VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

URČENÍ AZIMUTU NATOČENÍ HLAVY V ZÁZNAMU BEZPEČNOSTNÍ KAMEROU

DETERMINING HEAD ROTATION IN VIDEO FROM SECURITY CAMERA

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **BC. ONDŘEJ BLUCHA**
AUTHOR

VEDOUCÍ PRÁCE — **ING. TOMÁŠ GOLDMANN**
SUPERVISOR

BRNO 2017

Zadání diplomové práce/20088/2016/xbluch00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2016/2017

Zadání diplomové práce

Řešitel: **Blucha Ondřej, Bc.**
Obor: Počítačová grafika a multimédia
Téma: **Určení azimutu natočení hlavy v záznamu bezpečnostní kamerou**
Determining Head Rotation in Video from Security Camera
Kategorie: Zpracování obrazu

Pokyny:

1. Prostudujte a sumarizujte vlastnosti běžných bezpečnostních kamer. Dále analyzujte, které algoritmy se používají pro detekci obličeje.
2. Nastudujte, které charakteristické rysy v obličeji můžeme detekovat pomocí metod počítačového vidění. Tyto rysy podrobně popište a zhodnoťte jejich přínos pro určení azimutu natočení hlavy.
3. Navrhněte algoritmus, který dokáže na základě detekovatelných charakteristických rysů určit natočení hlav jednotlivých osob ve video záznamu.
4. Algoritmus implementujte v libovolném programovacím jazyce a vytvořte pro něj jednoduché uživatelské rozhraní (není stěžejní součásti práce).
5. Proveďte experimenty s algoritmem a srovnejte výsledky s datasetem pořízeným na UITS. U chybných výsledku zdůvodněte původ chyby.
6. Navrhněte řešení zjištěných problémů a rozšíření vaší práce.

Literatura:

- Drahanský M., Orság F. et al. *Biometrie*. Brno: Computer Press, s.r.o, 2011. ISBN 978-80-254-8979-6.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Goldmann Tomáš, Ing.**, UITS FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 24. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav inteligentních systémů
612 66 Brno, Božetěchova 2

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

Abstrakt

Tato diplomová práce se zabývá tvorbou aplikace k určení natočení hlavy v záznamu pořízeném bezpečnostní kamerou. Celá aplikace se skládá ze tří částí, a to z detekce obličejů, lokalizace charakteristických bodů v obličeji a ze samotného výpočtu úhlů natočení hlavy. Detekce obličejů byla naimplementována pomocí algoritmů Viola-Jones a HOG. Lokalizace charakteristických bodů v obličeji byla provedena pomocí algoritmu založeného na principu aktivní šablony. Samotný výpočet úhlů natočení hlavy byl proveden pomocí dvou metod. První metoda vychází z antropometrických vlastností hlavy a druhá metoda je založena na natáčení modelů hlavy a hledání správné pozice pomocí algoritmu Perspective-n-Point. Následně byly všechny implementované algoritmy otestovány a bylo nalezeno nejvhodnější nastavení parametrů.

Abstract

This thesis attempts to create an application to determine head rotation angle in video recorded from a security camera. The application consists of three parts: face detection, facial landmarks detection and determination of person's head rotation. The face detection has been implemented using Viola-Jones and HOG algorithms. Facial landmarks detection has been done using algorithm based on active shape model. Two methods to calculate the head rotation angles have been used: the first method works with anthropometric head features. The second method uses Perspective-n-Point algorithm to find the right rotation angles. Finally, all algorithms implemented have been tested and the proper parameters have been determined.

Klíčová slova

bezpečnostní kamerové systémy, CCD, CMOS, detekce obličeje, Viola-Jones, HOG, Active Shape Model, Active Appearance Model, určení natočení hlavy, antropometrie hlavy, Perspective-n-Point

Keywords

CCTV, CCD, CMOS, face detection, Viola-Jones, HOG, Active Shape Model, Active Appearance Model, head rotation assessment, head anthropometry, Perspective-n-Point

Citace

Ondřej Blucha: Určení azimutu natočení hlavy v záznamu bezpečnostní kamerou, diplomová práce, Brno, FIT VUT v Brně, 2017

Určení azimutu natočení hlavy v záznamu bezpečnostní kamerou

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Tomáše Goldmanna.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Ondřej Blucha
24. 5. 2017

Poděkování

Rád bych poděkoval svému vedoucímu diplomové práce Ing. Tomáši Goldmannovi za odbornou pomoc, cenné rady a věnovaný čas při řešení mé diplomové práce.

Obsah

1 Úvod

S bezpečnostními kamerami se v současnosti setkáváme prakticky denně a můžeme je najít téměř na každém rohu, a to jak v exteriérech, tak v interiérech. Staly se již nedílnou součástí našeho života, a že nás snímají na letištích nebo v hypermarketech již nikoho nepřekvapí a bereme to jako samozřejmost. Mnoho lidí používá bezpečnostní kamery i k zabezpečení vlastního domu.

Dnešní bezpečnostní kamerové systémy již nejsou složeny pouze z kamer jako před 30 lety, které dokázaly pouze zobrazit a uložit záznam a nic víc. Dnešní bezpečnostní kamerové systémy již velmi často obsahují i speciální software, který je schopen například detekovat pohyb, spočítat osoby v záběru nebo rozpoznat obličeje. Někdy nám ale prosté rozpoznání obličeje nestačí a bylo by užitečné, kdyby byla kamera schopna rozpoznat i směr natočení hlavy vůči kameře. To znamená, že by byla schopna určit kam, popřípadě na jaký předmět, se daná osoba v daný čas pravděpodobně dívá. A tímto se budu v mém diplomovém projektu zabývat.

Cílem této práce je vytvořit aplikaci, která dokáže určit natočení hlavy ze záznamu pořízeného bezpečnostní kamerou. Aplikace funguje tak, že v každém snímku obrazu jsou detekovány všechny obličeje, následně jsou v každém obličeji lokalizovány charakteristické body a pro každý obličej jsou pak na základě těchto bodů určeny Euklidovy úhly. Výstupem této aplikace jsou pro každý obličej hodnoty tří úhlů.

K detekci obličejů byl použit algoritmus Viola-Jones a algoritmus založený na histogramech orientovaných gradientů (HOG). K lokalizaci charakteristických bodů v obličeji byl použit jeden z algoritmů fungujících na principu aktivní šablony, konkrétně se jedná o algoritmus „One Millisecond Face Alignment with an Ensemble of Regression Trees“. K výslednému určení natočení hlavy byly implementovány dva algoritmy. První algoritmus počítá úhel natočení hlavy na základě antropometrických vlastností hlavy, druhý algoritmus se snaží natáčet model hlavy tak, aby co nejvíce odpovídal hlavě na snímku. Ke hledání správné pozice hlavy byl použit algoritmus Perspective-n-Point (PnP). Na závěr budou výše zmíněné algoritmy otestovány a bude nalezeno nejvhodnější nastavení parametrů.

Tato práce obsahuje celkem 8 kapitol. Následující kapitola se bude zabývat bezpečnostními kamerovými systémy a podrobně zde budou popsány dva dnes nejpoužívanější obrazové senzory, a to CCD a CMOS. Další kapitola se bude zabývat detekcí obličejů a budou zde popsány algoritmy Viola-Jones a HOG. Následně budou popsány algoritmy pro lokalizaci charakteristických bodů v obličeji. Další kapitola se bude zabývat vším, co je potřeba pro samotné určení natočení hlavy vůči kameře a budou zde podrobně popsány oba implementované algoritmy. V šesté kapitole bude popsána implementace aplikace. V předposlední kapitole provedu testování jednotlivých algoritmů a jejich zhodnocení. V poslední kapitole provedu shrnutí celé práce.

Tato diplomová práce navazuje na semestrální projekt, který se zabýval zejména teorií potřebnou k pochopení daného problému a návrhem potřebných algoritmů. Konkrétně se jedná o kapitoly 2, 3 a částečně o kapitolu 4.

2 Bezpečnostní kamerové systémy

Za bezpečnostní kamerový systém obvykle považujeme sadu kamer, které sledují určitý prostor, zobrazují záběry na monitorech a popřípadě je archivují. Někdy bývají i vybaveny softwarem pro detekci podezřelých objektů nebo analýzou obrazových dat.

První kamery jako takové začaly vznikat na konci 19. století. První bezpečnostní systém byl nainstalován v roce 1942 v Německém Peenemünde ve středisku pro vývoj raket V-2 [1]. První město, které zavedlo zabezpečovací kamerový systém k boji s kriminalitou, bylo roku 1964 Olean v americkém státě New York [2]. Od té doby se počet bezpečnostních systémů neustále zvyšuje. Zejména pak přechod k digitálním obrazovým senzorům v 90. letech 20. století přispěl k výraznému růstu počtu bezpečnostních systémů, který trvá až dodnes.

Bezpečnostní kamerové systémy můžeme dělit podle toho, jestli jsou analogové nebo digitální. Analogové kamerové systémy ukládají záznam na videopásku, neumožňují současně nahrávat záznam a sledovat dříve uložený záznam. Digitální kamerové systémy nám umožnují ukládat záznamy na hard disk, SD kartu atd., vysílat digitální signál po síti a případně provádět online detekci podezřelých objektů nebo analyzovat obrazová data. Uložená data jsou digitální, a proto je lze normálně zpracovávat na PC. Nové bezpečnostní kamerové systémy jsou téměř výhradně digitální [3]. Digitální kamery mají nejčastěji obrazové senzory CCD nebo CMOS. Samotné bezpečnostní kamery pak lze rozdělit podle toho, jestli jsou pouze statické anebo je lze natáčet a mají zoom, takové kamery se pak obvykle označují jako PTZ (pan-tilt-zoom cameras – kamera s otáčením-nakláněním-zoomem).

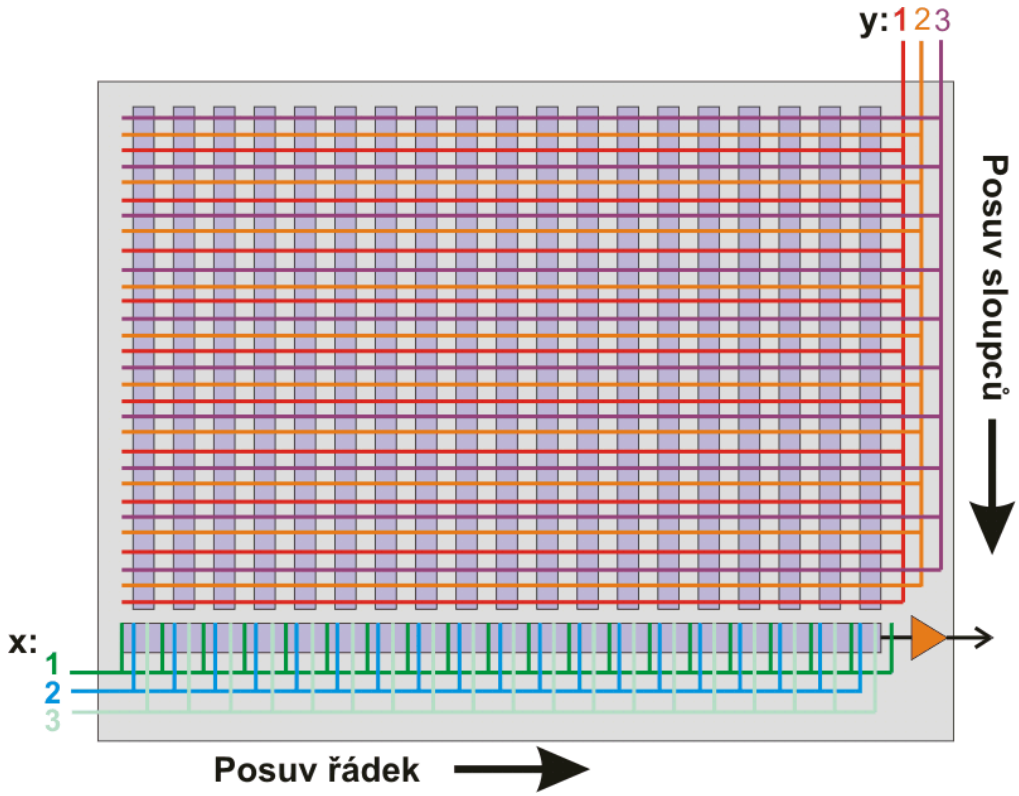
Mezi hlavní komponenty kamer, které mají největší vliv na kvalitu výsledného záznamu, patří obrazové senzory. V minulosti se používala takzvaná vidikonová trubice, dnes se téměř výhradně používají dva typy senzorů, a to CCD (podkapitola 2.1.1) a CMOS (podkapitola 2.1.2) a tyto senzory budou popsány v následujících podkapitolách.

2.1 Technologie CCD

Charge coupled device (zařízení s vázanými náboji) je digitální obrazový senzor. Vynalezli jej v roce 1969 Willard Boyle a George E. Smith v Bellových laboratořích [4], kteří za tento vynález obdrželi roku 2009 Nobelovu cenu za fyziku [5].

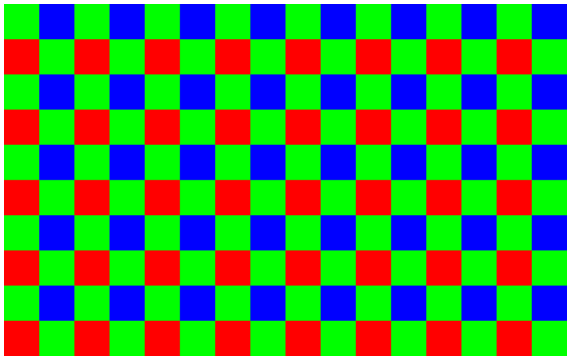
Hlavní část CCD čipu je světlocitlivý křemíkový plátek uložený nad kovovou elektrodou, která je izolována vrstvou oxidu křemičitého (SiO_2). Snímání obrazu funguje tak, že fotony dopadají na atomy křemíku uvnitř CCD, odkud se na základě fotoefektu uvolňují záporně nabité elektrony. Kladně nabitá elektroda, která je umístěna pod každým pixelem a oddělena od křemíku izolační vrstvou oxidu křemičitého, zachycuje takto uvolněné elektrony. Dopadne-li na CCD více světla, vytváří se obraz, v němž jasné oblasti odpovídají pixelům, z nichž se uvolnilo více elektronů, je tam tedy větší elektrický náboj.

Po sejmutí obrazu je potřeba transportovat náboj na okraj čipu, kde bude převeden na napětí a typicky digitalizován. Náboj se nejprve přesune do tzv. transportních registrů, což jsou posuvné registry vytvořené právě CCD technologií. Tyto registry již nejsou citlivé na světlo a jsou schopny téměř beze změny náboj posouvat. Pracují tak, že soustavou tzv. řádkových a sloupcových registrů je postupně náboj odsouván do místa na čipu, v němž je integrován převodník náboje na napětí (zesilovač s velmi vysokou vstupní impedancí). Základní konstrukce CCD pro snímání dvourozměrného obrazu je zobrazena na obrázku 2.1.



Obrázek 2.1: Základní konstrukce CCD pro snímání dvourozměrného obrazu [6].

Takto sestavený senzor bez barevných filtrů je schopen zaznamenat pouze černobílý obraz. Aby však byl senzor schopen rozpoznat barvy, obvykle červenou, zelenou a žlutou, je potřeba před snímač umístit příslušné barevné filtry. Existují dvě metody umístění barevných filtrů před snímač. První metodou je použití tří samostatných snímačů a před každý umístit filtr příslušné barvy. Toto uspořádání se používá zejména u profesionálních TV kamer. Častěji se používá druhá metoda, kdy senzor obsahuje pouze jeden snímač a před každý pixel je umístěn filtr určité barvy. Nejčastěji se používá tzv. Bayerovo uspořádání (viz. obrázek 2.2). Toto uspořádání využívá faktu, že lidské oko je nejvíce citlivé na zelenou barvu, proto tento filtr obsahuje dvojnásobný počet zelených pixelů oproti červeným a modrým pixelům.



Obrázek 2.2: Bayerův filtr [7].

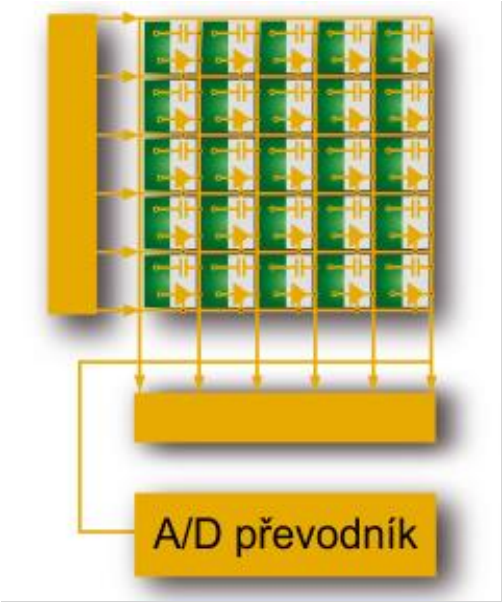
Mezi výhody CCD senzorů patří zejména možnost sejmout celý obraz v jediném okamžiku, a proto jsou tyto senzory vhodné pro dynamicky se měnící obraz. Další výhodou je, že technologie umožňuje konstrukci senzorů s velmi nízkým šumem a prakticky nulovým strukturálním šumem.

Mezi nevýhody CCD senzorů patří vysoká spotřeba energie, nutnost použití řady napájecích napětí a nemožnost integrace řídicích obvodů a A/D převodníků přímo na čip senzoru. Proto je v systému vždy nutno používat více čipů, což se projevuje i na velikosti samotného senzoru. Pokud je použita elektronická závěrka, tak u CCD senzorů může nastat, že když na některý pixel dopadne tolik světla, že přeteče jeho kapacita, tak se přebytečné elektrony roztečou do okolních pixelů v řadě. Tím v okolí silného světla vzniknou na fotografii rovnoběžné čárky nepravidelných délek. To se nazývá blooming. [4] [8] [9]

2.2 Technologie CMOS

Novější typ digitálního obrazového senzoru je CMOS (Complementary Metal–Oxide–Semiconductor). CMOS mají světlocitlivé buňky podobného typu jako CCD. Rozdíl je v tom, že hodnota náboje je převáděna na napětí přímo v jednotlivých buňkách a toto napětí je přenášeno na okraj čipu pomocí soustavy analogových multiplexorů. Poté je napětí digitalizováno, často na samotném čipu. [9] [8]

CMOS čipy se dělí na pasivní (passive-pixel sensor – PPS), které jsou tvořeny pouze fotodiodami, a aktivní (active-pixel sensor – APS), které mají u každé buňky navíc zesilovač a obvod odstraňující šum. Současné kamery používají vždy aktivní CMOS obrazové senzory. Základní schéma aktivního CMOS obrazového senzoru s A/D převodníkem na čipu je zobrazeno na obrázku 2.3.



Obrázek 2.3: Základní schéma aktivního CMOS obrazového senzoru s A/D převodníkem na čipu [10].

Z principu činnosti CMOS senzorů je zřejmé, že expozice pixelů nemůže probíhat současně, protože čtení obrazu pixelů probíhá sekvenčně, a tedy ani začátek expozice nemůže být současný, pokud expoziční doba všech pixelů obrazu má být stejná. Proto začátek expozice musí probíhat ve stejném časovém sledu jako vyčítání obsahu pixelů. Čtení pixelů se obvykle provádí postupně odshora

dolů. U statických snímků toto nevadí, ovšem u rychle se měnících snímků může dojít k tzv. „rolling shutter“ efektu, který je patrný například u rychle se otáčejících vrtulí.

Další nevýhodou CMOS senzorů je, že mají v porovnání s CCD senzory vyšší šum i strukturální šum. CMOS senzory mají pro každý pixel svůj vlastní zesilovač a každý zesilovač může zesilovat jinak, a proto mají vyšší strukturální šum než CCD. Dalším důvodem šumu je, že ony zesilovače jsou velmi blízko samotné fotodiody. Tyto zesilovače také způsobují, že je plocha pro dopadající světlo zmenšena, proto mají tyto senzory horší světelnou citlivost. Běžné CMOS senzory se snaží šum a horší světelnou citlivost co nejvíce eliminovat, a proto obsahují různé optimalizace.

Výhodou CMOS senzorů je, že na jeden CMOS čip lze integrovat i řídicí obvody a další elektroniku a lze používat jen jedno napájecí napětí. To má pozitivní vliv na spotřebu elektrické energie, která je u CMOS mnohem nižší než u CCD, a taky na velikost CMOS senzoru. Mezi další výhody CMOS senzorů patří jejich nízká cena, dále pak možnost načíst pouze výřez obrazu, není potřeba vždy načítat celý obraz. [9] [8]

3 Detekce obličeje a charakteristických bodů v něm

V této kapitole budou popsány algoritmy počítačového vidění, které slouží k detekci obličejů. V první části budou popsány algoritmy, které slouží s detekci obličeje jako celek, v druhé části pak budou popsány algoritmy, které slouží k lokalizaci jednotlivých charakteristických bodů v obličejích.

Algoritmy určené k detekci obličejů (respektive čehokoliv jiného) lze rozdělit podle toho, jestli používají manuálně vytvořený klasifikátor nebo vyučovaný klasifikátor. Manuálně vytvořené klasifikátory se dnes již téměř nepoužívají, protože dosahují horších výsledků než vyučované klasifikátory. Z algoritmů používajících vyučovaný klasifikátor se dnes používají zejména detektor Viola-Jones a detektor založený na histogramech orientovaných gradientů (HOG), a to především proto, že jsou schopny detekovat s velmi vysokou úspěšností a dokážou pracovat v reálném čase, což je pro detekci z videa nezbytné.

V druhé části této kapitoly se budu zabývat algoritmy sloužící k lokalizaci charakteristických bodů. K tomu se často používají metody fungující na principu aktivního modelu (šablony). Tyto metody slouží k tomu, aby na obličej (popřípadě jakýkoliv jiný objekt) přesně namapovaly model skládající se z bodů, které si v různých obličejích významově odpovídají. Může se jednat o koutek levého a pravého oka, špičku nosu atd.

Rozlišujeme metodu Active Shape Model (AMS) [11] (kapitola 3.4) a Active Appearance Model (AAM) [11] (kapitola 3.6). Model pro metodu ASM se skládá z bodů a hran mezi nimi, tento model se pak nazývá Statistical Shape Models [11] (kapitola 3.3). Model pro metodu AAM se skládá z bodů a hran, ale i z informace o textuře uvnitř modelu, tento model se pak nazývá Statistical Models of Appearance [11] (kapitola 3.6). Všechny výše zmíněné metody byly navrženy T. F. Cootesem, C. J. Taylorem a kolektivem. [11] [12]

Další možností, jak lokalizovat charakteristické body v obličeji, je použití metody „One Millisecond Face Alignment with an Ensemble of Regression Trees“ [13] (kapitola 3.7).

3.1 Viola-Jones

V roce 2001 Paul Viola a Michael J. Jones [14] představili detektor obličejů, který byl schopen zpracovat obrázky velmi rychle a zároveň dosahoval vysoké úspěšnosti detekce. Mezi jeho další pozitivní vlastnosti patří značná nezávislost na osvětlení a velikosti sledovaného objektu. Jedná se o obecný detektor, který je schopen detekovat cokoliv, na co byl tento detektor natrénován, tím pádem je vhodný i pro detekci očí, nosu, pusy atd.

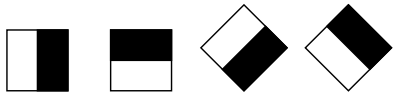
Vychází ze tří hlavních poznatků. Zaprvé byla zavedena nová reprezentace obrazu zvaná integrální obraz (podkapitola 3.1.2), který způsobuje, že jsou příznaky v detektoru počítány velmi rychle. Druhým je jednoduchý a efektivní klasifikátor, který používá algoritmus AdaBoost (podkapitola 3.1.4), který vybere malý počet kritických příznaků z velkého množství potenciálních příznaků. Třetí je kaskáda klasifikátorů (podkapitola 3.1.3), která umožňuje rychle zahodit výřezy s pozadím, zatímco více času stráví na výpočtech výřezů, kde je potenciální obličej. Tento detektor pracuje s obrázky ve stupních šedi. [14] [15]

3.1.1 Haarovy příznaky

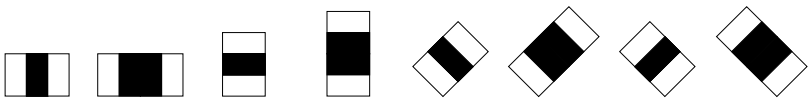
Při detekci obličeje se klasifikují obrázky na základě hodnoty jednoduchého příznaku. Použití příznaků je mnohem efektivnější a rychlejší než používat přímo pixely. Snahou detektoru Viola-Jones je získat velkou řadu jednoduchých příznaků s minimálními výpočetními nároky. Takovým typem příznaků jsou příznaky založené na principu podobnému definici Haarových příznaků (tzv. Haar-like features). Tyto příznaky používají změny kontrastu hodnot mezi sousedními obdélníkovými skupinami pixelů. Rozdíl kontrastu mezi skupinami pixelů je použit k určení relativně světlých a tmavých míst.

Používáme tři typy příznaků. Pokud spolu sousedí dva obdélníky, jedná se o hranový příznak (viz obrázek 3.1) a hodnota příznaku se počítá jako rozdíl mezi oběma obdélníkovými oblastmi. Pokud spolu sousedí tří obdélníky, jedná se o čárový příznak (viz obrázek 3.2) a hodnota příznaku se počítá jako součet vnějších obdélníků, od kterých se odečte vnitřní obdélník. Poslední používanou možností je, že jeden obdélník je obsažen ve druhém. Pak se jedna o příznaky středové (viz obrázek 3.3) a hodnota příznaku se počítá jako rozdíl vnějšího a vnitřního obdélníku.

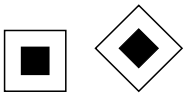
Jednotlivé příznaky jsou použity na celý vstupní obraz, přičemž zároveň dochází ke změně velikostí jednotlivých příznaků (tj. velikosti jednotlivých obdélníků) z velikosti 1×1 až na velikost odpovídající vstupnímu obrazu. Za předpokladu, že základní rozlišení detektoru je 24×24, dostaneme přibližně 160 000 hodnot příznaků. [14] [15] [16]



Obrázek 3.1: Hranové příznaky [17].



Obrázek 3.2: Čárové příznaky [17].



Obrázek 3.3: Středové příznaky [17].

3.1.2 Integrální obraz

Pro výpočet Haarových příznaků z běžného obrazu, bychom museli pro každý obdélníkový příznak počítat sumu intenzit pixelů daného příznaku. Proto byl zaveden tzv. integrální obraz, pomocí kterého lze spočítat obdélníkové příznaky velmi rychle. Každý bod tohoto obrazu je počítán jako součet všech bodů, které se nacházejí nad a zároveň nalevo od zadaného bodu. To nám definuje vzorec:

$$i_{int}(x,y) = \sum_{x' \leq x, y' \leq y} i(x',y') \tag{3.1}$$

kde $i_{int}(x,y)$ je hodnota integrálního obrazu a $i(x',y')$ je hodnota původního obrazu.

Výpočet hodnot integrálního obrazu se provádí podle následujících vzorců:

$$s(x,y) = s(x,y-1) + i\,(x,y) \tag{3.2}$$

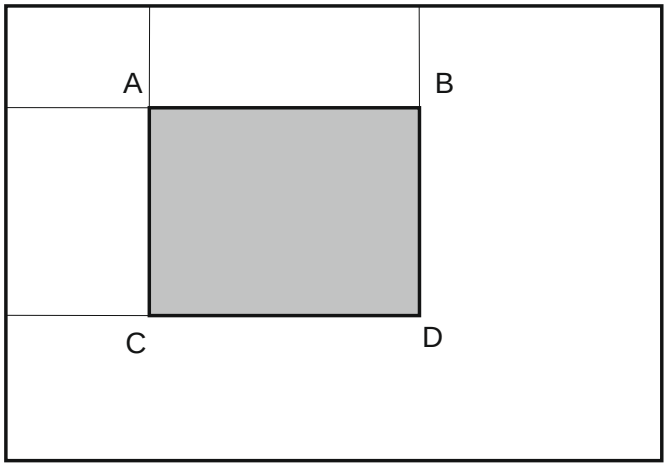
$$i_{int}(x,y) = i_{int}(x-1,y) + s(x,y) \tag{3.3}$$

kde $s(x,y)$ je kumulovaný součet hodnot v řádku obrazu, za podmínek, že $s(x,-1) = 0$ a $i_{int}(-1,y) = 0$.

Celý integrální obraz lze spočítat pouze jedním průchodem originálního obrazu. Při použití integrálního obrazu může být každý obdélník počítán pouze pomocí čtyř referencí do paměti, což nám umožní počítat libovolně velký obdélník konstantní rychlostí. Pokud máme příznak o dvou sousedních obdélnících, stačí nám jen 6 referencí do paměti a pokud máme příznak o třech sousedních obdélnících, stačí nám jen 8 referencí do paměti. [14] [15] [16]

Hodnota obdélníkového příznaku S z obrázku 3.4 bude vypočítána pomocí vzorce:

$$S = i_{int}(A) - i_{int}(B) - i_{int}(C) + i_{int}(D) \tag{3.4}$$



Obrázek 3.4: Ukázka výpočtu hodnot obdélníkových příznaků z integrálního obrazu [14].

V předchozí části jsme si ukázali, jak se pomocí integrálního obrazu vypočtou obdélníkové příznaky, které jsou rovnoběžné s osou x. Často pracujeme s příznaky, které nejsou rovnoběžné s osou x. V praxi se nejčastěji používají příznaky natočené o 45°, jak můžeme vidět na obrázcích 3.1, 3.2 a 3.3. Při použití takovýchto příznaků potřebujeme tzv. rotovaný integrální obraz. Rotovaný integrální obraz je spočítán pomocí dvou průchodů obrazem. První průchod je prováděn zleva doprava a shora dolů a druhý průchod je prováděn zprava doleva a zdola nahoru. Pro výpočet obdélníkového příznaku nám opět stačí jen čtyři reference do paměti. [17]

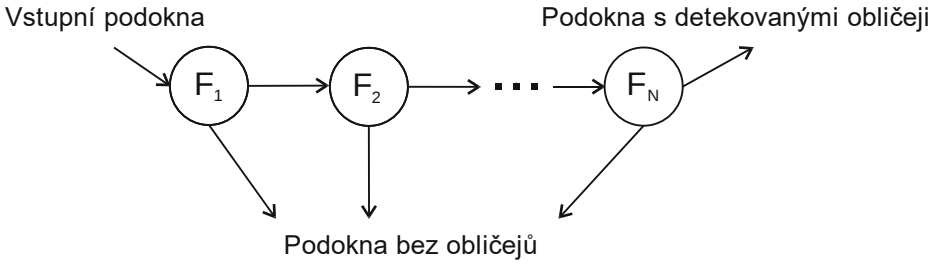
3.1.3 Kaskáda klasifikátorů

Snahou detektoru je snížení průměrné doby, kterou detektor stráví prohledáváním každého podokna. Toho bylo dosaženo kaskádovým zapojením klasifikátorů. Bylo využito poznatku, že podoken, která neobsahují hledaný objekt, je mnohem více než podoken, která obsahují hledaný objekt. Proto jsou podokna bez hledaného objektu detekována rychleji.

Kaskáda klasifikátorů je degenerovaný binární rozhodovací strom. V každém stupni kaskády se rozhoduje, zda dané podokno obsahuje hledaný objekt nebo ne. Jsou tak detekovány téměř všechna podokna s hledanými objekty, zatímco zamítnuta je pouze část podoken bez hledaného objektu. U zamítnutého podokna je ukončena klasifikace. Podokno, které není zamítnuté, postupuje do dalšího

stupně kaskády. To se opakuje tak dlouho, až se dojde k poslednímu stupni kaskády. Ten, když úspěšně projde, tak je dané podokno označené za podokno s hledaným objektem. Jedná se o sekvenční průchod. Kaskáda klasifikátorů je zobrazena na obrázku 3.5.

Každý stupeň kaskády je složen ze slabých klasifikátorů. Tyto slabé klasifikátory nedokážou spolehlivě detekovat hledaný objekt, ten dokáže detekovat až silný klasifikátor složený ze slabých klasifikátorů. Každý stupeň kaskády obsahuje více klasifikátorů než předchozí stupeň, čili dokáže detekovat lépe. [14] [15] [16]



Obrázek 3.5: Kaskáda klasifikátorů [14].

3.1.4 AdaBoost

Aby bylo možné detekovat objekty, musí být nejprve natrénován kaskádový klasifikátor. Pro podokno o velikosti 24×24 pixelů se vypočítá 160 000 Haarových příznaků, což je mnohem více než počet pixelů. I přesto, že každý příznak může být vypočten velmi efektivně, výpočet všech příznaků je velmi náročný. Experimentováním bylo dokázáno, že jen s pomocí velmi malého počtu Haarových příznaků může být vytvořen efektivní klasifikátor [14]. Kvalitní klasifikátor by měl tyto příznaky najít.

Detektor Viola-Jones používá jako klasifikační algoritmus AdaBoost (Adaptive Boosting), který v roce 1995 představili Y. Freund a R. Schapire a který vychází z metody strojového učení zvané boosting. Používá se jak na výběr příznaků, tak na trénování klasifikátorů. Základním principem je lineární kombinace několika slabých klasifikátorů, někdy též označovaných jako slabí žáci, do jednoho silného klasifikátoru (silný žák), který je úspěšnější než jakýkoliv ze slabých klasifikátorů. Jedinou podmínkou slabého klasifikátoru je, aby jeho chyba byla menší než 50%. AdaBoost se od základního boostingu liší tím, že po každém kole učení je vypočtena nová váha slabého klasifikátoru podle velikosti jeho chyby.

K natrénování klasifikátoru potřebujeme jako vstup dvě sady obrázků. První sadou jsou pozitivní obrázky, čili obrázky s hledaným objektem, a druhou sadou jsou negativní obrázky, čili obrázky, na kterých se nenachází hledaný objekt.

Na začátku jsou všechny váhy D_t nastaveny rovnoměrně. V každé smyčce algoritmu se pak vybere slabý klasifikátor s nejmenší váženou chybou klasifikace ε_t při daných váhách D_t. Musí se zkontrolovat, jestli chyba klasifikátoru ε_t není větší než 50%, pokud je, tak se daný algoritmus ukončí. Dále se vypočte váha α_t slabého klasifikátoru podle velikosti jeho chyby. V každém cyklu se musí nakonec aktualizovat jednotlivé váhy D_t trénovací množiny. Váhy u dobře klasifikovaných dat klesají, naopak u špatně klasifikovaných dat stoupají. Díky tomuto se nevybere v každém kroku stejný slabý klasifikátor. Váhu vzorku lze přímo převést na výsledek klasifikátoru. Zvyšování počtu klasifikátorů může vést k přetrénování, ovšem na většině dat nemá AdaBoost tendenci se přetrénovat. Pseudokód algoritmu je popsán v algoritmu 3.1. [15] [16] [18]

Algoritmus 3.1: AdaBoost [18]

Vstup: $(x_1, y_1), \dots, (x_m, y_m)$, kde $x_i \in X$ – hodnota příznaku, $y_1 \in \{-1,1\}$ – třída odpovídající příznaku i

Inicializace vah $D_1(i) = \frac{1}{m}$

for $t = 1, \dots, T$ **do**

Vyber slabý klasifikátor s nejmenší váženou chybou

$$\varepsilon_t = \sum_{i=1}^{m} D_t(i)\left[y_i \neq h_j(x_i)\right]$$

if $\varepsilon_t \geq \frac{1}{2}$ **then** stop

Nastav $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$

Aktualizuj váhy:

$$D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$$

kde Z_t je normalizační faktor

end

Výstup:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

3.2 Histograms of oriented gradients (HOG)

Dalším velmi používaným typem detektoru je detektor založený na histogramech orientovaných gradientů, který poprvé v roce 2005 představili Navneet Dalal a Bill Triggs [19]. Hlavní myšlenkou tohoto detektoru je, že hledaný objekt může být popsán pomocí intenzity gradientů nebo směrnicemi hran. Mezi pozitivní vlastnosti detektoru HOG patří především odolnost vůči změně osvětlení, změně kontrastu, šumu, změně měřítka, velikosti, avšak není odolný vůči rotaci. Lze pracovat jak s fotografiemi ve stupních šedi, tak i s barevnými fotografiemi.

V prvním kroku se nejprve musí detekovat hrany v obraze. To se nejčastěji realizuje pomocí první derivace ve směru x a y. Před samotným výpočtem se nejprve provede Gaussovo rozostření. Derivace se provádí pomocí konvoluce vstupního obrazu s vhodným konvolučním jádrem. Jako vhodná konvoluční jádra byly experimentováním zjištěny matice:

$$D_x = [1 \quad 0 \quad -1] \tag{3.5}$$

$$D_y = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \tag{3.6}$$

kde D_x a D_y jsou matice sloužící k derivaci pro osu x a y.

Autoři dále zkoušeli použít k detekci hran i jiné masky jako například Sobelův operátor nebo diagonální masku. Experimentováním však zjistili, že použitím první derivace ve směru x a y je dosahováno nejlepších výsledků.

$$D_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{3.7}$$

$$D_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{3.8}$$

Dále autoři zjistili, že pokud před výpočtem derivace provedeme rozmazání vstupního obrazu pomocí Gaussova filtru, dosáhneme lepších výsledků. To je dáno především tím, že Gaussův filtr dokáže zamaskovat šum a některé nepodstatné detaily.

Dále se okno obrázku rozdělí na menší stejně velké oblasti, tzv. buňky. Buňky mohou být buď čtvercové (R-HOG - 3.6 vlevo), nebo kruhové (C-HOG 3.6 vpravo), většinou se však používají čtvercové buňky. Pro každou buňku se ze všech pixelů v buňce vypočítají lokální jednorozměrné histogramy orientovaných gradientů. Výsledná velikost úhlu gradientu může být buď v rozsahu 0° – 180°, nebo 0° – 360°. Velikosti gradientů obsažených v jednotlivých buňkách se rozdělí do několika tříd, ze kterých se následně vytvoří histogram pro každou buňku. Nejčastěji se rozsah úhlů dělí do 9 tříd, pokud používáme rozsah 0° – 180°, pak má každá třída šířku 20°.

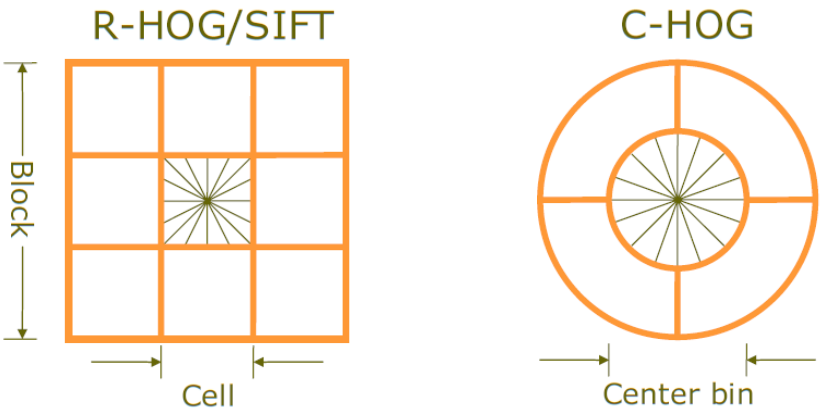
Pro zvýšení odolnosti na změně osvětlení, kontrastu mezi jednotlivými buňkami se používá normalizace. Normalizace se provádí nad částečně překrývajícími se bloky, které se při použití čtvercových buněk skládají nejčastěji z 3×3 buněk (obrázek 3.6). K normalizaci lze použít jednu ze čtyř odlišných normalizačních metod: L2-norm (2.9), L1-norm (2.9), L1-sqrt (2.10) a dále L2-hys, která je L1-norm následována clippingem (omezení maximální hodnoty v na 0,2) a renormalizace. Experimentováním bylo zjištěno, že poslední tři metody mají podobný výkon, zatímco první je o něco méně výkonná. [19]

$$L2 - norm\colon f = \frac{v}{\sqrt{\|v\|_2^2 + e^2}} \tag{3.9}$$

$$L1 - norm\colon f = \frac{v}{\|v\|_2^2 + e} \tag{3.10}$$

$$L1 - sqrt\colon f = \sqrt{\frac{v}{\|v\|_1 + e}} \tag{3.11}$$

kde v je nenormalizovaný vektor obsahující všechny histogramy v daném bloku, $\|v\|_k$ je jeho k-norm pro $k = 1{,}2$ a e je malá konstanta (přesná hodnota je nedůležitá).



Obrázek 3.6: Zobrazení jednoho bloku a buněk. Vlevo R-HOG, vpravo C-HOG [19]

3.2.1 Support vector machines (SVM)

K natrénování klasifikátoru založenému na HOG se nejčastěji používá algoritmus SVM (support vector machines - algoritmus podpůrných vektorů). Jedná se o metodu strojového učení, která spadá do kategorie učení s učitelem. Cílem algoritmu je najít takovou nadrovinu, která optimálně rozděluje dvě třídy trénovacích dat, tj. nadrovinu, která má maximální vzdálenost nejbližších bodů k nadrovině. K popisu nadroviny stačí pouze body, které jsou nejblíže nadrovině, tyto body nazýváme podpůrné vektory (suppot vector). Výhodou tohoto algoritmu je, že dokáže oddělit i nelineárně oddělitelná data (např. kružnici) lineární funkcí. Pro nelineárně oddělitelná data se používají jádrové transformace (kernel transformations), pomocí kterých je možné mapovat vstup do prostoru o více dimenzích. Zjednodušeně lze říci, že každý n-dimenzionální vektor lze převést na n+1-dimenzionální vektor, přidáním dalšího atributu závislého na původních n atributech. Pomocí toho lze oddělit nelineárně ohraničená data nadrovinou tím, že posuneme jednu třídu dat podél osy nové dimenze. Obecný popis lineárního klasifikátoru pak má tvar:

$$f(x) = w^T x + b \tag{3.12}$$

kde w je váhový vektor a b je posunutí (bias).

Jestliže $f(x) \geq 0$ předpokládáme, že se jedná o pozitivní klasifikaci a jestliže $f(x) < 0$ předpokládáme, že se jedná o negativní klasifikaci.

V algoritmu SVM se optimální lineární oddělovač hledá pomocí metody kvadratického programování. Předpokládejme, že jsou příklady x_i s klasifikací $y = \pm 1$ a cílem je najít optimální oddělovače. Problém lze převést na hledání hodnot parametrů α_i, které maximalizují výraz:

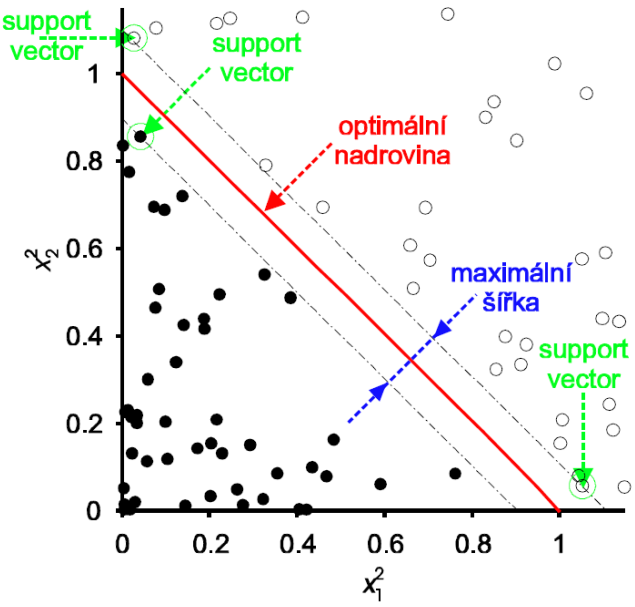
$$\sum_t \alpha_i - \frac{1}{2} \sum_{t,j} \alpha_i \alpha_j y_t y_j (x_i, x_j) \tag{3.13}$$

kde x_i je vstupní příklad, y_i je klasifikace vstupního příkladu $y = \{-1, +1\}$ a $\alpha_i$je váhový koeficient. Přičemž platí omezení:

$$a_i \geq 0 \tag{3.14}$$

$$\sum_t \alpha_i y_i = 0 \tag{3.15}$$

Na obrázku 3.7 je zobrazeno použití lineárního SVM na rozdělení dvou tříd objektů ve 2D prostoru. Zeleně jsou okroužkovány podpůrné vektory (*support vector*), červená přímka zobrazuje optimální nadrovinu a modrá úsečka zobrazuje onu maximální vzdálenost nejbližších bodů (podpůrných vektorů) k nadrovině.



Obrázek 3.7: Ukázka použití lineárního SVM na rozdělení dvou tříd objektů ve 2D prostoru [20].

Kromě lineárního jádra se často používá polynomiální jádro, RBF (Radial basis function) nebo sigmoidní jádro, které dokážou lépe oddělit oblasti, které nejsou lineárně oddělitelné.

Pokud používáme SVM k natrénování detektoru HOG, tak jedna třída bude obsahovat pozitivní obrázky a druhá třída bude obsahovat negativní obrázky. Hledaná nadrovina nám určuje, kterým směrem od nadroviny leží hledané objekty a kterým směrem od nadroviny leží pozadí obrázku. [20] [21] [22]

3.3 Statistical Shape Models

V této podkapitole bude popsán statisticky model tvarů, který bude použit pro reprezentaci objektů v obrázku. Tvar každého objektu je reprezentován sadou n bodů, které se mohou nacházet v několika dimenzích, nejčastěji se pracuje s body ve dvou případně třech dimenzích.

V další části budou popsány kroky nutné k sestavení modelu. Tato podkapitola bude vycházet z [11] [23] [24].

3.3.1 Vhodné body

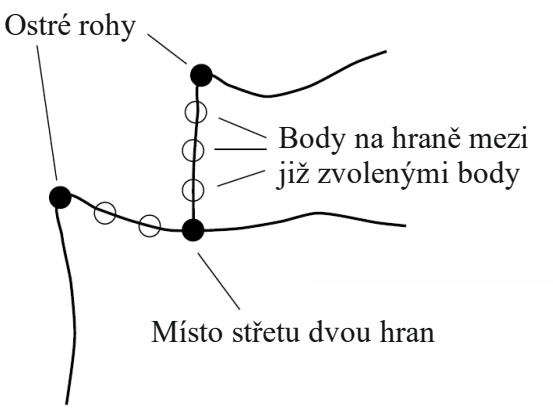
Vhodně zvolené body by se měly nacházet na všech obličejích a měly by se na každém obličeji nacházet na podobném místě. V první řadě je potřeba označit na celé sadě obrázků všechny body, které si odpovídají, a to s co nejvyšší přesností. Nejlepší metoda k označení všech bodů na všech obrázcích je manuálně člověkem, to je však velmi časově náročné. V praxi se často používají automatické nebo poloautomatické metody k označení všech bodů. Vhodně zvolené body by se měly ideálně nacházet v ostrých rozích, v místech střetu dvou hran nebo snadně biologicky lokalizovatelných bodech, jako

například střed oka, špička nosu atd. Takovýchto bodů bývá většinou málo, proto se ještě přidávají body, které se nacházejí na hranách mezi dvěma již zvolenými body. Správně zvolené body jsou zobrazeny na obrázku 3.8. Fotografii člověka s již označenými body lze vidět na obrázku 4.2.

Pokud je model tvořen n body v d dimenzích, pak je model tvořen nd prvkovým vektorem. Jednotlivé dimenze jsou řazeny ve vektoru za sebe. Například pokud máme 2D obrázek, můžeme reprezentovat n bodů tvaru $\{(x_i, y_i)\}$ jako $2n$ prvkový vektor x:

$$x = (x_1, \dots, x_n, y_1, \dots, y_n)^T \tag{3.16}$$

Pokud máme N prvkovou trénovací sadu obrázků, pak bude vygenerováno N vekotorů x_i.



Obrázek 3.8: Zobrazení správně zvolených bodů [11].

3.3.2 Zarovnání trénovací sady

Model běžně považujeme za nezávislý na měřítku, natočení a pozici. Proto před tím, než bude provedeno samotné sestavení výsledného modelu, musí být tato závislost odstraněna. K tomu lze použít několik různých algoritmů, jedním z nejpoužívanějších je Prokrustova analýza. Ta se snaží, aby suma čtverců vzdáleností modelů od průměrného modelu ($D = \sum|x_i - \bar{x}|^2$) byla minimální. Přestože existují analytická řešení, jednoduchý iterativní postup navržený panem Cootesem [11] je dostačující k zarovnání množiny modelů. Tento postup je popsán v algoritmu 3.2.

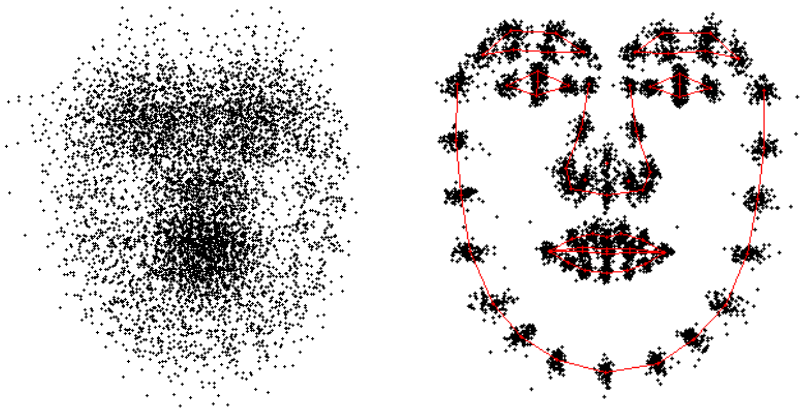
Algoritmus 3.2: Zarovnání trénovací sady [11]

1. Přesuň každý model tak, aby jeho těžiště bylo v počátku.
2. Vyber jeden model jako počáteční odhad průměrného tvaru a měřítka, tak že $|\bar{x}| = 1$.
3. Ulož první odhad jako $\bar{x}_0$.
4. Zarovnej všechny modely se současným odhadem průměrného modelu.
5. Znovu urči průměr ze seřazených modelů:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

6. Změň měřítko průměrného modelu $\bar{x}_0$, tak aby $|\bar{x}| = 1$.
7. Jestliže se $\bar{x}$ příliš nemění, zastav, jinak pokračuj na bod 4.

Ve většině případů stačí pouze dvě iterace algoritmu 3.2 k dostatečnému zarovnání modelů. Na obrázku 3.9 lze vidět trénovací sadu před a po zarovnání.



Obrázek 3.9: Trénovací sada před zarovnáním (vlevo) a po zarovnání (vpravo) [24].

3.3.3 Sestavení modelu

Každý zarovnaný model lze považovat za jeden bod v nd-dimenzionálním prostoru a celou trénovací sadu lze považovat jako mračno bodů v tomto prostoru. Aby bylo možné zachytit statistickou změnu tvaru modelu, použije se PCA (Analýza hlavních komponent). PCA je zobrazené v algoritmu 3.3. Cílem PCA je výpočet vlastních vektorů ϕ_i kovarianční matice S, které odpovídají hlavním osám mračen bodů.

Algoritmus 3.3: Analýza hlavních komponent (PCA) [24]

1. Vypočítej průměr dat

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

2. Vypočítej kovarianci dat

$$S = \frac{1}{N-1}\sum_{i=1}^{N} (x_i - \bar{x})\,(x_i - \bar{x})^T$$

3. Vypočítej vlastní vektor ϕ_i a odpovídající vlastní číslo λ_i z S.
4. Seřaď vlastní vektory tak, aby $\lambda_i \geq \lambda_{i+1}$
5. Ulož prvních t vlastních vektorů jako matici $\Phi = (\phi_1|\phi_2|\ldots|\phi_t)$

PCA umožňuje, že každý model x z trénovací sady může být aproximován pomocí průměrného modelu a malého počtu parametrů b:

$$x \approx \bar{x} + \Phi b \tag{3.17}$$

kde b je t-dimenzionální vektor získaný promítnutím x do podprostoru definovaného průměrným modelem a maticí Φ:

$$b = \Phi^T(x - \bar{x}). \tag{3.18}$$

3.4 Active Shape Model (ASM)

Pokud již máme model, pak je potřeba daný model co nejpřesněji namapovat na obličej. To se provádí pomocí nastavování parametrů tak, aby daný model co nejlépe pasoval na vstupní obrázek. Parametr b definuje tvar modelu, parametr (X_t, Y_t) definuje pozici modelu, parametr θ definuje natočení modelu a parametr s definuje změnu měřítka. [11] [23]

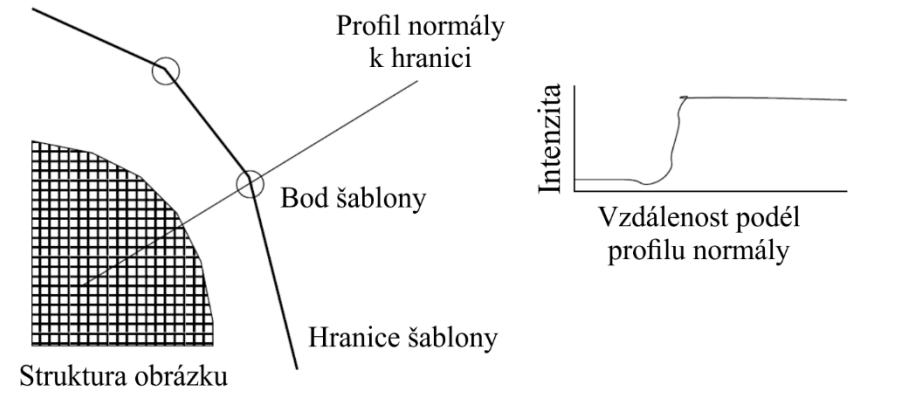
Na počátku známe přibližnou startovací pozici a tvar daného modelu. Poté se snažíme iterativně zdokonalovat napasování modelu na objekt na obrázku. To se provádí pomocí algoritmu 3.4.

Algoritmus 3.4: Algoritmus Active Shape Model [11]

1. Prozkoumej oblast obrázku okolo každého bodu x_i a najdi nejlepší blízkou pozici pro nový bod x_i'
2. Aktualizuj parametry (X_t, Y_t, s, θ, b) tak, aby model co nejvíce pasoval na nově nalezené body
3. Opakuj, dokud dochází ke zpřesňování mapování modelu na objekt na obrázku.

3.4.1 Nalezení nových bodů

Nalezení nových bodů se provádí tak, že pro každý bod se podíváme na profil normály k hranici modelu procházející daným bodem (obrázek 3.10). Pokud budeme předpokládat, že hranice modelu bude odpovídat hraně, pak lze jednoduše najít nejsilnější hranu (včetně orientace, pokud ji známe) na profilu normály. Tato pozice nám udává nové odhadované umístění bodu modelu.



Obrázek 3.10: Ukázka profilu normály k hranici modelu [11].

Nicméně body modelu se nemusí vždy nacházet na nejsilnější hraně, mohou se také nacházet na slabších vedlejších hranách nebo na některé jiné struktuře v obraze. Nejlepším řešením je učit se z trénovací sady, co přesně hledat v cílovém obrázku.

Existují dva obecné přístupy. Prvním je vytvářet statistické modely struktury obrazu kolem každého bodu a během hledání jednoduše najít ty body, které nejlépe odpovídají modelu (tato metoda bude popsána níže). Druhým přístupem je nahlížet na problém jako na klasifikační úlohu. Vytvoříme dvě třídy, kde jedna bude obsahovat vzorek příklady rysů z požadované oblasti a druhá bude obsahovat vzorek příznaku z jiných oblastí. Klasifikátor pak lze použít k rozhodnutí, který bod nejlépe odpovídá dané oblasti.

Dále bude popsána jedna jednoduchá metoda modelování struktury. V podstatě funguje tak, že vezmeme vzorek podél normály a sestavíme statistický model šedotónové struktury.

Pro každý bod modelu vezmeme k pixelů na každou stranu podél profilu normály. Pak dostaneme $2k + 1$ vzorků, které mohou být uloženy ve vektoru g_i. Aby nebyly hodnoty závislé na hodnotě globálního osvětlení, ukládají se do vektoru pouze hodnoty změny intenzity osvětlení. Poté probíhá normalizace, kdy je každý vzorek podělen sumou absolutních hodnot profilů:

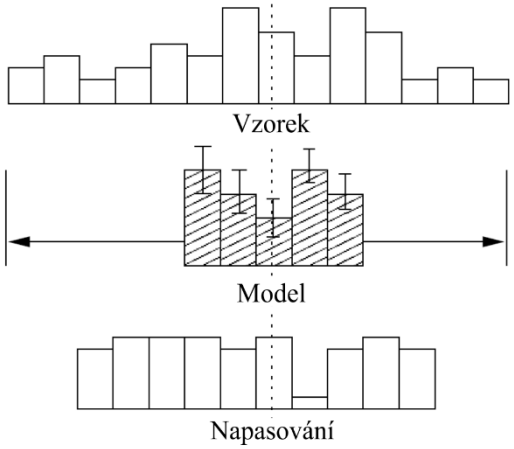
$$g_i \rightarrow \frac{1}{\sum_j |g_{ij}|} g_i. \tag{3.19}$$

Toto opakujeme pro každý obrázek v trénovací sadě. Pak dostaneme množinu normalizovaných vzorků $\{g_i\}$ pro každý bod modelu. Předpokládáme, že se jedná o vícedimenzionální Gaussovu funkci a odhadneme její střední hodnotu $\bar{g}$ a kovariantní matici S_g. To nám dává statický model profilu kolem bodu. Toto opakujeme pro každý bod modelu a dostaneme model každého bodu.

Kvalita mapování nového vzorku na model se spočítá pomocí Mahalanobisovy vzdálenosti takto:

$$f(g_s) = (g_s - \bar{g})^T S_g^{-1} (g_s - \bar{g}). \tag{3.20}$$

Během hledání nového bodu vytváříme profil o m pixelech na každou stranu od současného pixelu $(m > k)$. Posléze se testuje kvalita napasování modelu na každý z $2(m - k) + 1$ možných pozic podél vzorku (obrázek 3.11) a vybere se jeden, který dává největší shodu (nejnižší hodnota $f(g_s)$). To se provádí pro každý bod v modelu, čímž se navrhuje nová pozice pro každý bod.



Obrázek 3.11: Nalezení profilu podél vzorku, jenž nejlépe pasuje na model [11].

3.4.2 Přizpůsobení modelu novým bodům

Při aktivní deformaci modelu je potřeba znát algoritmus, který co nejlépe přizpůsobí model zadaným bodům. Pro popis modelu se používá parametr b popisující tvar modelu, parametr (X_t, Y_t) popisující

pozici modelu, parametr θ popisující natočení a parametr s popisující změnu měřítka. Pozice bodů modelu v obrázku se spočítá:

$$x = T_{X_t,Y_t,s,\theta}(\bar{x} + \phi b) \tag{3.21}$$

kde funkce $T_{X_t,Y_t,s,\theta}$ provádí rotaci o úhel θ, změnu měřítka o s a posun o (X_t, Y_t). Pokud je funkce použita pro jeden bod, tak může mít například tvar:

$$T_{X_t,Y_t,s,\theta}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} X_t \\ Y_t \end{pmatrix} + \begin{pmatrix} s\cos\theta & s\sin\theta \\ -s\sin\theta & s\cos\theta \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} \tag{3.22}$$

Nyní je potřeba najít nejlepší nastavení parametrů posuvu a změny tvaru tak, aby instance modelu x nejvíce pasovala na množinu obrazových bodů Y. To se provede pomocí minimalizace sumy čtverečních vzdáleností mezi odpovídajícími body v modelu a body v obrázku pomocí výrazu:

$$\left|Y - T_{X_t,Y_t,s,\theta}(\bar{x} + \phi b)\right|^2 \tag{3.23}$$

Iterativní algoritmus, který demonstruje, jak toho dosáhnout, je popsán v algoritmu 3.5.

Algoritmus 3.5: Nalezení nejlepší shody instance x a nových bodů Y [11]

1. Inicializuj parametr b, který popisuje tvar modelu, na 0
2. Vygeneruj instanci modelu $x = \bar{x} + \phi b$
3. Najdi parametry (X_t, Y_t), s a θ, které nejlépe mapují x na Y
4. Proveď inverzní transformaci T^{-1} pro všechny body Y a tím dosáhni transformaci bodů Y do počátku souřadnicového systému modelu:

$$y = T^{-1}_{X_t,Y_t,s,\theta}(Y)$$

5. Promítni y na rovinu tečny k $\bar{x}$ za pomocí $1/(y\bar{x})$
6. Uprav parametry modelu tak, aby odpovídaly y:

$$b = \phi^T(y - \bar{x})$$

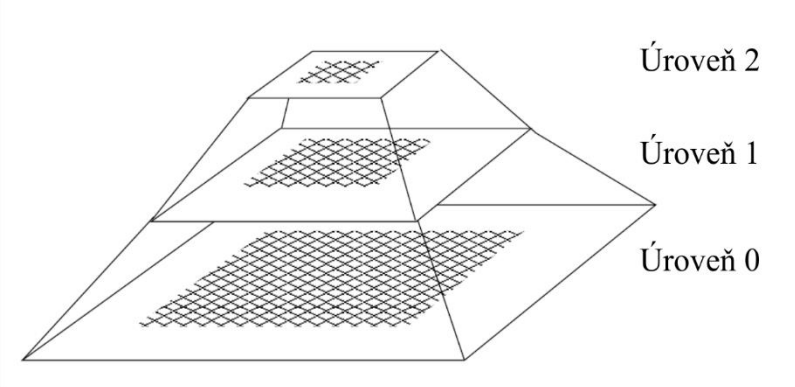
7. Aplikuj omezující podmínky, např. $b_i < \sqrt{\lambda_i}\lambda$
8. Pokud dochází k významné změně parametru b, přejdi na bod 2.

3.4.3 ASM s více rozlišeními

Ke zvýšení efektivity a robustnosti algoritmu se používá varianta AMS s použitím více rozlišení. Nejprve se provádí hledání bodů pomocí ASM na obrázcích s nízkým rozlišením a poté se přechází na obrázky s vyšším rozlišením. To vede k urychlení algoritmu a k tomu, že je algoritmus odolnější proti uvíznutí v nesprávné pozici.

Pro každý trénovací obrázek se vytvoří Gaussova pyramida. Na nejspodnější pozici pyramidy (úroveň 0) je původní obrázek. Obrázek na vyšší úrovni pyramidy je vytvořen pomocí vyhlazení původního obrázku a následném podvzorkování tak, abychom dostali obrázek o polovičních

rozměrech. Podobně se postupuje i pro další úrovně pyramidy. Gaussova pyramida je zobrazena na obrázku 3.12.



Obrázek 3.12: Gaussova pyramida [11]

3.5 Statistical Models of Appearance

U modelu Statistiacal Models of Appearence je objekt popsán nejen pomocí jeho tvaru, ale i pomocí textury (intenzita nebo barva přes celou oblast objektu). Pro vytvoření takového to statistického modelu je potřeba sestavit statistický model tvarů (popsáno v kapitole 3.3), statistický model textury a vzájemně propojit oba modely.

Poté bude popsáno vytváření statistického modelu textury a propojení obou modelů. Tato podkapitola vychází z [11] [23] .

3.5.1 Statistical Models of Texture

K vytvoření statistického modelu textury je potřeba nejprve zdeformovat každý trénovací obrázek tak, aby jeho charakteristické body lícovaly s body průměrného modelu tvarů. Tím se odstraní nežádoucí proměnnost textury způsobená rozdílným tvarem a natočením hlavy. Poté se navzorkuje intenzita z tvarově normalizovaného obrázku nad oblastí, která je definována průměrným modelem tvarů, a tato intenzita se uloží do vektoru textury g_{im}. Na obrázku 3.13 je na levé části zobrazen trénovací obličej s vyznačenými charakteristickými body, na pravé straně jsou pak nahoře zobrazeny pouze charakteristické body, které slouží pro vytvoření modelu tvarů, dole je pak zobrazen tvarově normalizovaný obličej dle průměrného modelu tvarů, který slouží pro vytvoření modelu textury.

Aby byla minimalizovaná závislost na globálním osvětlení, tak se normalizují trénovací obrázky pomocí váhy α a ofsetu β:

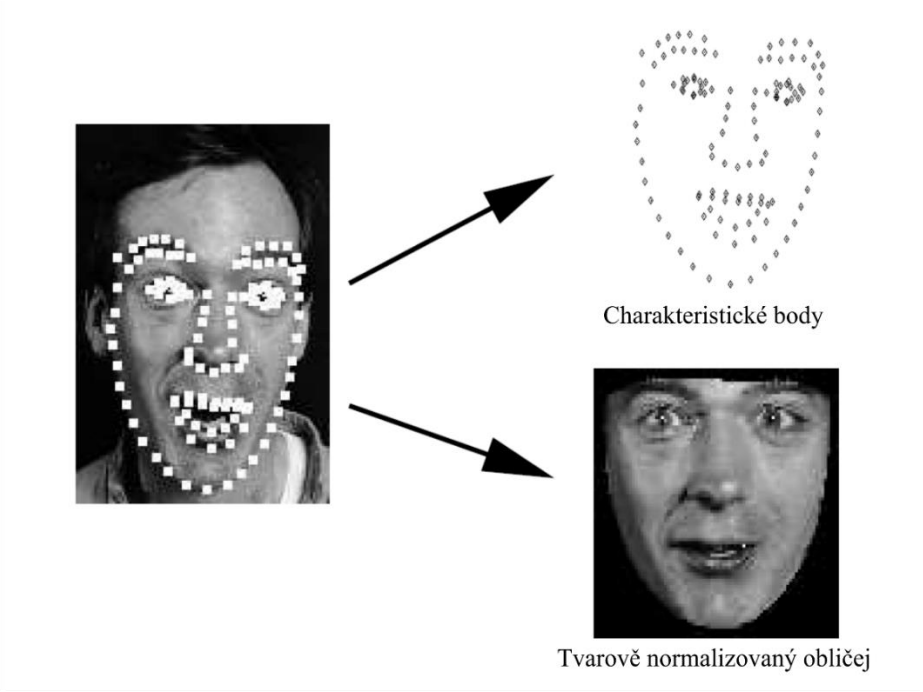
$$g = \frac{(g_{im} - \beta 1)}{\alpha} \tag{3.24}$$

Hodnoty α a b jsou zvoleny tak, aby vektor intenzity nejlépe odpovídal normalizovanému průměru. Nechť $\bar{g}$ je průměrná hodnota normalizovaných dat, váhy a ofsetu, pak součet všech prvků je nula a všechny prvky jsou si rovny. Hodnoty α a β potřebné pro získání normalizovaného g_{im} se pak spočítají:

$$\alpha = g_{im} . \bar{g} \tag{3.25}$$

$$\beta = \frac{(g_{im}.1)}{n} \tag{3.26}$$

kde n je počet prvků vektoru.



Obrázek 3.13: Nalevo je zobrazen trénovací obličej s vyznačenými body, vpravo nahoře jsou zobrazeny charakteristické body a vpravo dole je zobrazen tvarově normalizovaný obličej [11].

Výpočet průměru normalizovaných dat se pak spočítá pomocí rekurzivního algoritmu. Nejprve se vybere jeden vektor jako první odhad, pak se všechny vektory zarovnají (pomocí rovnic (3.24), (3.25) a (3.26)) a vypočítá se nový průměr. To se opakuje, dokud se mění hodnota průměru.

Pomocí použití PCA na normalizovaná data získáme lineární model:

$$g = \bar{g} + P_g b_g \tag{3.27}$$

kde $\bar{g}$ je průměrný normalizovaný vektor, P_g množina ortogonálních vektorů a b_g je množina parametrů.

Textura obrázku může být vygenerována pomocí parametrů textury b_g a normalizačních parametrů α a β. Kvůli linearitě jsou tyto parametry reprezentovány vektorem $u = (\alpha - 1, \beta)^T$. V tomto zápisu jsou identické transformace reprezentovány nulovým vektorem. Textura v obrázku je pak definována pomocí rovnice:

$$g_{im} = T_u(\bar{g} + P_g b_g) = (1 + u_1)(\bar{g} + P_g b_g) + u_2 1 \tag{3.28}$$

3.5.2 Kombinace obou modelů

Model tvarů a model textury zle sloučit do jednoho modelu pomocí vektorů parametrů b_s a b_g. Aby se vytvořil vztah mezi tvarem a texturou, použijeme PCA na získaná data. Pro každá trénovací data bude vygenerován propojovací vektor:

$$b = \begin{pmatrix} W_s b_s \\ b_s \end{pmatrix} = \begin{pmatrix} W_s P_s^T (x - \bar{x}) \\ P_g^T (g - \bar{g}) \end{pmatrix} \tag{3.29}$$

kde W_s je diagonální matice vah pro každý parametr ovlivňující tvar. Po použití PCA na tyto vektory dostaneme následující model:

$$b = P_c c \tag{3.30}$$

kde P_c jsou vlastní vektory a c je vektor, který ovlivňuje jak tvar, tak i úrovně šedi (texturu) modelu. Od této chvíle mají parametry tvaru i textury průměr nula.

Celý model pak lze deformovat pomocí rovnic:

$$x = \bar{x} + P_s W_s^{-1} P_{cs} c \tag{3.31}$$

$$g = \bar{g} + P_g P_{cg} c \tag{3.32}$$

kde

$$P_c = \begin{pmatrix} P_{cs} \\ P_{cg} \end{pmatrix} \tag{3.33}$$

Nebo pokud rovnice více zobecníme, pak:

$$x = \bar{x} + Q_s c \tag{3.34}$$

$$g = \bar{g} + Q_s c \tag{3.35}$$

kde

$$Q_s = P_s W_s^{-1} P_{cs} \tag{3.36}$$

$$Q_g = P_g P_{cg} \tag{3.37}$$

3.6 Active Appearance Model (AAM)

Algoritmus ASM dokáže nalézt body v novém obrázku za pomocí modelu tvarů, a to na základě tvarových omezení společně s informací o struktuře obrázku v malém okolí bodu. Kdežto algoritmus AAM pracuje se SMA, který kromě tvarových omezení obsahuje i informace o textuře z celého obrázku. Proto je AAM obecně schopen dosahovat lepších výsledků než ASM. Tato podkapitola vychází z [11] [23].

3.6.1 Přehled algoritmu AAM

Algoritmus lze interpretovat jako optimalizační problém, ve kterém se snažíme minimalizovat rozdíl mezi novým obrázkem a vygenerovaným obrázkem pomoci SMA. Rozdílový vektor δI může být definován jako:

$$\delta I = I_i - I_m \tag{3.38}$$

kde I_i je vektor intenzity barvy v obrázku a I_m je vektor intenzity barvy pro obrázek vygenerovaný pomocí současně nastavených parametrů.

Pro nalezení nejlepší shody mezi modelem a obrázkem je potřeba minimalizovat velikost rozdílového vektoru $\Delta = |\delta I|^2$ pomocí nastavování parametrů modelu c. Jelikož SMA může obsahovat

mnoho parametrů, tak se jedná o obtížný mnohodimenzionální optimalizační problém. Avšak víme, že každý pokus o nalezení shody mezi modelem a novým obrázkem ve skutečnosti odpovídá optimalizačnímu problému. Pak prostorový vzor δI kóduje informaci o tom, jak by měly být změněny parametry modelu, aby nastala co nejlepší shoda. Řešení tohoto problému lze rozdělit na dvě části. Zaprvé naučit se vztah mezi δI a chybou parametrů modelu δc, zadruhé pak použití těchto znalostí v iterativním algoritmu pro minimalizaci Δ.

3.7 One Millisecond Face Alignment with an Ensemble of Regression Trees

V této podkapitole bude stručně popsán algoritmus, který je použit v knihovně Dlib pro detekci charakteristických bodů v obličeji. Tento algoritmus navrhli Vahid Kazemi a Josephine Sullivan a je podrobně popsán v [13]. Algoritmus používá pro detekci charakteristických bodů v obličeji regresní stromy, pomocí kterých hledá v řídké podmnožině pixelů intenzit. Takový algoritmus dosahuje vysoké kvality a dokáže pracovat v reálném čase. Použité regresní stromy obsahují oproti klasické implementaci dva klíčové prvky.

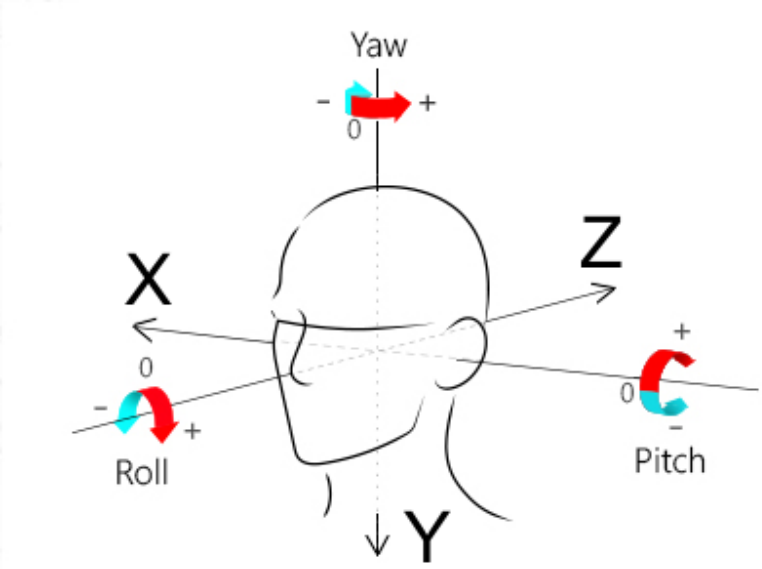
Prvním z nich se zabývá indexováním intenzity pixelů vzhledem k aktuálnímu odhadu. Získané charakteristické body se mohou značně měnit důsledkem tvarové deformace a dalších faktorů jako například změně osvětlení. Proto je obtížné najít pomocí těchto bodů přesný odhad tvaru. Problém nastává v tom, že na jednu stranu potřebujeme kvalitní charakteristické body k přesnému odhadu tvaru, na druhou stranu potřebujeme kvalitní odhad tvaru k získání odpovídající charakteristických bodů. Tento problém se řeší iterativním způsobem. Namísto regrese tvarových parametrů založených na charakteristických bodech získaných z globálního souřadnicového systému obrázku, je obrázek transformován do normalizovaného souřadnicového systému, který je založen na aktuálním odhadu tvaru. Následně je pomocí charakteristických bodů aktualizován vektor tvarových parametrů. To se opakuje, dokud dochází k významné změně parametrů.

Druhým klíčovým prvkem je, jak vyřešit problém predikce. Při testování musí algoritmus zarovnání odhadnout tvar (mnohodimenzionální vektor) a najít nejlepší shodu mezi obrazovými daty a modelem tvaru. Tento problém je nekonvexní s mnoha lokálními maximy. Většina algoritmů řeší tento problém tak, že předpokládají, že odhadovaný tvar musí ležet v lineárním podprostoru, který lze určit například pomocí nalezení hlavních komponent trénovacích tvarů. Tento předpoklad může výrazně snížit počet potenciálních tvarů zvažovaných během predikce a může pomoci se vyhnout lokálnímu maximu.

Spojením těchto dvou prvků vznikla efektivní regresní učící funkce. Ta je optimalizovaná pomocí vhodné ztrátové funkce a výběrem charakteristických bodů v datovém režimu. Konkrétně se každý regresor učí prostřednictvím *gradient boostingu* pomocí chyby čtverců ztrátové funkce. Jedná se o stejnou ztrátovou funkci, kterou se snažíme při testování minimalizovat. Řídká sada pixelů, která se používá jako vstup regresoru, je vybraná pomocí *gradient boostingu* a předchozí pravděpodobnosti vzdálenosti dvojic vstupních pixelů. Předcházející distribuce umožňuje *boostingu* efektivně prozkoumat velké množství relevantních charakteristických bodů. Výsledkem je kaskáda regresorů, která dokáže lokalizovat charakteristické body v obličeji, pokud je inicializována pomocí průměrné pozice obličeje. [13]

4 Natočení hlavy vůči kameře

Tato kapitola se bude zabývat výpočtem natočení hlavy vůči kameře. Nejprve budou vysvětleny pojmy, které jsou nezbytné pro pochopení dané problematiky, konkrétně se bude jednat o antropometrii hlavy (kapitola 4.1) a signifikantní rysy na obličeji (kapitola 4.2). V následující části pak budou popsány dvě konkrétní metody pro výpočet úhlů natočení hlavy (kapitoly 4.3 a 4.4). Pro vyjádření výsledného natočení hlavy použiji Eulerovy úhly. Výsledkem tedy budou tři úhly, pro každou osu jeden. Pro pojmenování jednotlivých úhlů použiji anglickou terminologii známou z letectví. Tedy rotaci kolem osy x budu nazývat *pitch*, rotaci kolem osy y budu nazývat *yaw* (v češtině někdy též označované jako azimut) a rotaci kolem osy z budu nazývat *roll*. Model hlavy se znázorněním jednotlivých os lze vidět na obrázku 4.1.



Obrázek 4.1: Model hlavy s třemi vyznačenými osami [25].

4.1 Antropometrie hlavy

Antropometrie jsou metody a techniky měření lidského těla, založené na standardizaci používaných bodů, rozměrů a nástrojů, což zajišťuje reprodukovatelnost i srovnatelnost různých antropometrických výzkumů lidských populací. K sjednocení metodik a používaných nástrojů došlo na sjezdu v Monaku (1906) a Ženevě (1912). Vznik a rozvoj antropometrie souvisí se zájmem o měření lidského těla a o popis tvaru lebky a s četnými pokusy o matematické vyjádření jejího tvaru. Počátky antropometrie sahají až do 17. století. [26] [27] [28]

Antropometrie se dělí na osteometrii, která se zabývá měřením a rekonstrukcí proporcí těla člověka na základě rozměrů jeho kosterních pozůstatků, a somatometrii, která se zabývá měřením proporcí těla žijícího člověka k účelům antropologickým, lékařským, statistickým aj. [26] Tato práce se dále bude zabývat výhradně somatometrii.

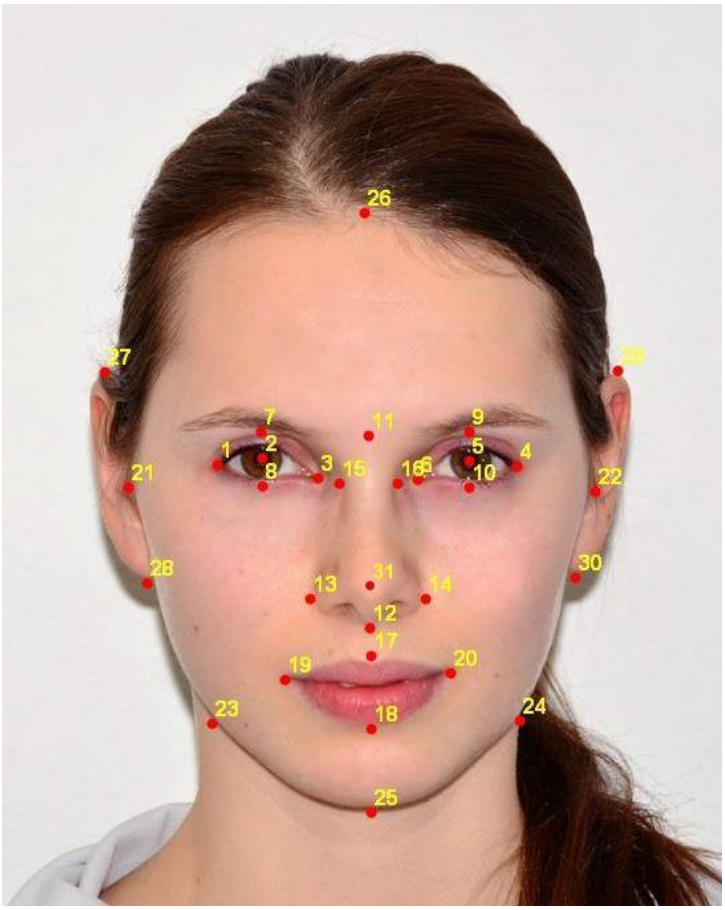
Podmnožinou antropometrie, která se zabývá metodami k měření jednotlivých částí lebky, se nazývá kraniometrie. [26]

V knize Human Engeneering Design Data [29] jsou podrobně provedena měření všech měřitelných rozměrů na obličeji. Podrobné tabulky antropometrie hlavy z této knihy jsou součástí přílohy B, ze kterých bude čerpáno v následujících podkapitolách.

4.2 Signifikantní rysy obličeje

Jako nejvhodnější a nejlépe detekovatelné rysy v obličeji lze považovat tzv. somatometrické body. Tyto body jsou definovány Prokopcem v knize Antropologie [30]. Jsou to body, které se nacházejí na těle, představující stejnojmenné body na kostře promítnuté na povrch těla.

V této práci budou popsány somatometrické body, které se nacházejí na obličeji a jsou snadno detekovatelné při pohledu zepředu. Každý bod má svůj název odvozený z latiny nebo řečtiny a zkratku. Většina bodů se na obličeji nachází dvakrát, a to na levé a pravé straně obličeje. Všechny popisované somatometrické body jsou vyznačeny na fotografii obličeje na obrázku 4.2. Mezi vybrané somatometrické body patří [30] [31]:



Obrázek 4.2: Fotografie obličeje s vyznačenými somatometrickými body [31].

- **Ektokantion (ex)** – bod na vnější straně oka na spojnici obou víček, vnější koutek oka. (body 1 a 4 z obrázku 4.2).
- **Pupila (pu)** – střed zornice oka (body 2 a 5).
- **Entokantion (en)** – bod ve vnitřním koutku oka, kde se stýká okraj horního a dolního víčka (body 3 a 6).

- **Palpebra superior (pas)** – bod, který vznikne jako průsečík linie procházející středem zornice s okrajem horního víčka. (body 7 a 9).
- **Palpebra inferior (pai)** – bod, který vznikne jako průsečík linie procházející středem zornice s okrajem dolního víčka (body 8 a 10).
- **Nasion (n)** – bod ležící v mediánní rovině (rovině procházející vertikálně středem obličeje) na kořenu nosu na horním okraji nosních kůstek (bod 11).
- **Subnasale (sn)** – nejníže viditelný bod nosu v mediální rovině. Pokud nelze na fotografiích bod vidět, je umístěn na spodním okraji nosního hrotu (bod 12).
- **Alare (al)** – bod, který se nachází nejvíce od mediální roviny na nosním křídle (body 13 a 14).
- **Radix nasi (ran)** – body ležící na kořenu nosu, které udávají hranici nejužší části nosního kořene (body 15 a 16).
- **Labrale superius (ls)** – průsečík tangenty proložené horním okrajem červeně rtu a mediánní roviny (bod 17).
- **Labrale inferius (li)** – bod ležící v mediální rovině na spodní hranici červeně dolního rtu (bod 18).
- **Cheilion (ch)** – bod ležící v koutku úst v místě styku hranice červeně horního a dolního rtu (body 19 s 20).
- **Zygion (zy)** – bod na jařmovém oblouku ležící nejdále od středu obličeje (body 21 a 22).
- **Gonion (go)** – bod na úhlu dolní čelisti, který leží nejvíce dole a nejdále od středu obličeje (body 23 a 24).
- **Gnathion (gn)** – bod ležící v mediánní rovině na dolním okraji brady nejvíce dole (bod 25).
- **Trichion (tr)** – bod ležící v mediální rovině na vlasové hranici. (bod 26).
- **Superaurale (sa)** – bod, který se nachází nejvíce nahoře na uchu (body 27 a 29).
- **Subaurale (sba)** – bod, který se nachází na spodním okraji ušního lalůčku ležící nejvíce dole (body 28 a 30).
- **Pronasale (prn)** – bod ležící na špičce nosu nejvíce vpředu (bod 31).

4.3 Určení natočení hlavy na základě antropometrických vlastností hlavy

Tato metoda vychází ze znalostí o antropometrii hlavy. Bude provádět detekci natočení hlavy pomocí charakteristických bodů v obličeji. Proto je nezbytně nutné, aby před tímto krokem byly nejprve detekovány obličeje a na těchto obličejích byly detekovány všechny předem definované body z šablony. Vzorový obličej s 31 detekovanými očíslovanými body je vidět na obrázku 4.2. Všechny výpočty v následujících podkapitolách budou předpokládat šablonu s takto rozmístěnými a očíslovanými body. Výsledné úhly natočení hlavy vůči kameře se budou počítat pro každou osu zvlášť.

4.3.1 Určení natočení hlavy v ose z (roll)

K detekci natočení hlavy v ose z je nejvhodnější použít dva body na obličeji nacházející se ve stejné výšce, pokud je natočení hlavy v ose z nulové. Jako nejvhodnější se zdají být body levého a pravého vnějšího koutku oka (ektokantion (ex) – body 1 a 4 z obrázku 4.2), případně body levého a pravého středu zornice oka (pupila (pu) – body 2 a 5). Jako další vhodné body by mohly být zvoleny levý a pravý koutek úst, ovšem oči jsou nejméně ovlivněny mimikou obličeje a je u nich malá pravděpodobnost chybné detekce.

Pokud budou použity body 1 a 4, pak výsledný úhel natočení γ se spočte pomocí arkus tangens takto:

$$\gamma = tan^{-1}\frac{y_4 - y_1}{x_4 - x_1} \tag{4.1}$$

kde x_1 je x-ová souřadnice bodu 1 a y_1 je y-ová souřadnice bodu 1.

Pokud budeme předpokládat počátek souřadnicového systému obrázku (tedy bod o souřadnicích $[0,0]$) vlevo nahoře, jak je tomu v OpenCV, pak pokud vyjde úhel γ kladný, tak je obličej natočený po směru hodinových ručiček z pohledu pozorovatele, a pokud vyjde záporný, tak je natočený proti směru hodinových ručiček z pohledu pozorovatele.

4.3.2 Určení natočení hlavy v ose y (yaw)

Další v pořadí je detekce natočení hlavy v ose y. Jako body vhodné k této detekci lze vybrat bod ležící pod špičkou nosu (bod 12) a dva body ležící na obrysu obličeje ve stejné výšce (body 21 a 22). Nejprve je potřeba si spočítat jaký tvar má průměrná hlava. Pro výpočet průměrného tvaru hlavy bude použita příloha A, kde jsou zobrazeny jednotlivé antropometrické hodnoty pro lidskou hlavu (z knihy Human engeneering design data [29]). Při využití těchto dat jsem vždy použil hodnotu mediánu pro dané měření (v tabulce je to hodnota 50. percentilu). A jelikož jsou v tabulce zvlášť hodnoty pro muže a ženy, tak výslednou hodnotu určím jako průměr mužského a ženského mediánu.

Dle tabulky 6 je mediánová vzdálenost mezi ušima 13,9 cm. Dle tabulky 12 je mediánová vzdálenost mezi bodem pod nosem a nejvzdálenějším místem vzadu na hlavě 19,55 cm. Pokud tedy budeme považovat, že polovina této vzdálenosti je střed hlavy, pak od středu hlavy k nosu to je 9,775 cm. Ve skutečnosti nás nezajímají přesné rozměry hlavy, ale poměr vzdálenosti mezi ušima ke vzdálenosti od středu hlavy ke špičce nosu. Tento poměr si zaznamenáme jako konstantu k a vypočítáme ji takto:

$$k = \frac{9{,}775}{13{,}9} = 0{,}703 \tag{4.2}$$

K určení natočení hlavy v ose y je potřeba zjistit, jak moc se bod nacházející se pod špičkou nosu vychyluje od středu hlavy. Nejprve je potřeba přesunout bod nacházející se pod špičkou nosu na přímku, která je tvořena dvěma body na obrysu obličeje. K tomu je potřeba si spočítat přímku mezi dvěma body na obrysu obličeje (body 21 a 22). Přímka bude spočítaná ve směrnicovém tvaru $y = kx + q$. Směrnice přímky se spočítá:

$$k_1 = \frac{y_{22} - y_{21}}{x_{22} - x_{21}} \tag{4.3}$$

Pokud známe směrnici přímky a alespoň jeden bod na přímce, pak lze již jednoduše spočítat koeficient q. Pro výpočet koeficientu q_1 použijeme např. bod 21, pak rovnice bude vypadat takto:

$$q_1 = y_{21} - k_1 x_{21} \tag{4.4}$$

Následně je potřeba spočítat rovnici přímky, která je kolmá k předchozí přímce a zároveň prochází bodem nacházející se pod špičkou nosu (bod 12). Směrnici nové přímky spočítáme jako převrácenou hodnotu směrnice první přímky s opačným znaménkem takto:

$$k_2 = -\frac{1}{k_1} \tag{4.5}$$

Koeficient q_2 se spočte obdobně jako v předchozím případě:

$$q_2 = y_{12} - k_2 y_{12} \tag{4.6}$$

Potom je na řadě spočítat průsečík obou přímek. X-ová souřadnice nově vzniklého bodu se spočte takto:

$$x_p = \frac{q_2 - q_1}{k_1 - k_2} \tag{4.7}$$

Y-ová souřadnice se pak může spočítat z první nebo druhé přímky, pro první přímku bude vypadat rovnice takto:

$$y_p = k_1 x_p + q_1 \tag{4.8}$$

V tuto chvíli již máme všechny tři body na jedné přímce. Nyní je na řadě zjistit, jak moc se bod ležící pod špičkou nosu blíží k levému nebo pravému kraji obličeje a na základě znalostí o antropometrii hlavy pak určit pod jakým úhlem je obličej v ose y natočen.

Nejprve si podle zjištěné šířky obličeje odvodíme pomocí konstanty k vzdálenost l od středu hlavy ke špičce nosu:

$$l = \sqrt{((x_{22} - x_{21})^2 + (y_{22} - y_{21})^2)} * k \tag{4.9}$$

Dále je potřeba si dopočítat střed hlavy s. Ten spočítáme jako střed mezi levým a pravým okrajem obličeje:

$$x_s = \frac{x_{22} + x_{21}}{2} \tag{4.10}$$

$$y_s = k_1 x_s + q_1 \tag{4.11}$$

Pak již lze vypočítat výsledný úhel β natočení hlavy v ose y. K výpočtu využijeme faktu, že se hlava otáčí kolem svého středu a že vzdálenost špičky nosu od středu je pořád konstantní, proto špička nosu opisuje kružnici kolem středu hlavy. S těmito fakty pak již můžeme jednoduše vypočítat výsledný úhel β pomocí arkus sinus takto:

$$\beta = \sin^{-1} \frac{\sqrt{\left(x_p - x_s\right)^2 + \left(y_p - y_s\right)^2}}{l} \tag{4.12}$$

Pokud vyjde úhel β kladný, pak se hlava otáčí doprava z pohledu pozorovatele, a pokud vyjde záporný, tak se otáčí doleva z pohledu pozorovatele.

Místo bodu ležícím pod špičkou nosu se dá v podstatě použít jakýkoliv jiný bod ležící ve středu obličeje. Celý výpočet by byl v podstatě stejný, jen by se lišila konstanta k.

4.3.3 Určení natočení hlavy v ose x (pitch)

Poslední v pořadí je určení natočení hlavy v ose x (*pitch*). Hodnota *pitch* se určí podle poměrné vzdálenosti špičky nosu (bod 31) mezi očima a ústy. Bod reprezentující oči spočtu jako bod ležící uprostřed mezi levým a pravým koutkem oka (body 1 a 4). X-ová souřadnice tohoto bodu se spočte podle vzorce:

$$x_{so} = \frac{x_4 - x_1}{2} \tag{4.13}$$

Y-ová souřadnice se pak spočte takto:

$$y_{so} = \frac{y_4 - y_1}{2} \tag{4.14}$$

Obdobným způsobem se spočítá bod, který se nachází uprostřed mezi levým a pravým koutkem úst (body 19 a 20), pomocí vzorců (4.13) a (4.14).

Nyní je potřeba přesunout bod nacházející se na špičce nosu do jedné přímky mezi body reprezentující střed očí a úst. Tato část výpočtu je podobná jako u výpočtu *yaw*. Nejprve je potřeba si spočítat směrnicový tvar přímky procházející body reprezentující střed očí a úst. Směrnice přímky se spočítá pomocí vzorce (4.3) a koeficient q se pak spočítá pomocí vzorce (4.4). Následně je potřeba spočítat rovnici přímky, která je kolmá k předchozí přímce a zároveň prochází bodem nacházejícím se na špičce nosu. Směrnici nové přímky se spočítá jako převrácená hodnota směrnice první přímky s opačným znaménkem dle rovnice (4.5). Koeficient q se pak spočítá pomocí vzorce (4.6). Poté je potřeba spočítat průsečík obou přímek. X-ová souřadnice průsečíku se spočítá pomocí rovnice (4.7), y-ová souřadnice se pak spočítá pomocí rovnice (4.8).

V tuto chvíli již leží všechny tři body na jedné přímce. Pokud si tyto body představíme z profilu, tak nám vytvoří trojúhelník. Tento trojúhelník spojuje bod reprezentující oči (S_o), bod reprezentující ústa (S_u) a bod reprezentující špičku nosu (S_n). Tento trojúhelník je zobrazen na obrázku 4.3. Na základě antropometrických vlastností hlavy budou spočteny průměrné velikosti jednotlivých stran trojúhelníku, čímž bude zjištěn tvar trojúhelníku. K výpočtu jednotlivých stran budou použity tabulky z přílohy A, konkrétně pak budou použity hodnoty mediánu pro jednotlivé rozměry.

Dle tabulek 9 a 10 vychází vzdálenost mezi očima a ústy (v_2) 7,2 cm. Dle tabulek 12 a 8 vychází hloubka nosu vůči očím (u_1) 3,75 cm. Vzdálenost mezi špičkou nosu a ústy z tabulek vyčíst nelze, ale lze ji vyčíst z natrénované šablony, jako poměr mezi očima a ústy ke špičce nosu a ústy. Tento poměr vychází 0,46. Pak tedy vzdálenost mezi špičkou nosu a ústy vychází 7,2*0,46 a to se rovná 3,312 cm. Nyní již známe přesný tvar trojúhelníku.

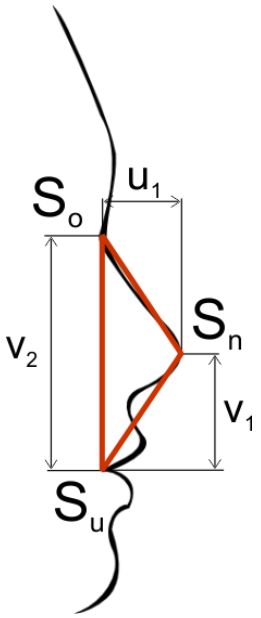
Pokud tento trojúhelník budeme rotovat kolem bodu S_u o úhel α, pak jsme schopni na základě známého úhlu α dopočítat y-ové souřadnice bodů S_o a S_n, a to pomocí vzorců:

$$y'_{sn} = u_1 \sin\alpha + v_1 \sin\alpha \tag{4.15}$$

$$y'_{so} = v_2 \cos\alpha \tag{4.16}$$

Ovšem hodnoty y'_{sn} a y'_{so} neznáme a ani nejsme z obrazu schopni zjistit jejich přesnou hodnotu, ale zato můžeme zjistit poměr $\frac{y'_{so}}{y'_{so}}$. Proto dané rovnice je potřeba upravit tak, ať je z ní jedna rovnice, která obsahuje člen $\frac{y'_{so}}{y'_{so}}$. Proto tedy obě strany rovnice vydělíme hodnotou o velikosti y'_{so} a dostaneme jednu rovnici tvaru:

$$\frac{y'_{sn}}{y'_{so}} = \frac{u_1 \sin\alpha + v_1 \cos\alpha}{v_2 \cos\alpha} \tag{4.17}$$



Obrázek 4.3: Obličej z profilu s vyznačeným trojúhelníkem.

Nás zajímá výsledný úhel α. Po úpravách dostáváme rovnici:

$$\alpha = \tan^{-1} \frac{\frac{y'_{sn}}{y'_{so}} - \frac{v_1}{v_2}}{\frac{u_1}{v_2}} \tag{4.18}$$

kde $\frac{y'_{so}}{y'_{so}}$ vyjadřuje poměr vzdálenosti mezi špičkou nosu a ústy ke vzdálenosti mezi očima a ústy. Jelikož se jedná o poměr, proto můžou být hodnoty y'_{sn} a y'_{so} vyjádřeny i v jiných jednotkách než zbytek rovnice (například v pixelech).

Pokud tedy za y'_{sn} a y'_{so} dosadíme y-ovou vzdálenost mezi špičkou nosu a pusou, respektive očima a pusou, získanou odečtením ze snímku a za v_1, v_2 a u_1 dosadíme předem vypočítané hodnoty, pak dosazením do rovnice (3.34) dostaneme rovnici:

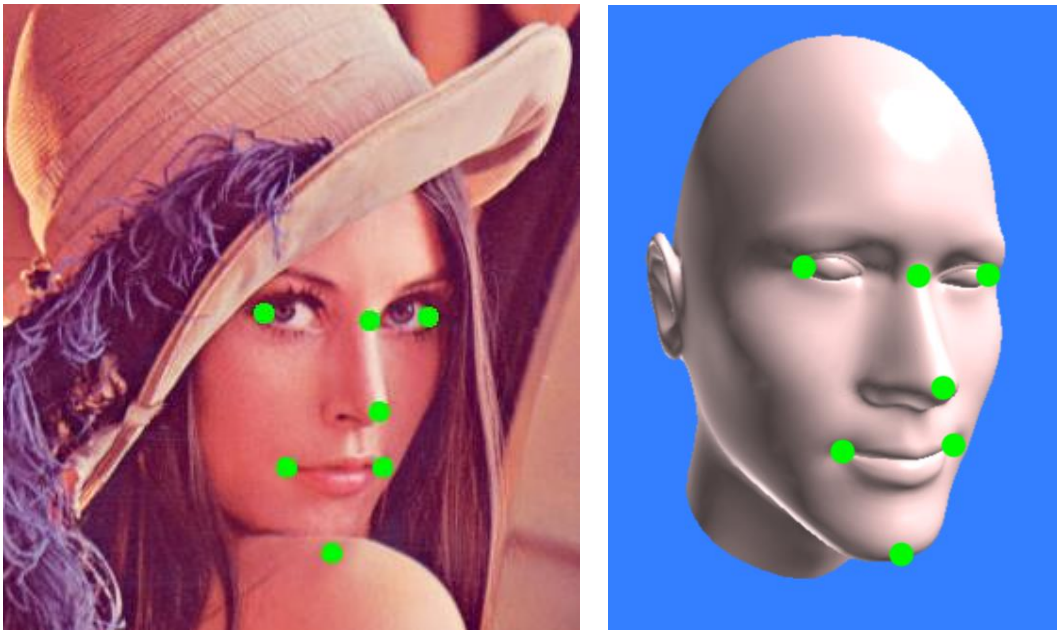
$$\alpha = \tan^{-1} \frac{\frac{y_{sn} - y_{sp}}{y_{so} - y_{sp}} - \frac{3{,}312}{7{,}2}}{\frac{3{,}75}{7{,}2}} \tag{4.19}$$

Úhel α je již výsledná hodnota rotace hlavy kolem osy x (*pitch*). Pokud vyjde úhel α kladný, pak se hlava otáčí nahoru, a pokud vyjde záporný, tak se hlava otáčí dolů.

4.4 Detekce natočení hlavy na základě natáčení modelu hlavy

Druhou metodou detekce natočení obličeje je metoda, která pro svou práci potřebuje model hlavy [32] [33]. Při použití této metody je potřeba nejprve si zvolil N bodů, které se nacházejí na obličeji, jež jsou detekovatelné při pohledu zepředu. Tyto body je potřeba vyhledat na modelu obličeje, který má střed modelu umístěný ve středu souřadnicového systému (bod [0,0,0]). Ty stejné body je potřeba nalézt i na vstupním 2D obrázku s detekovaným obličejem. K detekci bodů v obličeji je vhodné použít některý z algoritmů v kapitole 3. Poté je potřeba pomocí posunů a rotací transformovat model obličeje tak, aby jeho body ve 2D (bez souřadnice z) co nejlépe pasovaly na body detekované na 2D obrázku. Zjištěná rotační matice nám pak udává výsledné natočení hlavy. Algoritmy, které se zabývají napasováním bodů ve 3D na body ve 2D, se nazývají *Perspective-n-Point* (PnP). Existuje několik různých variant těchto algoritmů.

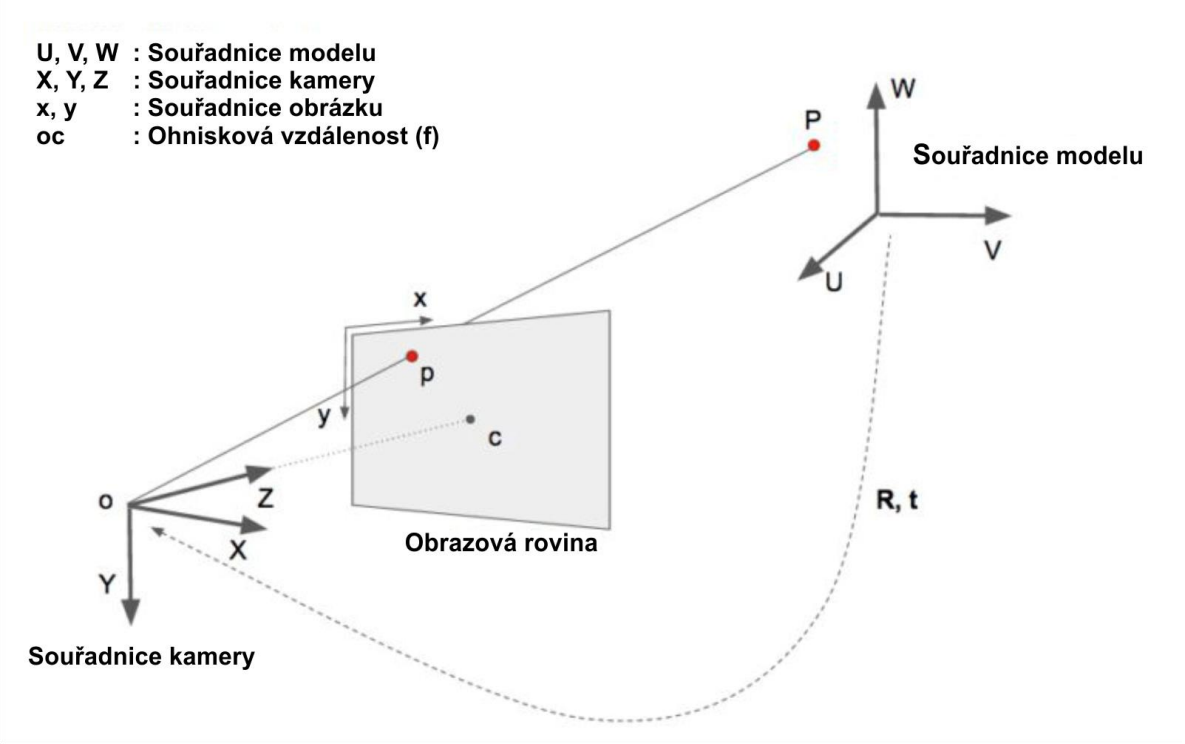
Na obrázku 4.4 je zobrazena fotografie obličeje a model hlavy. Na obou obrázcích jsou zvýrazněny stejné charakteristické body, model hlavy je natočen pomocí PnP tak, aby jeho body ve 3D co nejvíce odpovídaly bodům ve 2D na fotografii.



Obrázek 4.4: Nalevo je vstupní fotografie s vyznačenými charakteristickými body v obličeji, napravo je model hlavy natočený tak, aby jeho body co nejlépe korespondovaly na body na fotografii.

4.4.1 Princip algoritmu Perspective-n-Point (PnP)

Základní princip algoritmu Perspective-n-Point (PnP) vychází z následujících poznatků. Máme tři souřadnicové systémy. Zaprvé je to 3D souřadnicový systém modelu, pak je to 3D souřadnicový systém kamery a na závěr je to 2D souřadnicový systém obrázku. Pokud známe rotaci a posun modelu, můžeme 3D body ze souřadnicového systému modelu transformovat na 3D body souřadnicového systému kamery. 3D body ze souřadnicového systému kamery mohou být promítnuty na obrazovou rovinu (souřadnicový systém obrázku) pomocí vnitřních parametrů kamery (ohnisková vzdálenost, optický střed atd.). Všechny souřadnicové systémy a vztahy mezi nimi jsou zobrazeny na obrázku 4.5.



Obrázek 4.5: Zobrazení vztahu mezi jednotlivými souřadnicovými systémy [33].

Na obrázku 4.5 je o střed souřadnicového systému kamery a zobrazená rovina je obrazová rovina. Naším cílem je zjistit pomocí jakých rovnic se provede projekce p 3D bodu P na obrazovou rovinu. Předpokládejme, že známe polohu (U, V, W) 3D bodu P v souřadnicovém systému modelu. Jestliže známe rotaci R (3x3 matice) a posun t (3x1 vektor) souřadnic modelu vzhledem k souřadnicím kamery, můžeme dopočítat pozici (X, Y, Z) bodu P v souřadnicovém systému kamery za pomocí následující rovnice:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = R \begin{bmatrix} U \\ V \\ W \end{bmatrix} + t \tag{4.20}$$

$$\Rightarrow \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = [R|t] \begin{bmatrix} U \\ V \\ W \\ 1 \end{bmatrix} \tag{4.21}$$

V rozšířené podobě pak tato rovnice bude vypadat takto:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} r_{00} & r_{01} & r_{02} & t_x \\ r_{10} & r_{11} & r_{12} & t_y \\ r_{20} & r_{21} & r_{22} & t_z \end{bmatrix} \begin{bmatrix} U \\ V \\ W \\ 1 \end{bmatrix} \tag{4.22}$$

Pokud známe dostatečný počet odpovídajících si bodů ((X, Y, Z) a (U, V, W)), pak se jedná o lineární systém rovnic, kde r_{ij} a (t_x, t_y, t_z) jsou neznámé a tyto neznámé lze jednoduše dopočítat. Avšak (X, Y, Z) jsme schopni zjistit až po zjištění změny měřítka a tedy nejedná se o lineární systém rovnic.

To, co v tuto chvíli známe, jsou 3D pozice bodů modelu v souřadnicovém systému modelu (U, V, W). Avšak neznáme pozice bodů v souřadnicovém systému kamery (X, Y, Z). Dále známe 2D pozice bodů v souřadnicovém systému obrázku (x, y). Pokud zanedbáme radiální zkreslení, pak se souřadnice (x, y) bodu p v souřadnicovém systému obrázku spočítají:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = s \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{4.23}$$

kde f_x a f_y jsou ohniskové vzdálenosti v x-ových a y-ových souřadnicích a (c_x, c_y) je optický střed. Pokud bychom brali v potaz radiální zkreslení, pak by byl výpočet o něco složitější.

Dále s je neznámá změna měřítka. V rovnici je obsažen, protože u obrázku neznáme jeho hloubku. Jestliže propojíme libovolný bod P ve 3D se středem kamery o, pak v místě protnutí tohoto paprsku s plochou obrázku se nachází bod p. Ovšem všechny body, které se nacházejí na paprsku spojujícím střed kamery s bodem P, se promítnou na stejné místo na obrázku. Když tedy použijeme rovnici (4.23) tak získáme (X, Y, Z) až po zjištění změny měřítka s.

Pokud vezmeme toto v potaz, tak úpravou rovnice (4.22) získáme rovnici:

$$s \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} r_{00} & r_{01} & r_{02} & t_x \\ r_{10} & r_{11} & r_{12} & t_y \\ r_{20} & r_{21} & r_{22} & t_z \end{bmatrix} \begin{bmatrix} U \\ V \\ W \\ 1 \end{bmatrix} \tag{4.24}$$

Tato rovnice lze vyřešit pomocí metody *Direct linear transformation* (DLT). DLT zle použít, když je rovnice téměř lineární, ovšem tato metoda je nepřesná, pokud neznáme změnu měřítka.

Samotné DLT není příliš přesné, a to kvůli následujícím důvodům. Prvním důvodem je, že rotace R má tři stupně volnosti, ale matice reprezentující použité DLT řešení má 9 členů. Pomocí DLT navíc nelze vytvořit rotační matici pomocí odhadnuté 3x3 matice. Dalším důvodem je, že DLT řešení neminimalizuje správnost objektivní funkce. V ideálním případě bychom chtěli minimalizovat chybu znovuprojekce popsanou níže.

Jak je uvedeno v rovnicích (4.22) a (4.23), jestliže známe správný posun a rotaci (matice R a t), dokážeme určit 2D pozice bodů, které odpovídají 3D pozicím bodů, a to pomocí projekce 3D bodů do 2D obrázku. Jinými slovy, pokud známe R a t, tak jsme schopni najít na obrázku bod p, pro každý 3D bod P.

Takže pokud známe 2D pozici charakteristických bodů v obličeji, tak se můžeme podívat na vzdálenost mezi promítnutými 3D body v obličeji a 2D body v obličeji odhadnutými z obrázku. Pokud jsou odhadované pozice a natočení modelu již velmi dobře určeny, tak pak budou 3D body promítnuté do obrazové roviny téměř přesně pasovat s 2D charakteristickými body v obličeji.

Pomocí DLT lze tedy najít přibližný posun a natočení modelu (matice R a t). Nativní způsob, jak zdokonalit DLT, je náhodně měnit posun a natočení (matice R a t) o malou hodnotu a sledovat, zda se chyba projekce snižuje. Pokud ano, můžeme nový odhad posunu a natočení přijmout. Toto můžeme iterativně opakovat, dokud daná chyba neklesne pod určitou mez. Tento postup bude fungovat, avšak bude pracovat velmi pomalu. Existují však metody, které dokáží iterativně měnit hodnoty posunu a rotací (matice R a t) tak, aby docházelo ke snižování chyby projekce. Jedna z takovýchto metod se nazývá Levenberg-Marquardtova optimalizace. Jedná se o iterační metodu, která řeší problém minimalizace sumy kvadrátu odchylek obecné nelineární funkce. Princip metody je založený na hledání globálního minima chyb minulých výstupů z modelované soustavy a výstupů z modelu přes paměť posledních hodnot. [32] [33]

4.4.2 Algoritmy PnP

Jak již bylo zmíněno, existuje několik různých algoritmů *Perspective-n-Poin,* které slouží k napasování bodů ve 3D na body ve 2D. V této podkapitole budou vyjmenovány některé z nich, a to ty, které jsou implementovány v OpenCV [34].

- **Iterative** – jedná se o implementaci, která vychází z výše uvedeného postupu, tedy z DLT s Levenberg-Marquardtovou optimalizací.
- **EPnP** – jedná se o implementaci, kterou představili F.Moreno-Noguer, V.Lepetit a P.Fua v dokumentu „EPnP: Efficient Perspective-n-Point Camera Pose Estimation“ [35].
- **DLS** – jedná se o metodu, která vychází z dokumentu od Joela A. Hesche a Stergiose I. Roumeliotise s názvem „A Direct Least-Squares (DLS) Method for PnP“ [36].
- **UPNP** – jedná se o metodu založenou na dokumentu od A.Penate-Sancheze, J.Andrade-Cetta, F.Moreno-Noguera s názvem “Exhaustive Linearization for Robust Camera Pose and Focal Length Estimation” [37].

5 Implementace

V této kapitole budou nejprve popsány použité nástroje (kapitola 5.1) a následně bude popsána samotná implementace aplikace. Nejprve se detekují obličeje (kapitola 5.2), v každém detekovaném obličeji se naleznou charakteristické body (kapitola 5.3) a na závěr se pro každý obličej určí samotné úhly natočení hlavy vůči kameře na základě lokalizovaných charakteristických bodů v obličeji (kapitola 5.4).

V současnosti se jedná o konzolovou aplikaci. Jako vstup lze zadat obrázek, video nebo výstup z webkamery. Výstup je buď zobrazen přímo na obrazovku, nebo je uložen do odpovídajícího souboru.

Pokud je na vstupu video nebo výstup z webkamery, tak je předem potřeba video rozdělit na jednotlivé snímky. Každý snímek se pak zpracovává nezávisle na ostatních snímcích.

5.1 Použité nástroje

K implementaci aplikace jsem se rozhodl použít programovací jazyk C++ a knihovny OpenCV (verze 3.2.) a Dlib.

Knihovna OpenCV (Open Source Computer Vision Library) [38] je open source knihovna počítačového vidění a softwaru pro strojové učení. Knihovna je dostupná pod licencí BSD, a proto ji lze zdarma používat a upravovat jak pro akademické, tak pro komerční účely. Knihovna OpenCV je dostupná pro programovací jazyky C++, C, Python a Java a podporuje operační systémy Windows, Linux, Mac OS, iOS a Android. OpenCV byla navržena tak, aby byla výpočetně efektivní a byl zde kladen velký důraz na real-time aplikace [38]. Tato knihovna značně ulehčí práci s obrazovými daty a jsou zde již implementovány některé algoritmy, které jsou potřeba pro tvorbu tohoto programu.

Druhou použitou knihovnou je knihovna Dlib [39]. Jedná se o moderní C++ nástroj obsahující zejména algoritmy pro strojové učení a nástroje pro tvorbu komplexního softwaru v C++ pro řešení problému reálného světa. Tuto knihovnu zle použít jak pro komerční, tak i pro akademické účely, a to například pro robotiku, vestavěné zařízení, mobilní zařízení, počítačové vidění atd. Dlib je poskytovaná pod licencí Open source, což umožňuje její bezplatné použití v jakékoliv aplikaci [39]. Po použití v mé aplikaci je tato knihovna vhodná zejména proto, že obsahuje již naimplementovaný algoritmus na detekci charakteristických bodů v obličeji.

Jazyk C++ jsem se rozhodl využít mimo jiné proto, že je podporovaný oběma knihovnami. Jeho výhodou oproti samotnému C je vyšší míra abstrakce a možnost využít objektově orientované knihovny.

Pro vývoj programu jsem použil vývojové prostředí Microsoft Visual Studio 2015, které jsem používal pod operačním systémem Windows 10.

5.2 Detekce obličeje

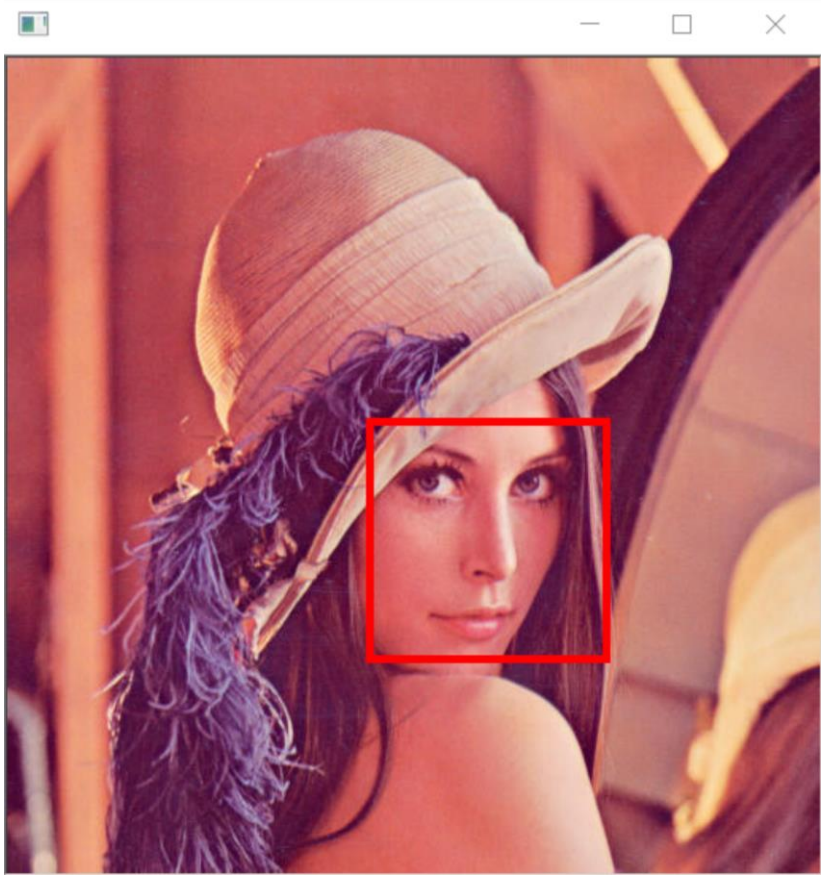
Detekci obličejů jsem se rozhodl implementovat pomocí dvou metod. Na základě provedených experimentů a zhodnocení úspěšnosti, použiji ve výsledném programu jednu z nich.

K implementaci první metody sloužící k detekci obličejů jsem použil knihovnu OpenCV. Tato knihovna již obsahuje implementaci algoritmu Viola-Jones a obsahuje také již natrénovaný kaskádový klasifikátor Haarových příznaků, což ulehčí značné množství práce. Samotná detekce pak probíhá pomocí metody `detectMultiScale()`, které je pomocí parametrů předán vstupní snímek a požadované nastavení a ona vrátí seznam obdélníků, které ohraničují detekované obličeje. U příslušné

metody lze nastavit, kolikrát se má změnit velikost podokna v každém měřítku, kolik sousedních podoken musí detekovat potenciální obličej, aby byl detekován jako obličej, a minimální a maximální velikost obličeje v pixelech.

Druhou možností, jak detekovat obličeje, kterou jsem se rozhodl použít, je využití implementace, která je obsažena v knihovně Dlib. Tato knihovna má implementovanou metodu, která je založena na histogramech orientovaných gradientů (HOG). Tato metoda ovšem neumožňuje nastavení jakýchkoliv parametrů a ani neumožňuje změnit natrénovaný klasifikátor.

Výstupem po této části je snímek, na němž jsou ohraničeny červeným obdélníkem všechny detekované obličeje. Ukázka aplikace po detekci obličejů je zobrazena na obrázku 5.1.

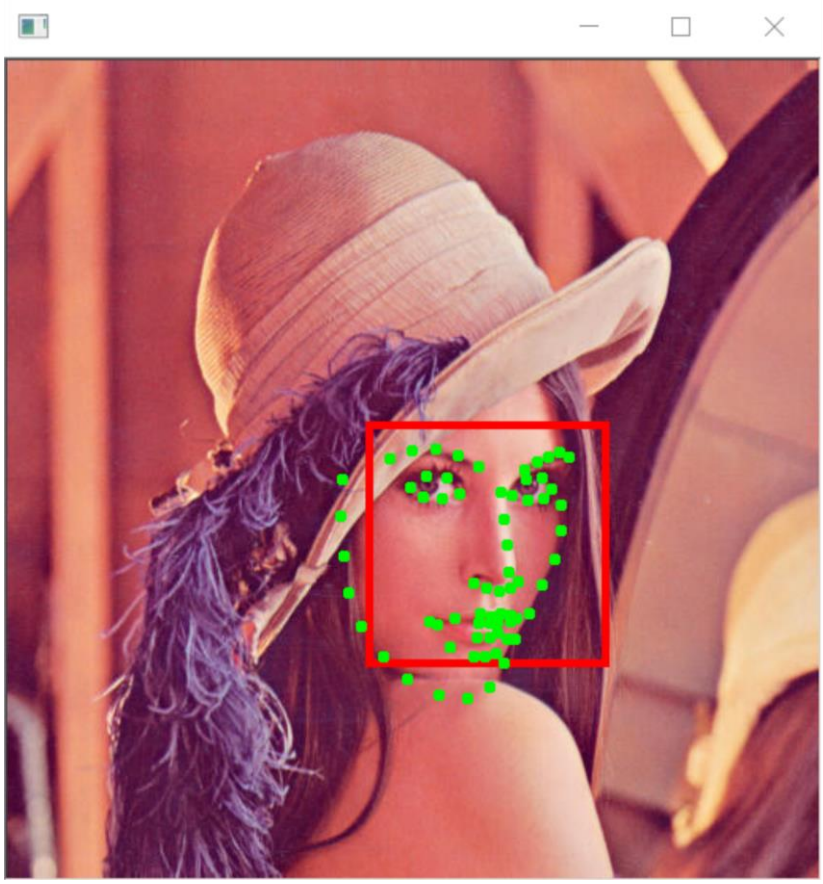


Obrázek 5.1: Ukázka aplikace s detektovaným obličejem.

5.3 Lokalizace charakteristických bodů v obličeji

K lokalizaci charakteristických bodů v obličeji jsem se rozhodl použít knihovnu Dlib. Ta má již naimplementovaný algoritmus sloužící k lokalizaci charakteristických bodů, konkrétně se jedná o algoritmus „One Millisecond Face Alignment with an Ensemble of Regression Trees“, který je popsán v kapitole 3.7. Tato knihovna již také obsahuje natrénovanou šablonu s 68 charakteristickými body na obličeji. Žádné další parametry k nastavení tato metoda již neobsahuje. Tato lokalizace se provádí pro každý detekovaný obličej zvlášť.

Charakteristické body jsou ve výstupním snímku zobrazeny jako zelené kruhy. Ukázka aplikace po lokalizaci charakteristických bodů v obličeji je zobrazena na obrázku 5.2.



Obrázek 5.2: Ukázka aplikace s lokalizovanými charakteristickými body v obličeji.

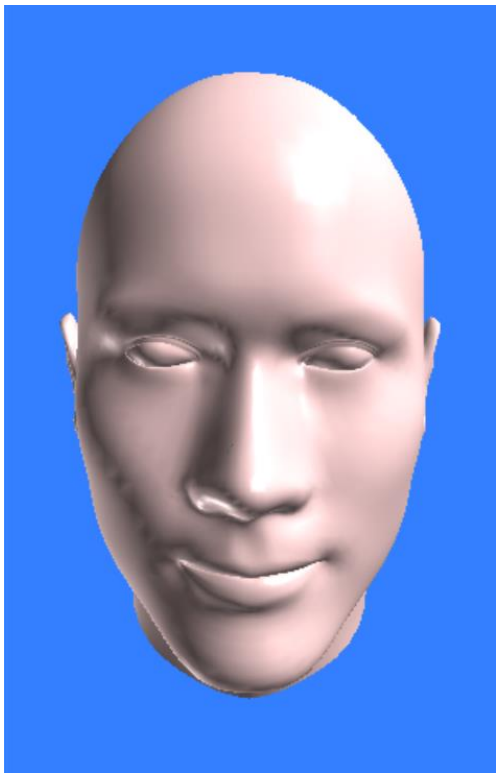
5.4 Určení natočení hlavy

Na základě detekovaných obličejů a lokalizovaných charakteristických bodů se může provádět výsledný výpočet úhlů natočení hlavy. To jsem implementoval pomocí dvou metod.

První metoda provádí výpočet určení natočení hlavy na základě antropometrických vlastností hlavy. Výpočet se zde provádí pro každou ze tří os zvlášť. Pro rotaci kolem každé osy je vytvořena jedna metoda. Tyto metody jsou naprogramovány přesně podle postupu popsaného v podkapitolách 4.3.1, 4.3.2 a 4.3.3.

Druhá metoda provádí výpočet natočení hlavy na základě natáčení modelu hlavy. Tato metoda vychází z podkapitoly 4.4. Pro tuto metodu je potřeba referenční model hlavy, který se bude natáčet tak, aby jeho body odpovídaly bodům detekovaným na obrázku. Jako referenční model hlavy jsem použil model použity v HeadViewer [40]. Referenční model hlavy použitý pro výpočet je zobrazen na obrázku 5.3.

Dále je potřeba si zvolit určitý počet charakteristických bodů na obličeji jako například vnější koutek oka, špičku nosu, koutek úst atd. Tyto body je potřeba lokalizovat na obličeji na obrázku pomocí postupu z kapitoly 6.3 a ze všech lokalizovaných bodů vybrat ty zvolené a uložit je do vektoru 2D bodů v předem daném pořadí. Následně je potřeba najít ty stejné body na referenčním modelu hlavy. Ve skutečnosti ovšem nepotřebujeme načítat celý model, ale stačí nám načíst pouze pozice 3D bodů, které reprezentují dané charakteristické body. K tomu lze použít jakýkoliv software na prohlížení 3D modelů. Pozice těchto bodů je potřeba uložit do stejně velkého vektoru jako u 2D bodů a sobě odpovídající body se musejí nacházet na stejné pozici v obou vektorech.



Obrázek 5.3: Referenční model hlavy [40].

V dalším kroku je potřeba nastavit parametry kamery, jako ohniskovou vzdálenost, optický střed a radiální zkreslení. Tyto parametry ovšem ve skutečnosti většinou neznáme, a proto je potřeba je přibližně dopočítat na základě předpokládaných vlastností kamery. Optický střed můžeme aproximovat pomocí optického středu obrázku, ohniskovou vzdálenost můžeme přibližně spočítat jako šířku obrázku v pixelech [41] a radiální zkreslení zanedbáme. Výsledná matice A s nastavením kamery pak má tvar:

$$A = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{5.1}$$

kde f je mnou spočítaná přibližná hodnota ohniskové vzdálenosti a (c_x, c_x) je mnou spočtený optický střed. Ještě je potřeba vytvořit vektor zkreslení, to však zanedbáváme, a proto se bude jednat o 4prvkový vektor, jehož všechny prvky budou nulové.

V tuto chvíli již známe vektor 2D souřadnic charakteristických bodů, vektor 3D souřadnic charakteristických bodů, matici kamery a vektor zkreslení, a tak můžeme přejít k samotnému výpočtu *Perspective-n-Point* (PnP). K tomu využiji knihovnu OpenCV. Tato knihovna obsahuje metodu `solvePnP()`, která slouží k výpočtu PnP. Vstupy této metody je vektor 2D souřadnic bodů, vektor 3D souřadnic bodů, matice kamery a vektor zkreslení. Dále je u metody „solvePnP“ potřeba nastavit „flag“, který nám udává, pomocí jakého algoritmu se bude PnP výpočet provádět. Na výběr jsou algoritmy „iterative“, „EPnP“, „DLS“ a „UPNP“. Všechny tyto algoritmy byly stručně popsány v kapitole 4.4.2. Výstupem této metody je pak vektor rotace a vektor posunu. Pomocí vektoru rotace ovšem nejsme schopni určit jednotlivé úhly, proto si jej musíme převést na rotační matici, a to pomocí metody Rodrigues(). Pokud již známe rotační matici R:

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{5.2}$$

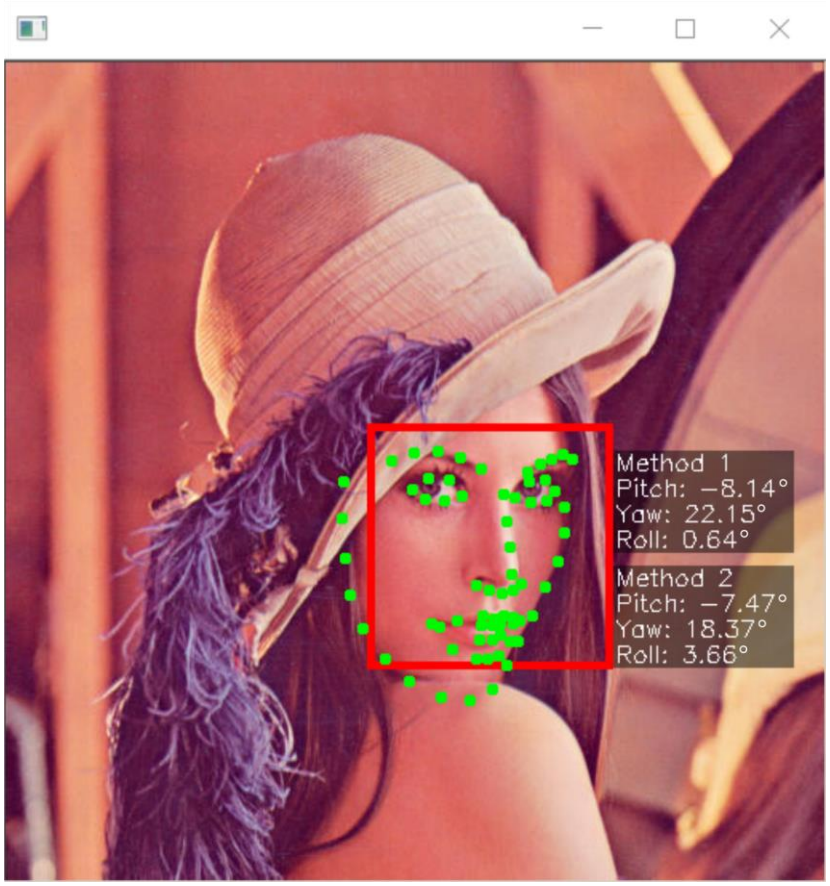
pak si z ní už můžeme spočítat konkrétní eulerovské úhly pomocí následujících rovnic [42]:

$$\theta_x = tan^{-1}\left(\frac{r_{32}}{r_{33}}\right) \tag{5.3}$$

$$\theta_y = tan^{-1}\left(\frac{-r_{31}}{\sqrt{r_{32}^2 + r_{33}^2}}\right) \tag{5.4}$$

$$\theta_z = tan^{-1}\left(\frac{r_{21}}{r_{11}}\right) \tag{5.5}$$

V tuto chvíli již známe hodnoty všech tří úhlů. Všechny výpočty úhlů probíhají v radiánech, pro člověka je však přirozenější reprezentace ve stupních, proto je nakonec ještě potřeba převést všechny vypočtené úhly v radiánech na stupně.



Obrázek 5.4: Ukázka aplikace s vypsanými výslednými úhly natočení hlavy pro obě metody.

Výstupem z této části tak budou pro obě metody hodnoty tří úhlů, které udávají natočení hlavy pro všechny tři osy. Výsledné úhly jsou pak vypsány do obrázku, pod kterými je vykreslen poloprůhledný obdélník. Případně mohou být výsledné úhly vypsány též do konzole. Ukázka finální aplikace s vyznačeným obličejem, vyznačenými charakteristickými body a vypsanými úhly pro obě metody je zobrazena na obrázku 5.4.

6 Testování

Testování bude rozděleno na dvě části. V první části budu testovat detekci obličejů (kapitola 6.2), porovnám obě implementované metody a budu se snažit najít ideální nastavení parametrů. V druhé části budu testovat samotné určování natočení hlavy vůči kameře (kapitola 6.3). Nejprve však budou popsány použité datasety (kapitola 6.1).

6.1 Použité datasety

K testování jsem použil dva referenční datasety. První dataset obsahoval videa s hlavami lidí, u nichž byly změřeny konkrétní Eulerovy úhly natočení hlavy, druhý dataset pak obsahuje fotografie s vyznačenými hlavami.

Jako referenční dataset, který obsahoval konkrétní úhly natočení hlav, jsem použil dateset od Gi4E [43]. Tento dataset obsahuje 120 videí, která zachycují 10 osob (6 mužů a 4 ženy), přičemž každá osoba natočila 12 videí. Každé video trvá 10 s a je tvořeno 300 snímky. Celkově je tedy celý dataset tvořen 36 000 snímky. Videa byla pořízena běžnou webkamerou v rozlišení 1280x720 pixelů. Ke každému snímku jsou v příslušném souboru uloženy konkrétní hodnoty úhlů natočení hlavy pro všechny tři osy. Přesná hodnota úhlů natočení hlavy byla určena pomocí senzorů umístěných na hlavě. Malou nevýhodou tohoto datasetu je, že všechny osoby, které se nacházejí na videích, jsou europoidní rasy. Ukázkové snímky z datasetu jsou zobrazeny na obrázku 6.1.

Obrázek 6.1: Ukázka snímků z datasetu od Gi4E [43].

Jako druhý referenční dataset jsem použil volně dostupnou databázi obrázků, která byla vytvořena na University of Massachusetts [44]. Tato databáze obsahuje obrovský počet fotografií, které jsou nashromážděné z internetu. K samotnému testování jsem využil předpřipravené sady obrázků, které obsahují náhodně vybrané obrázky z databáze společně s anotacemi obličejů. Použitá sada obrázků, nad kterou budu provádět experimenty, obsahuje 712 fotografií s 959 obličeji. Některé fotografie z tohoto datasetu jsou zobrazeny na obrázku 6.2.

6.2 Testování detekce obličejů

Každá lokalizace charakteristických bodů v obličeji a následné určení natočení hlavy se provádí vždy tam, kde byl před tím detekovaný obličej. Proto je potřeba mít kvalitní detektor obličejů, který detekuje co nejvíce správných obličejů a zároveň bude mít co nejmenší počet falešných detekcí. Navíc je potřeba,

aby byl detektor schopen detekovat obličeje v co možná největším úhlu. Takový detektor se pokusím najít pomocí testování.

V první řadě budu testovat kvalitu a rychlost detekce obličejů na běžných fotografiích a porovnám oba použité detektory. Testování budu provádět tak, že na sadě fotografií vyhledám obličeje použitím detektorů naimplementovaných v programu a ty pak porovnám se skutečnými obličeji, které se nacházejí na fotografii. Pro každou fotografii automaticky určím počet správných pozitivních detekcí (true positive - TP), nesprávně určených pozitivních detekcí (false positive - FP) a nesprávně určených negativních detekcí (false negative - FN). K tomu využiji dataset vytvořený v University of Massachusett s předpřipravenou sadou fotografií, která obsahuje 712 fotografií s 959 obličeji (kapitola 6.1). Fotografie s vyznačenými referenčními i detekovanými obličeji z použitého datasetu jsou zobrazeny na obrázku 6.2.

Obrázek 6.2: Ukázka obrázků z druhého testovacího datasetu. Červené obdélníky znázorňují detekce vypočtené programem a zelené obdélníky znázorňují referenční obličeje. Nalevo je obrázek se správnou detekcí, druhý zleva je obrázek s nesprávně určenou negativní detekcí, třetí zleva je obrázek s nesprávně určenou pozitivní detekcí a obrázek napravo obsahuje jak nesprávně určenou pozitivní detekci, tak nesprávně určenou negativní detekci. Obrázky použity z [44].

Nejprve je potřeba detekovat obličeje pomocí implementovaných detektorů a tyto detekované obličeje je poté potřeba porovnat s anotovanými obličeji z datasetu za účelem nalezení shodných obličejů. Problém je, že oba obdélníky určující obličej nebudou téměř nikdy úplně shodné. Proto jako správnou detekci beru tu, jejichž průnik obdélníků je alespoň 40 %. Toto procento se může zdát jako malé, ale problém je, že ony anotace určují celou hlavu osoby, kdežto detektor nachází pouze její obličej. Následně jsou pro každou fotografii spočteny správné pozitivní detekce, nesprávné pozitivní detekce a nesprávné negativní detekce a na konci testování je vypsán součet těchto hodnot pro všechny testované fotografie.

K vyhodnocení kvality detektorů použiji metriky precision, recall a F-measure. Precision udává pravděpodobnost, že náhodně vybraná detekce je správná. Vypočítá se pomocí vzoroce [45]:

$$precision = \frac{true\ positive}{true\ positive + false\ positive} \tag{6.1}$$

Recall udává pravděpodobnost, že náhodně vybraný obličej je detekován správně. Vypočítá se pomocí vzorce [45]:

$$recall = \frac{true\ positive}{true\ positive + false\ negative} \tag{6.2}$$

Pokud jsou vypočítány hodnoty precision a recall, je vhodné určit, jaký poměr těchto hodnot je nejvhodnější. K tomu se nejčastěji používá F-measure, což je harmonický průměr hodnoty precision a recall. Vzorec pro jeho výpočet vypadá takto [46]:

$$F = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{6.3}$$

Čím je hodnota F-measure blíže 1, tím je dané nastavení detektoru kvalitnější.

Nejprve provedu testování detekce obličejů pomocí detektoru Viola-Jones naimplementovaného pomocí knihovny OpenCV. U této implementace lze měnit hodnotu měřítka a počet sousedních detekcí, který je potřeba proto, aby byla potenciální detekce prohlášena za obličej. Testování provedu postupně pro 3 různé hodnoty změny měřítka, a to 1,05 a 1,1 a 1,2. V každém měřítku pak budu testovat parametr, který udává počet sousedních oken, který je potřeba k detekci obličeje. Tento parametr budu nastavovat na hodnoty od 1 do 20. Zjištěné hodnoty zapíši do tabulky a dopočtu k nim hodnoty precision a recall.

První testování jsem provedl pro hodnotu změny měřítka 1,05. Při nastavení této hodnoty trvala detekce předpřipravené sady obrázků průměrně 144 s. Při změně počtu oken nutných k detekci se doba detekce nemění, ta je závislá pouze na změně hodnoty měřítka. Naměřené hodnoty jsou zaznamenány v tabulce 6.1, kde PO je počet sousedních oken nutných k detekci, DO jsou detekované obličeje, TP (true positive) jsou správné pozitivní detekce obličeje, FP (false positive) jsou nesprávné pozitivní detekce obličeje, FN (false negative) jsou nesprávné negativní detekce a F je F-measure.

PO	DO	TP	FP	FN	Precision	Recall	F
1	1322	917	405	42	0,6936	0,9562	0,8040
2	1097	907	190	52	0,8268	0,9458	0,8823
3	1012	899	113	60	0,8883	0,9374	0,9122
4	970	891	79	68	0,9186	0,9291	0,9238
5	934	885	49	74	0,9475	0,9228	0,9350
6	914	875	39	84	0,9573	0,9124	0,9343
10	864	851	13	108	0,9850	0,8874	0,9336
20	796	794	2	165	0,9975	0,8279	0,9048

6.1: Tabulka detekce obličejů pomocí algoritmu Viola-Jones s krokem měřítka 1,05.

Následně jsem provedl testování s hodnotou měřítka 1,1. Pro toto měřítko trvala detekce všech obrázků průměrně 88 s. Naměřené hodnoty jsou zaznamenány v tabulce 6.2.

Poslední testování detekce obličejů pomocí algoritmu Viola-Jones jsem provedl s hodnotou změny měřítka 1,2. Zde byl průměrný čas detekce 67 s. Naměřené hodnoty jsou zaznamenány v tabulce 6.3.

Dále provedu testování detekce obličejů pomocí algoritmu HOG naimplementovaného pomocí knihovny Dlib. U této implementace nejdou měnit žádné parametry, takže pouze spustím testovací program s předpřipraveným datasetem nad danou implementací. Pak tedy vyšlo, že celkový počet detekovaných obličejů je 897, počet správných detekcí obličejů (true positive) je 894, počet nesprávných pozitivních detekcí obličejů (false positive) je 3 a počet nesprávných negativních detekcí obličejů (false negative) je 65. Z daných výsledků pak lze spočítat, že precision vychází 0,9956, recall vychází 0,9322 a F-measure vychází 0,9628. Celkový čas běhu nad danou testovací sadou byl 26 s.

PO	PD	TP	FP	FN	Precision	Recall	F
1	1092	904	188	55	0,8278	0,9426	0,8815
2	966	883	83	76	0,9141	0,9208	0,9174
3	905	864	41	95	0,9547	0,9009	0,9270
4	878	855	23	104	0,9738	0,8916	0,9309
5	862	845	17	114	0,9803	0,8811	0,9281
6	849	836	13	123	0,9847	0,8717	0,9248
10	797	795	2	164	0,9975	0,8290	0,9055
20	674	674	0	285	1,0000	0,7028	0,8255

6.2: Detekce obličejů pomocí algoritmu Viola-Jones s krokem měřítka 1,1.

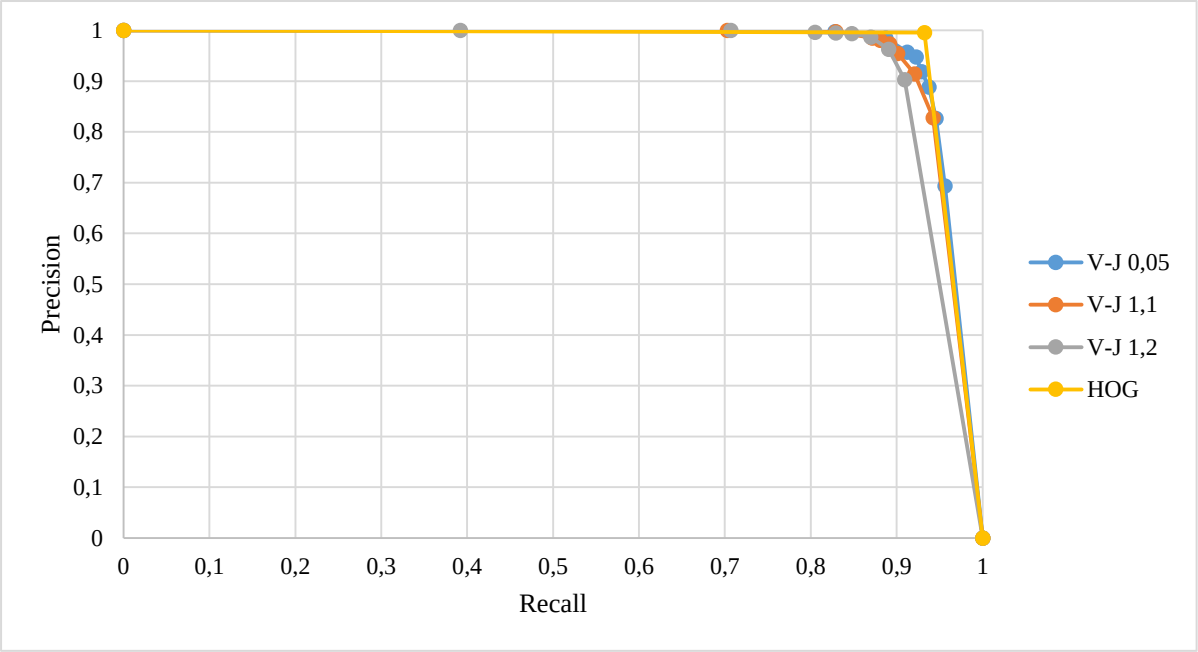
PO	PD	TP	FP	FN	Precision	Recall	F
1	966	872	94	87	0,9027	0,9093	0,9060
2	887	854	33	105	0,9628	0,8905	0,9252
3	845	834	11	125	0,9870	0,8697	0,9246
4	818	813	5	146	0,9939	0,8478	0,9150
5	799	795	4	164	0,9950	0,8290	0,9044
6	775	772	3	187	0,9961	0,8050	0,8904
10	678	678	0	281	1,0000	0,7070	0,8283
20	376	376	0	583	1,0000	0,3921	0,5633

6.3: Tabulka detekce obličejů pomocí algoritmu Viola-Jones s krokem měřítka 1,2.

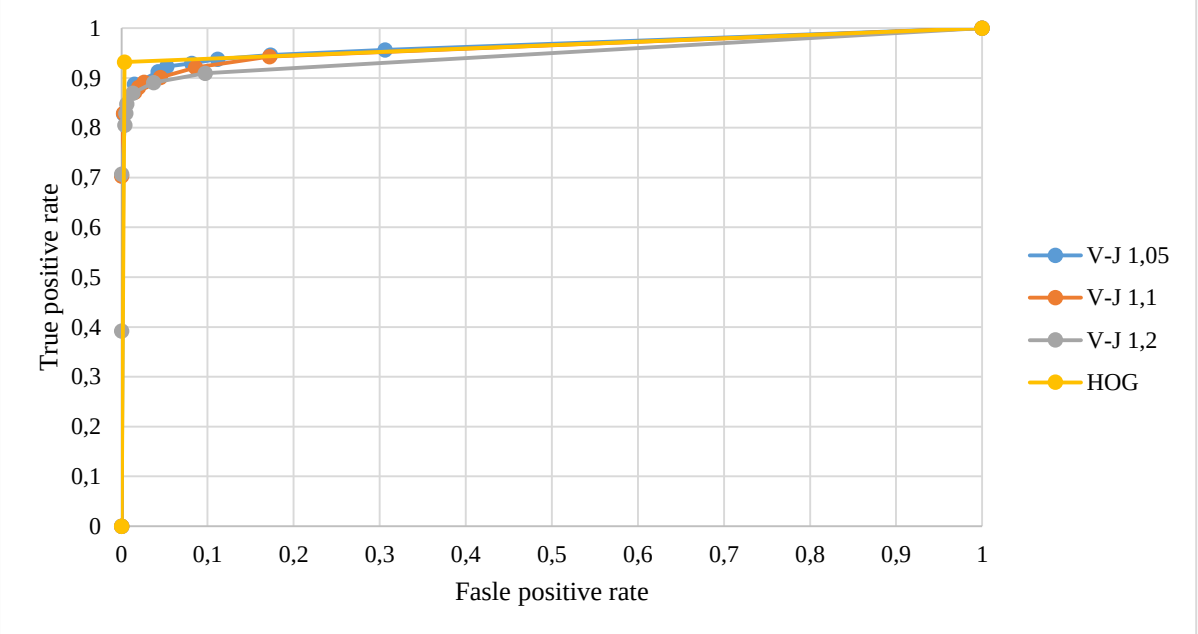
Ze všech naměřených výsledků jsem sestavil křivky precision-recall (obrázek 6.3) a ROC (obrázek 6.4). Pro algoritmus Viola-Jones je vytvořena křivka pro každou změnu měřítka a pak je ještě vytvořena jedna křivka pro algoritmus HOG. U precision-recall křivky víme, že každá křivka začíná v bodě [0,1] a končí v bodě [1,0]. Čím blíže je tato křivka k bodu [1,1], tím ji lze považovat za méně chybovou. ROC (receiver operating characteristic) křivka je definována jako poměr nesprávně určených pozitivních detekcí k poměru správně určených pozitivních detekcí (recall). U ROC křivky víme, že každá křivka začíná v bodě [0,0] a končí v bodě [1,1]. Čím víc se tento druh křivky blíží bodu [0,1], tím ji lze považovat za méně chybovou. [45]

Dále vytvořím tabulku, ve které porovnám implementaci algoritmů Viola-Jones a HOG. Z Viola-Jones vyberu pro každou změnu měřítka řádek s nejvyšší hodnotou F-measure a nakonec přidám hodnoty pro HOG. Do tabulky přidám i časy běhu nad testovacím datasetem (tabulka 6.4).

Z dané tabulky a daných precision-recall a ROC křivek lze vyčíst, že u implementace Viola-Jones platí, že při zmenšování změny měřítka dochází ke zkvalitňování detekce, ovšem dochází k razantnímu zvyšování času. Ale i když nastavíme změnu měřítka na 1,05, tak i přesto dosahuje implementace HOG mnohem lepších výsledků a dokáže pracovat několikanásobně rychleji než Viola-Jones při nastavení změny měřítka na 1,2. Proto se implementace HOG zdá být kvalitnější a vhodnější pro další použití.



Obrázek 6.3: Precision-recall křivka.



Obrázek 6.4: ROC křivka.

Algoritmus	ZM	PO	Precision	Recall	F-measure	Čas [s]
V-J	1,05	5	0,9475	0,9228	0,9350	147
V-J	1,1	4	0,9738	0,8916	0,9309	88
V-J	1,2	2	0,9628	0,8905	0,9252	67
HOG	-	-	0,9956	0,9322	0,9628	26

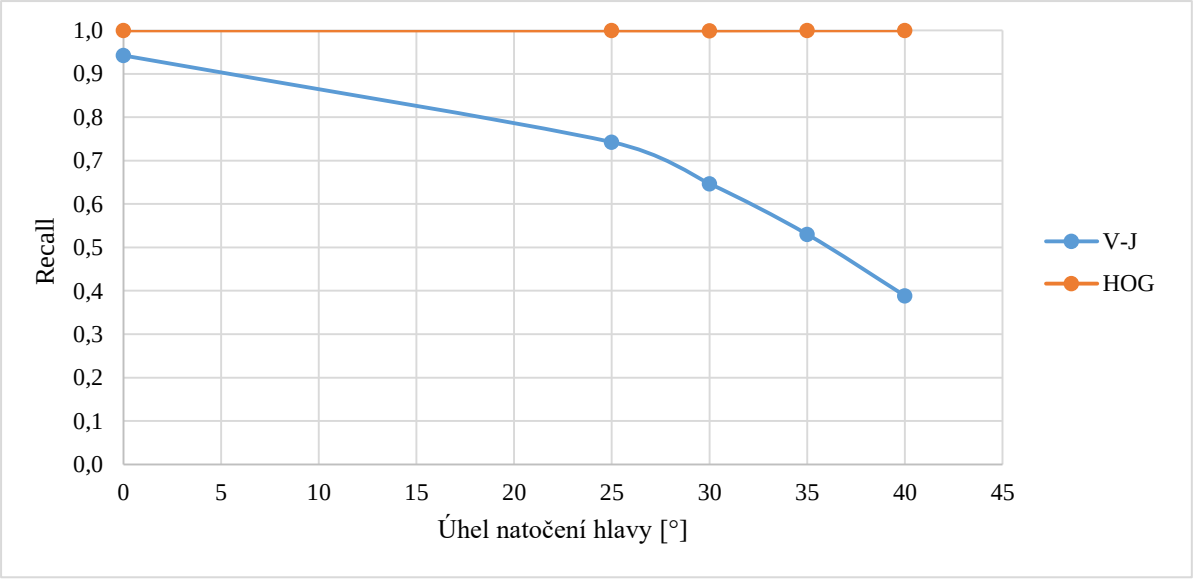
6.4: Tabulka porovnávající jednotlivé nastavení algoritmů.

6.2.1 Testování detekce natočených obličejů

Předchozí testování bylo prováděno na obecném datasetu, u kterého nebylo známo natočení hlav. Pro kvalitní určení natočení hlavy je potřeba, aby detektor obličejů byl schopen detekovat obličeje v co možná největším úhlu natočení. Proto provedu ještě testování detekce obličejů na datasetu od Gi4E. Tento dataset obsahuje hlavy natočené v úhlu až do 45°. Testování provedu tak, že budu zjišťovat, který detektor dokáže detekovat obličeje ve větším úhlu natočení hlavy. Postupně budu zvyšovat práh, který udává součet všech úhlů natočení hlavy, a budu zjišťovat, který detektor dokáže detekovat více obličejů natočených nad daný práh. Testovat budu detektor HOG a detektor Viola-Jones s nastavenou změnou měřítka na 1,1 a počtem oken nutných k detekci obličeje nastavených na 4 okna. V tabulce 6.5 jsou pak vidět výsledky daného testování. Na obrázku 6.5 je pak graf znázorňující vztah mezi úhlem natočení hlavy a hodnotou recall pro oba testované detektory.

	PO	**TP – HOG**	**Recall – HOG**	**TP – V-J**	**Recall – V-J**
0°	36000	35998	0,9999	33921	0,9423
25°	8015	8013	0,9998	5952	0,7426
30°	5442	5440	0,9996	3521	0,6470
35°	3756	3756	1,0000	1992	0,5304
40°	2449	2449	1,0000	952	0,3887

6.5: Tabulka zobrazuje, jak jsou detektory úspěšné při natočení hlavy nad určitou mez.



Obrázek 6.5: Vztah mezi úhlem natočení hlavy a recall pro Viola-Jones a HOG.

Z daných výsledků je patrné, že implementace detektoru HOG dosahuje mnohem kvalitnějších výsledků než implementace detektoru Viola-Jones. HOG dokáže detekovat na testovacím datasetu téměř všechny obličeje, a to nezávisle na úhlu natočení hlavy, kdežto úspěšnost detektoru Viola-Jones výrazně klesá se zvětšujícím se úhlem natočení hlavy. Od úhlu přibližně 36° již není schopen detekovat ani polovinu obličejů. Z tohoto důvodu jsem se rozhodl, že pro finální aplikaci budu již vždy používat detektor HOG naimplementovaný pomocí knihovny Dlib.

6.3 Testování určení natočení hlavy vůči kameře

Výsledné testování natočení hlavy vůči kameře provedu na referenčním datasetu od Gi4E, jenž zná přesné hodnoty úhlů natočení hlavy. Testování budu provádět pro obě implementace algoritmů.

O jak kvalitní detekci se jedná, budu vyhodnocovat na základě průměrné chyby a směrodatné odchylky pro všechny tři osy. Průměrná chyba pro každou osu se spočítá jako průměrná hodnota absolutní hodnoty z rozdílu výsledků testovaného algoritmu vůči referenčním výsledkům:

$$c = \frac{1}{N}\sum_{k=1}^{N}|m_k - r_k| \tag{6.4}$$

kde c je výsledná chyba pro danou osu, m_k je výsledná hodnota úhlu z mého programu pro k-tý vzorek, r_k je referenční hodnota úhlu pro k-tý vzorek a N je počet vzorků.

Směrodatná odchylka se pak spočítá:

$$\sigma = \sqrt{\frac{1}{N}\sum_{k=1}^{N}(m_k - \bar{c})^2} \tag{6.5}$$

kde σ je výsledná směrodatná odchylka pro danou osu, m_k je výsledná hodnota úhlu z mého programu pro k-tý vzorek, $\bar{c}$ je průměrná chyba pro danou osu a N je počet vzorků.

Problémem je, že obě implementace vyjadřují jinak úhel natočení hlavy. Algoritmus založený na antropometrických vlastnostech hlavy vyjadřuje úhel natočení hlavy vzhledem ke kameře. Kdežto algoritmus, který používá metodu PnP, vyjadřuje úhel natočení hlavy vzhledem k rovině kamery. Stejně jsou uložena referenční data v datasetu Gi4E. U datasetu Gi4E ale známe i přesnou pozici hlavy v prostoru, proto je pro potřeby testování nutné k úhlům z algoritmu založeného na antropometrických vlastnostech hlavy přičíst úhel, ve kterém se nachází hlava vůči kameře. Pro *yaw* se tento úhel spočítá:

$$\varepsilon = tan^{-1}\frac{x}{z} \tag{6.6}$$

a pro *pitch* se tento úhel spočítá:

$$\zeta = tan^{-1}\frac{x}{y} \tag{6.7}$$

kde ε a ζ jsou úhly, v níž se hlava odchyluje od středové osy kamery a x, y a z jsou pozice hlavy od ohniska kamery získané z databáze od Gi4E.

Samotné testování budu provádět pro oba implementované algoritmy a budu u nich měnit různé parametry za účelem co nejpřesnějšího odhadu úhlů. Dále budu měřit, jak dlouho průměrně trvá výpočet úhlů natočení pro jednu hlavu.

Konkrétně u algoritmu založeného na antropometrii hlavy budu zkoumat, zda u výpočtu *yaw* dosáhnu lepších výsledků, když zvolím bod ve středu obličeje jako špičku nosu (z obrázku 4.2 bod 31) nebo bod pod špičkou nosu (z obrázku 4.2 bod 12).

U algoritmu, který funguje na principu natáčení modelu hlavy, budu testovat různé metody pro řešení problému PnP, jenž jsou implementovány v metodě `solvePnP()`, jmenovitě Irerative, EPnP a DLS. Dále budu testovat jaký vliv má na kvalitu výpočtu počet zvolených bodů pro řešení PnP. Testování nejprve provedu pouze s pěti body, konkrétně se bude jednat o koutky úst (body 19 a 20), vnější koutky očí (body 1 a 4) a špičku nosu (bod 31). Poté přidám mezi testované body bradu (bod 25) a nakonec přidám bod nacházejí se na kořenu nosu (bod 11).

Výsledky testování algoritmu založeného na antropometrii hlavy jsou zobrazeny v tabulce 6.6. Výsledky testování algoritmu, který funguje na principu natáčení modelu hlavy, jsou zobrazeny v tabulce 6.7. V tabulce 6.8 jsou pak zobrazeny časy, jak dlouho průměrně trvá výpočet úhlů natočení pro jednu hlavu pro testované algoritmy.

Středový bod pro Yaw	**Roll – Pr. chyba [°]**	**Roll – Směr. odch. [°]**	**Yaw – Pr. chyba [°]**	**Yaw – Směr. odch. [°]**	**Pitch – Pr. chyba [°]**	**Pitch – Směr. odch. [°]**
Pod nosem	0,831	0,649	2,084	2,108	6,689	4,240
Špička nosu	0,831	0,649	2,116	1,934	6,689	4,240

6.6: Tabulka zobrazující výsledky testování pro algoritmus založený na antropometrii hlavy.

Metoda PnP	**Počet bodů**	**Roll – Pr. chyba [°]**	**Roll – Směr. odch. [°]**	**Yaw – Pr. chyba [°]**	**Yaw – Směr. odch. [°]**	**Pitch – Pr. chyba [°]**	**Pitch – Směr. odch. [°]**
Iterative	5	95,815	88,652	5,700	8,393	10,036	17,138
EPnP	5	0,856	0,783	3,347	3,465	7,441	4,748
DLS	5	0,856	0,783	3,347	3,465	7,441	4,748
Iterative	6	14,096	45,902	2,359	3,040	8,694	8,773
EPnP	6	1,099	0,947	3,132	3,316	7,020	4,807
DLS	6	1,099	0,947	3,132	3,316	7,020	4,807
Iretative	7	58,473	83,015	3,406	4,002	7,757	4,720
EPnP	7	1,214	1,055	3,290	3,299	7,049	4,848
DLS	7	1,214	1,055	3,290	3,299	7,049	4,848

6.7: Tabulka zobrazující výsledky testování pro algoritmus, který funguje na principu natáčení modelu hlavy.

Algoritmus	**Čas [μs]**
založený na antropometrii hlavy	0,771
PnP – Iteratice	274,247
PnP – EpnP	86,837
PnP – DLS	87,956

6.8: Tabulka zobrazující průměrnou dobu trvání výpočtu natočení pro jednu hlavu pro různé algoritmy.

Pro algoritmus založený na antropometrii hlavy lze z tabulky 6.6 vyčíst, že tento algoritmus má poměrně malou chybovost a průměrný čas výpočtu natočení hlavy je velmi nízký. Také jsem zjistil, že jestli zvolíme jako středový bod pro výpočet *yaw* bod nacházející se pod špičkou nosu nebo bod na špičce nosu, tak pokud správně dopočítáme konstantu k, tak na výsledný úhel *yaw* to má minimální vliv.

V tabulce 6.7 lze vidět, že jednoznačně nejhorší metoda pro výpočet PnP je metoda Iterative. Nejhorších výsledků dosahuje zejména pro *roll*, to je způsobeno hlavně proto, že u *roll* dochází často k výpočtu úhlu s chybou 180°. U metod EPnP a DLS zase můžeme vidět, že jejich chyby jsou naprosto totožné a čas výpočtu je také téměř stejný. Z toho lze usuzovat, že se jedna o chybu OpenCV a ve skutečnosti se jedná o jednu a tu samou metodu. Z těchto metod lze vyčíst, že nejlepších výsledků jsou schopny dosáhnout, pokud se použije 6 bodů, a to levý a pravý koutek úst, levý a pravý koutek oka, špička nosu a brada.

Pokud porovnáme průměrnou chybu všech metod, tak nejnižší chybovost vychází pro algoritmus založený na antropometrii hlavy, těsně za ním je metoda EPnP (respektive DLS) a nejhůře je na tom metoda Iterative. Pokud porovnáme průměrný čas výpočtu natočení pro jednu hlavu, tak jednoznačně nejrychlejší je algoritmus založený na antropometrii hlavy. To je způsobeno zejména tím, že se jedná o jednoduchý algoritmus, který není iterativní.

Největší průměrná chyba a největší směrodatná odchylka vychází u většiny algoritmu pro *pitch*. Pokud si však spočteme směrodatnou odchylku pro *pitch* pro každou osobu zvlášť, tak nám vyjde mnohem nižší. To je způsobeno zejména tím, že každý člověk má jiný tvar hlavy. U *pitch* je to způsobeno převážně tím, že výpočet *pitch* je závislý zejména na nose. Pokud má tedy někdo nos výše než je průměr, tak pro tohoto člověka bude hodnota *pitch* vždy o něco větší než u ostatních a naopak. Z tohoto důvodu je velmi obtížné dosáhnout toho, aby pro každého člověka byla při stejném natočení hlavy v ose *pitch* spočtena stejná hodnota pro *pitch*.

6.3.1 Porovnání výsledků na datasetu Gi4E

Pro zjištění úspěšnosti výpočtů je vždy vhodné porovnat vlastní výsledky s výsledky ostatních. K tomu použiji tabulku, která se nachází na stránkách datasetu Gi4E. Tabulka obsahuje 4 různé implementace a pro každou z nich je spočítána průměrná chyba pro *roll*, *yaw* a *pitch*. Tato tabulka vypadá takto:

Použitá metoda	**Roll – Pr. chyba [°]**	**Yaw – Pr. chyba [°]**	**Pitch – Pr. chyba [°]**
Posit + ASM 2D + BFM Model	1,12	2,97	4,04
Posit + AAM 2D + BFM Model	1,74	2,30	6,01
Posit + ASM 2D + Cylindrical Model	1,14	3,56	5,52
Posit + AAM 2D + Cylindrical Model	1,74	3,68	8,83

6.9: Tabulka zobrazující průměrné chyby 4 implementací na datasetu Gi4E [43].

Pokud porovnám výsledky z mého programu s výsledky v tabulce 6.9, tak mnou implementovaný algoritmus založený na antropometrii hlavy dosahuje nejlepších výsledků pro *roll* a *yaw* a mírně podprůměrných výsledků dosahuje pro *pitch*.

7 Závěr

Cílem této práce bylo vytvořit aplikaci, která by byla schopna určit úhel natočení hlavy v záznamu pořízeném bezpečnostní kamerou. To provádím tak, že nejprve detekuji obličeje ve snímku, v každém obličeji pak lokalizuji charakteristické body a na základě těchto bodů určím výsledný úhel natočení hlavy.

Pro detekci obličejů jsem použil algoritmus Viola-Jones a algoritmus založený na histogramech orientovaných gradientů (HOG). Testováním jsem zjistil, že algoritmus HOG dokáže správně detekovat větší počet obličejů oproti algoritmu Viola-Jones, zejména pak obličejů natočených pod větším úhlem, navíc je HOG mnohem rychlejší.

K lokalizaci charakteristických bodů jsem použil algoritmus „One Millisecond Face Alignment with an Ensemble of Regression Trees“, který je oproti jiným algoritmům fungujících na principu aktivní šablony rychlejší. Při úspěšné detekci obličeje dokáže tento algoritmus lokalizovat charakteristické body téměř bezchybně.

K samotnému určení natočení hlavy jsem naimplementoval dva algoritmy. První algoritmus počítá úhel natočení hlavy na základě antropometrických vlastností hlavy. Druhý algoritmus se snaží natáčet model hlavy tak, aby co nejvíce odpovídal hlavě na snímku. Tento algoritmus používá pro určení správné pozice hlavy algoritmus Perspective-n-Point (PnP). Testováním na referenčním datasetu jsem prověřil oba implementované algoritmy a zjistil jsem, že oba dosahují kvalitních výsledků. O něco lepších výsledků dosáhl algoritmus fungující na základě antropometrických vlastností hlavy. Průměrná chyba tohoto algoritmu oproti referenční výsledkům byla pro *roll* 0,831°, pro *yaw* (azimut) 2,084° a pro *pitch* (elevace) 6,689°, což byly pro hodnoty *roll* a *yaw* nejlepší výsledky ze všech algoritmů testovaných na stejném datasetu. Další výhodou tohoto algoritmu je, že pracuje velmi rychle. Určit úhel natočení jedné hlavy mu v průměru zabere 771 ns. Takto rychlého času bylo dosaženo tím, že se nejedná o iterační algoritmus. Proto takovýto algoritmus může být vhodné použít pro real-time aplikace nebo pro zařízení s malým výpočetním výkonem.

Tato práce může mít do budoucna přínos při zdokonalování metod pro rozpoznávání a identifikaci osob. Získané výsledky mohou mít pozitivní vliv na zlepšení využitelnosti dat z bezpečnostních kamerových systémů, například pro strojové rozpoznávání osob.

Do budoucna by se dala aplikace rozšířit o další algoritmy k určení natočení hlavy vůči kameře. Dále by se mohly natrénovat klasifikátory, které by byly schopny detekovat hlavu i z profilu. V neposlední řadě by se dala aplikace rozšířit o grafické uživatelské rozhraní.

Literatura

[1] DORNBERGER, Walter. *V-2.* : Ballantine Books, 1954. ASIN: B000P6L1ES.

[2] ROBB, Gray C. *Police Use of CCTV Surveillance: Constitutional Implications and Proposed Regulations.* : University of Michigan Journal of Law Reform, 1979.

[3] MANNETI, Mazurio. Analog or Digital CCTV Systems: Which one is right for your property? *http://hotelexecutive.com/.* [Online] http://hotelexecutive.com/business_review/2851/analog-or-digital-cctv-systems-which-one-is-right-for-your-property.

[4] JANESICK, James R. *Scientific Charged-Coupled Devices.* Bellingham : SPIE Press, 2001. ISBN 0-8194-3698-4.

[5] The Nobel Prize in Physics 2009. *nobelprize.org.* [Online] http://www.nobelprize.org/nobel_prizes/physics/laureates/2009/index.html.

[6] ŠERÝCH, Jakub. *commons.wikimedia.org.* [Online] 13. 4. 2006. [Citace: 4. 1. 2017.] commons.wikimedia.org/wiki/File:Plošné_CCD.png.

[7] *commons.wikimedia.org.* [Online] 1. 2. 2008. [Citace: 4. 1. 2017.] commons.wikimedia.org/wiki/File:Bayer_matrix.svg.

[8] ZEMČÍK, Pavel et al. *Zpracování obrazu.* Brno : Fakulta informačních technologií Vysokého učení technického v Brně, 2011.

[9] NIXON, Mark S., Alberto S. AGUARDO. *Feature Extraction and Image Processing.* Oxford : Elsevier, 2008. ISBN 978-0-12372-538-7..

[10] CMOS. *Stemmer imaging.* [Online] [Citace: 5. 1. 2017.] http://www.stemmer-imaging.fr/fr/donnees/cmos/.

[11] COOTES, F. T. a J. C. TAYLOR. *Statistical Models of Appearance.* Manchester : University of Manchester, 2004.

[12] DRAHANSKÝ, Martin, Filip ORSÁG et al. *Biometrie.* Brno : Computer Press a. s., 2011. stránky 165-166. ISBN 978-80-254-8979-6.

[13] KAZEMI, Vahid a Josephine SULLIVAN. *One Millisecond Face Alignment with an Ensemble of Regression Trees.* Stockholm : KTH, Royal Institute of Technology, 2014. CVPR.

[14] VIOLA, Paul a Michael JONES. *Robust Real-Time Face Detection.* [Online] Netherlands : Kluwer Academic Publishers, 2004. International Journal of Computer Vision 57.

[15] PŘINOSIL, Jiří a Martin KROLIKOWSKI. *Využití detektoru Viola-Jones pro lokalizaci obličeje a očí v barevných obrazech.* [Online] Brno, Česká republika : Vysoké učení technické v Brně, 2008.

[16] WILSON, Phillip Ian a John FERNANDES. *Facial feature detection using Haar classifiers.* [Online] 2006.

[17] LIENHART, Rainer a Jochen MAYTD. *An Extended Set of Haar-like Features for Rapid Object Detection.* [Online] Santa Clara, CA 95052, USA : Intel Labs, Intel Corporation.

[18] MATAS, Jiří a Jan ŠOCHMAN. *AdaBoost.* [Online] Prague : Czech Technical University.

[19] DALAL, Navneet a Bill TRIGGS. *Histograms of Oriented Gradients for Human Detection.* Montbonnot : INRIA Rhône-Alpes, 2005.

[20] ŽIŽKA, J. *Support vector machines (SVM).* Brno : Masarykova univerzita, 2004.

[21] NG, Andrew. *The Simplified SMO Algorithm.* 2009.

[22] POŠÍK, Petr. *Optimální rozdělující nadplocha. Support vector machine. Adaboost.* Prague : Czech Technical University in Prague, 2012.

[23] COOTES, T. F., C. J. TAYLOR, D. H. COOPER a J. GRAHAM. *Actice Shape Models - Their Training and Aplication.* Manchester : University of Manchester, 1994.

[24] KOEMENY, Jean a Sébastien MARCEL. *Active Shape Models Using Local Binary Patterns.* 2006.

[25] *Microsoft.* [Online] https://www.microsoft.com/cognitive-services/en-us/face-api/documentation/glossary.

[26] MALINA, Jaroslav et al. Andropologický slovník. [Online] Přírodovědecká fakulta Masarykovy univerzity. http://is.muni.cz/do/1431/UAntrBiol/el/antropos/slovnik.html.

[27] HRDLIČKA, Aleš. *Anthropometry.* Philadelphia : The Wistar institute of anatomy and biology, 1920.

[28] ULIJASZEK, S. J. a C. N. G. MASCILE-TAYLOR. *Anthropometry: the individual and the polulation.* Cambridge : Cambridge university press, 1994. ISBN-13 978-0-521-41798-3.

[29] 29. POSTON, Alan. *Human engineering design data digest.* Washington, DC : Departmen of defence, 2000.

[30] FETTER, Vojtěch et al. *Atropologie.* Praha : Academia, 1967.

[31] HRBÁČOVÁ, Alena. *Rysy ve tváři partnerů a jejich vliv na vzájemnou přitažlivost.* Olomouc : Univerzita Palackého v Olomouci, Filozofická fakulta, 2014.

[32] OHAYON, Shay a Ehud RIVLIN. *Robust 3D Head Tracking Using Camera Pose Estimation.* Haifa : Computer Science Department, Israel Institute of Technology, 2006.

[33] MALLICK, Satya. Head Pose Estimation using OpenCV and Dlib. *Learn OpenCV.* [Online] 26. 09. 2016. [Citace: 15. 05. 2017.] http://www.learnopencv.com/head-pose-estimation-using-opencv-and-dlib/.

[34] Camera Calibration and 3D Reconstruction. *OpenCV.* [Online] [Citace: 2017. 5. 16.] http://docs.opencv.org/3.0-beta/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html#solvepnp.

[35] LEPETIT, Vincent, Francesc MORENO-NOGUER, Pascal FUA. *EPnP: Efficient Perspective-n-Point Camera Pose Estimation.* Lausanne : École Polytechnique Fédérale de, 2008.

[36] HESCH, Joel A. a Stregios I. ROUMELIOTIS. *A Direct Least-Squares (DLS) Method for PnP.* Minneapolis : University of Minnesota, 2011.

[37] PENATE-SANCHEZ, Adrian, Juan ANDRADE-CETTO a Francesc MORENO-NOGUER. *Exhaustive Linearization for Robust Camera.* Barcelona : Institut de Robòtica i Informàtica Industrial.

[38] *OpenCV.* [Online] [Citace: 15. 5. 2017.] http://opencv.org/.

[39] KING, Davis E. Dlib-ml: A Machine Learning Toolkit. *Journal of Machine Learning Research.* 10, 2009, stránky 1755-1758.

[40] GOLDMANN, Tomáš. *HeadViewer - simulační nástroj pro určení pozice hlavy z pohledu CCTV kamery.* [Online] http://www.fit.vutbr.cz/~igoldmann/app/headviewer/.

[41] MALLICK, Satya. Approximate Focal Length for Webcams and Cell Phone Cameras. *Learn OpenCV.* [Online] 17. 6. 2016. [Citace: 15. 5. 2017.] https://www.learnopencv.com/approximate-focal-length-for-webcams-and-cell-phone-cameras/.

[42] HO, Nghia. Decomposing and composing A 3×3 Rotarion matrix. *Nghia Ho.* [Online] [Citace: 2017. 5. 15.] http://nghiaho.com/?page_id=846.

[43] ARIZ, Mikel, José J. BENGOECHEA, Arantxa VIllanueva a Rafael CABEZA. A novel 2D/3D database with automatic face annotation for head tracking and pose estimation. *Computer Vision and Image Understanding.* 2016, Sv. 148, stránky 201-210.

[44] JAIN, Vidit a Eric LEARNED-MILLER. *FDDB: A Benchmark for Face Detection in Unconstrained Settings.* Amherst : University of Massachusetts, 2010.

[45] DAVIS, Jesse a Mark GOARDITCH. *The Relationship Between Precision-Recall and ROC Curves.* Madison : Department of Computer Sciences and Department of Biostatistics and Medical Informatics, University of Wisconsin-Madison.

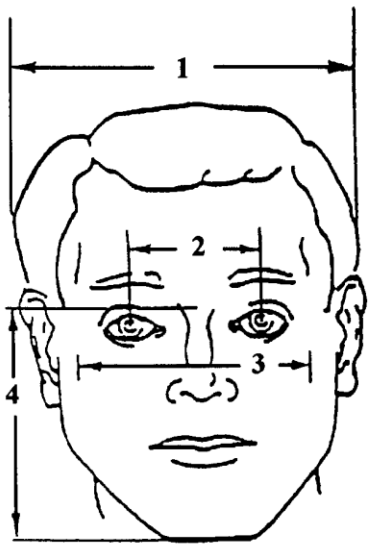
[46] POWERS, David M. W. *Evaluation: from precision, recall and F-measure to roc, informendness, markedness & correladion.* : AILab, School of Computer Science, Engineering and Mathematics, Flinders University, 2011.

A Obsah CD

- **src** – zdrojové kódy programu
- **bin** – spustitelné soubory
- **doc** – zdrojové soubory textu práce
- **dataset** – referenční dataset, na kterém bylo prováděno testování
- **DP_Ondrej_Blucha.pdf** – text diplomové práce
- **readme.pdf** – pokyny pro práci s programem

B Tabulky s antropometrií hlavy

Převzato z [29].



1 **Head breadth.** The maximum breadth of the head, usually above and behind the ears.

	Sample		Percentiles 1st	5th	50th	95th	99th
A	Men	cm	13.9	14.3	15.2	16.11	6.5
		(in)	(5.1)	(5.6)	(6.0)	(6.3)	(6.5)
B	Women	cm	13.3	13.7	14.4	15.3	15.7
		(in)	(5.2)	(5.4)	(5.7)	(6.0)	(6.1)

2 **Interpupillary breadth.** The distance between the centers of the pupils of the eyes (the eyes are looking straight ahead).

	Sample		Percentiles 1st	5th	50th	95th	99th
A	Men	cm	5.7	5.9	6.5	7.1	7.4
		(in)	(2.2)	(2.3)	(2.7)	(2.8)	(2.9)
B	Women	cm	5.5	5.7	6.0	6.9	7.0
		(in)	(2.8)	(2.2)	(2.4)	(2.7)	(2.8)

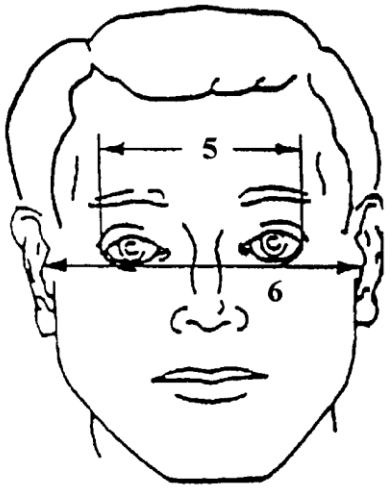
3 **Face breadth (bizygomatic).** The breadth of the face, measured across the most lateral projections of the cheek bones (zygomatic arches).

	Sample		Percentiles 1st	5th	50th	95th	99th
A	Men	cm	12.8	13.2	14.0	15.0	15.4
		(in)	(5.0)	(5.2)	(5.5)	(5.9)	(6.1)
B	Women	cm	12.1	12.3	12.8	14.0	15.4
		(in)	(4.8)	(4.8)	(5.1)	(5.5)	(5.7)

4 **Face length (menton-sellion).** The vertical distance from the tip of the chin (menton) to the deepest point of the nasal root depression between the eyes (sellion).

	Sample		Percentiles 1st	5th	50th	95th	99th
A	Men	cm	10.8	11.2	12.2	13.3	13.7
		(in)	(4.3)	(4.4)	(4.8)	(5.2)	(5.4)
B	Women	cm	10.1	10.4	11.3	12.4	12.9
		(in)	(3.4)	(4.1)	(4.5)	(4.9)	(5.1)

Exhibit 14.3.2.1 (continued) Static human physical characteristics (head)

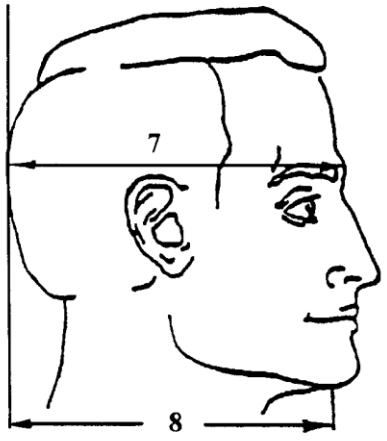


5 **Biocular breadth.** The distance from the outer corners of the eyes (right and left ectocanthi).

	Sample		Percentiles 1st	5th	50th	95th	99th
A	Men	cm	11.0	11.3	12.2	13.1	13.6
		(in)	(4.3)	(4.5)	(4.8)	(5.2)	(5.4)
B	Women	cm	10.8	11.1	11.6	12.9	13.3
		(in)	(4.3)	(4.4)	(4.3)	(5.1)	(5.3)

6 **Bitragion breadth.** The breadth of the head from the right tragion to the left. (Tragion is the cartilaginous notch at the front of the ear).

	Sample		Percentiles 1st	5th	50th	95th	99th
A	Men	cm	13.1	13.5	14.5	15.5	15.9
		(in)	(5.2)	(5.3)	(5.7)	(6.1)	(6.3)
B	Women	cm	12.5	12.8	13.3	14.3	15.0
		(in)	(4.3)	(5.4)	(5.4)	(5.7)	(5.9)



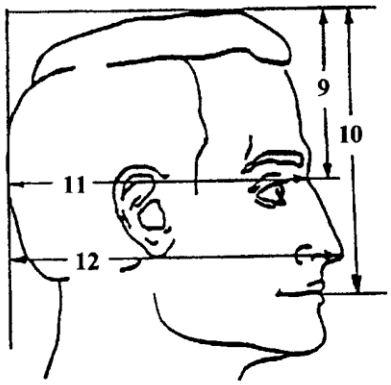
7 **Glabella to back of head.** The horizontal distance from the most anterior point of the forehead between the brow-ridges (glabella) to the back of the head, measured with a headboard.

	Sample		Percentiles 1st	5th	50th	95th	99th
A	Men	cm	18.3	18.8	20.0	21.1	21.7
		(in)	(7.2)	(7.4)	(7.9)	(8.3)	(8.5)
B	Women	cm	17.5	18.0	19.1	20.2	20.7
		(in)	(6.9)	(7.1)	(7.5)	(8.0)	(8.1)

8 **Menton to back of head.** The horizontal distance from the tip of the chin (menton) to the back of the head, measured with a headboard.

	Sample		Percentiles 1st	5th	50th	95th	99th
A	Men	cm	15.7	16.5	18.2	20.0	20.7
		(in)	(6.2)	(6.5)	(7.2)	(7.9)	(8.2)
B	Women	cm	15.2	15.8	17.3	18.9	19.6
		(in)	(6.0)	(6.2)	(6.8)	(7.4)	(7.7)

Exhibit 14.3.2.1 (continued) Static human physical characteristics (head)



9 Sellion to top of head. The vertical distance from the nasal root depression between the eyes (sellion), to the level of the top of the head, measured with a headboard.

	Sample		Percentiles 1st	5th	50th	95th	99th
A	Men	cm	9.7	10.1	11.2	12.4	12.9
		(in)	(3.8)	(4.0)	(4.4)	(4.9)	(5.1)
B	Women	cm	9.0	9.5	10.5	11.7	12.2
		(in)	(3.5)	(9.5)	(4.1)	(4.6)	(4.8)

10 Stomion to top of head. The vertical distance from the midpoint of the lips (stomion) to the level of the top of the head, measured with a headboard.

	Sample		Percentiles 1st	5th	50th	95th	99th
A	Men	cm	16.9	17.4	18.6	19.9	20.6
		(in)	(6.7)	(6.6)	(7.3)	(7.8)	(8.1)
B	Women	cm	15.7	16.3	17.5	18.8	19.4
		(in)	(6.1)	(6.4)	(6.9)	(7.4)	(7.6)

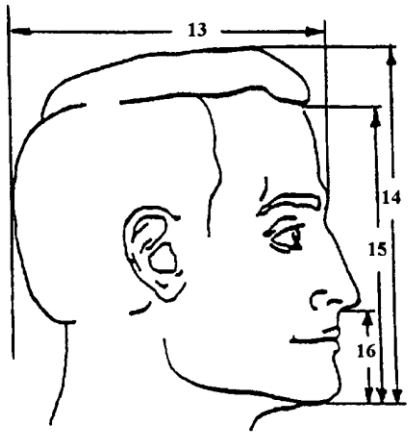
11 Sellion to back of head. The horizontal distance from the nasal root depression between the eyes (sellion), to the back of the head, measured with a headboard.

	Sample		Percentiles 1st	5th	50th	95th	99th
A	Men	cm	18.0	18.5	19.7	20.9	21.4
		(in)	(7.1)	(7.3)	(7.8)	(8.2)	(8.4)
B	Women	cm	17.4	17.8	18.9	20.0	20.5
		(in)	(6.6)	(7.1)	(7.4)	(7.9)	(8.4)

12 Pronasale to back of head. The horizontal distance from the tip of the nose (pronasale) to the back of the head, measured with a headboard.

	Sample		Percentiles 1st	5th	50th	95th	99th
A	Men	cm	20.0	20.5	22.0	23.2	23.9
		(in)	(7.9)	(8.1)	(8.7)	(9.1)	(9.4)
B	Women	cm	19.2	19.7	21.0	22.2	22.8
		(in)	(7.6)	(7.8)	(8.3)	(8.7)	(9.0)

Exhibit 14.3.2.1 (continued) Static human physical characteristics (head)



13 Head length. The maximum length of the head; measured from the most anterior point of the forehead between the brow-ridges (glabella) to the back of the head (occiput).

	Sample		Percentiles 1st	5th	50th	95th	99th
A	Men	cm	18.0	18.5	19.7	20.9	21.3
		(in)	(7.1)	(7.3)	(7.8)	(8.2)	(8.4)
B	Women	cm	17.2	17.6	18.7	19.8	20.2
		(in)	(6.8)	(7.0)	(7.4)	(7.8)	(8.0)

14 Menton to top of head. The vertical distance from the tip of the chin (menton) to the level of the top of the head, measured with a headboard.

	Sample		Percentiles 1st	5th	50th	95th	99th
A	Men	cm	21.2	21.8	23.2	24.7	25.5
		(in)	(8.4)	(8.6)	(8.6)	(9.1)	(9.4)
B	Women	cm	19.8	20.4	21.8	23.2	23.8
		(in)	(7.8)	(8.3)	(8.6)	(9.1)	(9.4)

15 Menton-crinion length. The vertical distance from the bottom of the chin (menton) to the midpoint of the hairline (crinion).

	Sample		Percentiles 1st	5th	50th	95th	99th
A	Men	cm	16.6	17.4	19.1	20.9	21.6
		(in)	(6.5)	(6.9)	(7.5)	(8.2)	(8.5)
B	Women	cm	15.5	16.1	17.7	19.2	19.9
		(in)	(6.1)	(6.3)	(7.0)	(7.6)	(7.8)

16 Menton-subnasale length. The distance from the bottom of the chin (menton) to the base of the nasal septum (subnasale).

	Sample		Percentiles 1st	5th	50th	95th	99th
A	Men	cm	6.1	6.5	7.3	8.3	8.7
		(in)	(2.4)	(2.7	(2.9)	(3.3)	(3.3)
B	Women	cm	5.7	6.0	6.5	7.8	8.3
		(in)	(2.2)	(2.4)	(2.7)	(3.1)	(3.8)