani-hlav-tisk.pdf` - text práce. Černé odkazy