Prohlášení

Předkládám tímto k posouzení a obhajobě diplomovou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.
Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne 18.5.2013

.....
vlastnoruční podpis

Anotace

Tato práce se zaměřuje na metody získávání, zpracování a následnou klasifikaci trojrozměrných dat tvaru lidského obličeje. Je zde popsán také vhodný algoritmus pro přizpůsobení 3D dat pořízených zařízením Kinect Xbox360 modelu vytvořeného univerzitou v Baselu, který je modifikován pomocí implementace hloubkového ořezu a volnosti v pohybu. Pro přizpůsobení je využit algoritmus simulovaného žíhání, pomocí kterého jsou zpracována reálná data celkem devíti různých osob. Následné klasifikace těchto dat probíhá pomocí tří typů klasifikátorů, z čehož nejlepší výsledky poskytuje SVM klasifikátor.

Klíčová slova

3D data, 3D rekonstrukce, tvarovatelný model lidského obličeje, přizpůsobování modelu, klasifikace, rozpoznávání obrazů

Summary

This thesis is focused on methods of capturing, editing and subsequent classification of three-dimensional shape data of a human face. There is also described an algorithm for fitting 3D data captured by device Xbox360 Kinect to model created by University in Basel, which is modified by implementing crop in depth and possibility of movement. Algorithm of simulated is used fitting 3D data. By using this algorithm we compile real data of nine different people. Subsequent classification of these data is made by using three types of classifiers, from which the best results are obtained by SVM classifier.

Keywords

3D data, 3D reconstruction, morphable face model, model fitting, classification, pattern recognition

Poděkování

Především bych rád poděkoval Ing. Zdeňku Krňoulovi Ph.D a to nejen za jeho profesionální vedení práce a cenné odborné rady, ale obzvlášť za jeho obrovskou trpělivost, která musela procházet mnoha tvrdými zkouškami během naší spolupráce. Dále bych chtěl poděkovat rodině, mojí přítelkyni a mým přátelům za jejich podporu během studia i mimo něj. Na závěr posílám své díky i několika ochotným kolegům pana Krňoula, sdílejícím s ním tutéž kancelář, za to, že rádi a ochotně poradili navzdory tomu, že to zdaleka nebyla jejich povinnost.

Obsah

Kapitola 1

Úvod

Tato práce plynule navazuje na moji bakalářskou práci, přičemž se ji snaží rozšířit a odstranit její problémy. Cílem je tedy hlouběji prostudovat stávající metody a zařízení, sloužící k získávání a následnému zpracování vizuálních dat lidské tváře. Dále se práce zabývá možnostmi klasifikace těchto dat.

Mým úkolem bylo přizpůsobením vhodného modelu data lidského obličeje zpracovat s co nejmenší chybou a zároveň s rozumnými časovými nároky. Na rozdíl od BP byla algoritmu dodána volnost pohybu a natočení modelu. Dále byla implementována možnost změny měřítka a vylepšen ořez modelem vygenerovaných instancí. Dalším zlepšením je možnost označit na snímku tzv. referenční body, které přispívají ke zlepšení finálního řešení.

Snímky reálných osob byly pořízeny zařízením Kinect Xbox360, které bylo shledáno jako vhodné a zároveň dostupné. Posléze jsem vybral vhodný klasifikátor a část získaných dat využil k jeho natrénování. Zbytek dat dále posloužil k experimentální klasifikaci. Pro klasifikaci jsem nejprve vybral modifikovaný k-means klasifikátor, který sice nevykazoval dramatické nedostatky,nicméně jsem zde stále viděl prostor pro zlepšení. To přinesl až SVM klasifikátor, který poskytuje velice uspokojivé výsledky.

Animační model byl ponechán původní model z BP, ve které byl zvolen 3D face model vyvinutý na univerzitě v Baselu, a to nejen kvůli jeho dostupnosti, ale také pro to, že jeho uživatelské prostředí je velmi příjemné a lehce modifikovatelné, čehož bylo dále hojně využito. Tento model také nabízí mnoho možností zobrazení lidského obličeje, a to včetně jeho textury, která však nebyla vzhledem k cíli práce využita.

Jako vývojové prostředí byl vybrán MATLAB, a to hned z několika důvodů. Za prvé, kvůli jeho efektivitě, co se týče zpracování matic, za druhé, vzhledem k faktu, že vybraný Baselský model také využívá toto vývojové prostředí a za třetí, kvůli přehlednosti a uživatelsky přátelskému rozhraní, které MATLAB nabízí.

Kapitola 2

Kolekce trojrozměrných dat

Trojrozměrná data mají v dnešní době široké využití, v posledních letech především v zábavním průmyslu (filmy, počítačové hry), průmyslovém designu, v lékařství (model protézy) a nebo například při návrzích prototypů. S tím vším je spojen i fakt existence mnoha rozličných metod na jejich pořizování a zpracování. Pokusil jsem se tyto metody prostudovat, přičemž jsem se zaměřil především na ty využitelné pro snímání lidské tváře [10].

2.1 Metody získávání obrazových dat lidského obličeje

Oproti minulosti zaznamenaly nové technologie posledních let znatelné zlepšení snímků, a to především v oblasti rozlišení a přesnosti. Díky tomuto faktu lze některé z těchto metod využít na získávání trojrozměrných dat lidského obličeje, což dříve nebylo dost dobře možné, vzhledem k absenci dostatečné přesnosti pro zpracování či rozpoznání těchto obrazových materiálů. Nyní uvádím několik metod používaných k pořizovaní vizuálních dat lidského obličeje.

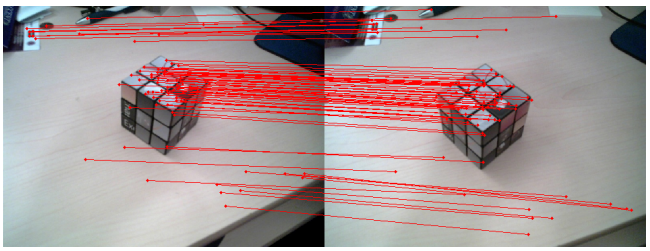
2.1.1 Stereofotogrammetrie

K pořízení trojrozměrných dat mohou být použity tzv. stereo páry obrazů [13]. Stereofotogrammetrie, založená na navzájem se překrývajících obrázcích, je základní metoda pro získávání 3D dat s použitím 2D obrazů, vychází tedy z faktu, že "normální" fotografie zachycuje věrný 2D obraz objektu, jako je tomu i u našeho oka. To, že člověk vidí ve 3D, je způsobeno tím, že každé z našich očí pořizuje tento obraz z jiného místa. Čím je objekt k našim očím blíže, tím více je navzájem v obrazech z každého oka posunut. A přesně na tomto jevu pracuje stereofotogrammetrie.

Metoda husté 3D rekonstrukce

Nejpoužívanější metodou stereofotogrammetrie je metoda husté 3D rekonstrukce. Při jejím použití většinou pracujeme se dvěma, ale můžeme i s více snímky daného subjektu. Prvním krokem je nalezení jednotlivých korespondencí mezi obrazy, viz obr. 2.1 tedy body, které jsou ve skutečnosti na reálném objektu identické. Pro hledání těchto korespondencí existuje několik metod, z nichž jmenujme například metodu SIFT [13].

Problém v této části nastává, pokud se několik významných bodů spáruje chybně, potom může dojít k tomu, že výsledek je špatný. Dalším krokem je pořízení disparitní mapy měřeného subjektu. I zde dochází k problémům a to v případě, že na pozorovaném objektu je málo význačných bodů. Proto je leckdy použita textura, která je na focený předmět promítnuta, čímž dojde k vysokému nárůstu korespondujících bodů. Posledním krokem je použití filtru, který nám zajistí doplnění chybějících dat a přemazání chybných bodů.



Obrázek 2.1: Ukázka stereofotografie s významnými body a jejich korespondencemi

2.1.2 3D sken

Dalším velmi rozšířeným přístupem je získání trojrozměrných dat pomocí 3D skenerů. Ke zkonstruování těchto skenerů existuje mnoho rozličných technologií, přičemž každá má svoje výhody, nevýhody a cenu. I přesto, že se tato technologie v posledních letech vyvíjí mílovými skoky, stále existují u většiny těchto přístrojů jistá omezení a problémy, a to například při skenování průhledných, třpytivých nebo čirých objektů. Úkolem 3D skenerů je vlastně získání sítě bodů na povrchu skenovaného objektu (jedná se soubor třídimenzionálních dat). Tyto body mohou být později použity pro extrapolaci tvaru skenovaného předmětu (tomuto procesu se říká rekonstrukce). Některé skenery snímají i barvu v každém bodě výše zmíněné sítě. V takovém případě může být později zrekonstruována i textura. 3D skenery se ve skutečnosti velmi podobají kamerám, s tím rozdílem, že účelem kamery je zachytit barvu objektu, zatímco skener snímá vzdálenost každého bodu předmětu od něj (neboli obrázek ze skeneru popisuje vzdálenost každého bodu povrchu snímaného objektu). Tento údaj a fakt, že je každý bod navíc skenerem zasazen do sférického souřadného systému, plně popisuje polohu každého bodu sítě v prostoru.

Pokud chceme kompletní 3D obraz subjektu, většinou nestačí pouze jediný sken, ale je jich potřeba více, někdy i několik desítek. Všechna tato vícenásobná skenování musí mít nějaký společný referenční systém (často nazývaný registrace), díky kterému mohou být později jednotlivé skeny zkombinovány. Jak již bylo řečeno, existuje celá řada technologií 3D skenování. Tyto technologie se obvykle dělí na kontaktní a bezkontaktní 3D skenery. Vzhledem k obsahu této práce a využitelnosti se dále zabývám pouze bezkontaktními 3D skenery.

Laserový skener (Time-of-flight scanner)

Tento skener využívá laser ke zkoumání daného subjektu, přičemž jeho hlavní součástkou je laserový dálkoměr. Ten funguje na principu měření doby, za kterou světlo urazí

vzdálenost k povrchu objektu a zpátky. Jelikož rychlost světla je známá, vypočte se potom vzdálenost daného bodu na povrchu tělesa jednoduše $d = \frac{c.t}{2}$. Dálkoměr však detekuje pouze vzdálenost jednoho bodu přímo ve směru pohledu, proto je potřeba buď otáčet přímo jím, a nebo, což je častější případ, lze pohled dálkoměru ovlivnit pomocí systému pohyblivých zrcadel. Obrovskou výhodou tohoto skeneru je jeho rychlost. Typický model dokáže sejmout 10 až 100 tisíc bodů za vteřinu. Mezi další výhody patří jeho dosah, díky němuž je ho možné využít například pro skenování budov. Naopak jeho nevýhodou je poměrně nízká přesnost.

Triangulační skener

Jedná se také o skener využívající laser, viz obr. 2.2, avšak má naprosto odlišné vlastnosti než laserový skener, tzn. dokáže snímané objekty zachytit s vysokou přesností, avšak jeho dosah je velmi omezený. Srdcem triangulačního skeneru je opět laserový dálkoměr, ale dále využívá také fotoaparát, který snímá místo, na které laser svítí. V závislosti na vzdálenosti skenovaného subjektu se laserová tečka na jeho povrchu objeví na určitém místě snímku fotoaparátu. Tato technika se nazývá triangulace právě proto, že fotoaparát, laserový emitor a laserový bod tvoří trojúhelník. Mezi triangulační skeny patří například ABW-3D, který byl použit pro pořízení dat Baselského modelu, který je dále využíván v této práci.

Obrázek 2.2: Triangulační skener

Světelný skener (Structured-light scanner)

Tento typ skenerů vrhá na předmět svazek světla a sleduje, jak je světlo zdeformováno na jeho povrchu. Svazek může být jednorozměrný a nebo dvourozměrný. Nespornou výhodou tohoto typu je jeho rychlost, tím že najednou neskenuje pouze jeden bod, nýbrž více bodů nebo dokonce celé zorné pole. To také snižuje nebo zcela eliminuje problém

zkreslení pohybu. Některé současné systémy jsou dokonce schopny snímat pohybující se objekty v reálném čase. Real-time skenery byly vyvinuty pro zachycení a rekonstrukci dynamických objektů, kde potřebujeme vysokou hustotu informace – například výrazy lidského obličeje. Nejlepší real-time skenery zvládají získávat skeny rychlostí 120 obrazů za sekundu.

2.2 Pořízení vlastních trojrozměrných dat popisující povrch lidské tváře

Již v bakalářské práci byl řešen problém s hledáním vhodného přístroje pro pořízení vlastní kolekce dat. Po několika rozličných návrzích byl zvolen Kinect Xbox360. Kinect byl vybrán nejen pro jeho dostupnost, ale také pro relativně krátký snímací čas. Vzhledem k tomu, že by námi vyvíjený algoritmus měl mít praktické uplatnění, jsou obě tyto skutečnosti velmi žádoucí.

2.2.1 Kinect Xbox360

Kinect, viz obr. 2.3, byl původně vyvinut jako vstupní zařízení k herní konzoly Xbox360 schopné snímat hráčův pohyb [2][10]. Jeho základem je webová kamera, která uživateli umožňuje ovládat a pracovat s Xbox360, aniž by člověk musel použít jakýkoliv jiný herní ovladač, pouze pomocí gest a přirozených pohybů. Projekt byl tedy zaměřen k rozšíření typické hráčské základny, což se podle statistik také povedlo (v prvních 60 dnech na trhu se prodalo neuvěřitelných 8 milionů kusů, čímž se zapsal do Guinessovy knihy rekordů jako nejrychleji prodávaný elektronický výrobek).

Obrázek 2.3: Zařízení Kinect Xbox360

Základem tohoto přístroje je 3D světelný skener. Používá svazek strukturovaného infračerveného světla a je schopen zaznamenávat pohyb uživatele real-time. Dále je přístroj vybaven RGB kamerou a mikrofonem, díky čemuž je Xbox360 schopný rozpoznat pohyb celého těla nebo obličeje, ale také je schopný rozpoznat lidský hlas. Původní verze Kinectu možnost rozpoznání hlasu neměla, nicméně později bylo rozhodnuto, že bude implementována do všech nových výrobků. Akustické zařízení je nyní schopno hlas nejen lokalizovat, ale také potlačit okolní šumy. Mikrofon obsahuje čtyři mikrofonní kapsle a operuje s 16-bitovým zpracováním zvuku se vzorkovací frekvencí 16kHz. Vzhledem k našim záměrům byl však mikrofon pro další práci irelevantní.

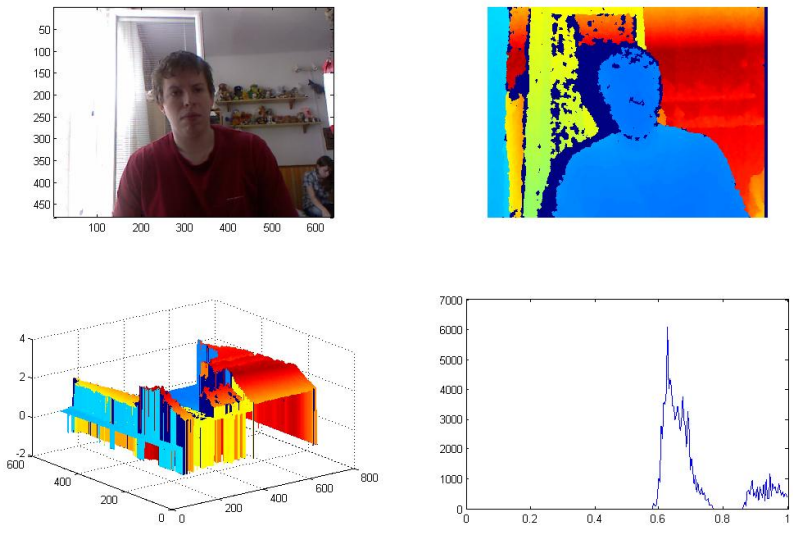
Samotné hloubkové čidlo se skládá z infračerveného laserového projektoru v kombinaci s monochromatickým (tzv. snímá daný objekt ve stupních šedi) CMOS senzorem, který je schopný snímat video za jakýkoliv světelných podmínek. Senzor Kinectu má rozsah 1.2 – 3.5 m s původním softwarem od výrobce. Výrobcem bylo rozhodnuto, že výstupní obraz Kinectu bude video o obnovovací frekvenci 30Hz a s rozlišením 640x480

pixelů. Senzor má základní zorné pole 57° horizontálně a 43° vertikálně, ale mechanický pivot je schopen natočit senzor o dalších 27° nahoru a dolů. To znamená, že při minimální pozorovací vzdálenosti, která je přibližně 0.7 m je rozlišení senzoru zhruba 1,3 mm na pixel. Co se týče softwaru dodávaného od výrobce, tak podle prodejce dokáže současně sledovat až 6 lidí, z čehož u dvou z nich zvládá aktivně sledovat i jejich gesta a obličeje.

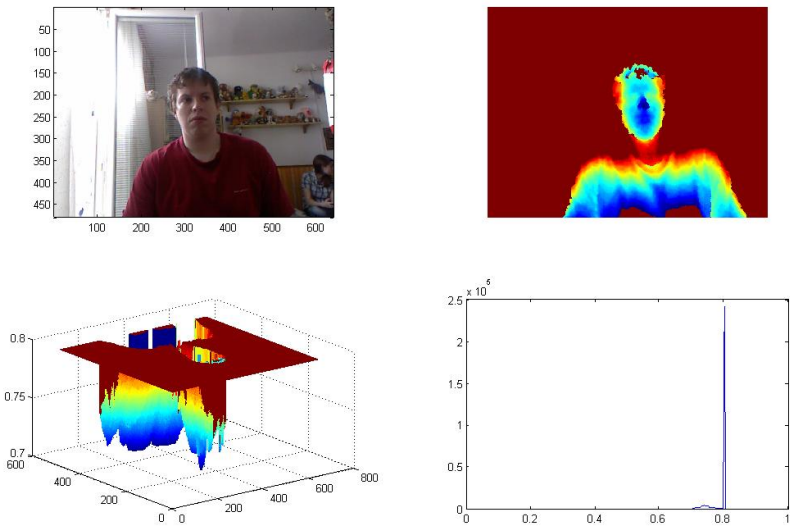
V bakalářské práci se Kinectem pořízená data ukázala jako lehce pořiditelná a jednoduše zpracovatelná, avšak přinášející i určité nedostatky. Mezi ty patřil například fakt, že minimální snímací vzdálenost je, jak již bylo řečeno, 0.7 metru, což znamená, že obličej snímaného subjektu zabírá minoritní část pořízeného snímku. Dále občas dochází ke vzniku tzv. děr ve snímku, čemuž se naštěstí dá zabránit vhodným výběrem snímků před zpracováním. Řešením těchto problému by byl nový Kinect for Windows, který disponuje vylepšenou kamerou a novým čidlem díky čemuž lze snímat obraz už na vzdálenost 40 cm bez ztráty ostrosti. Bohužel se toto zařízení nepodařilo pro naše účely sehnat, proto jsme se museli spokojit se starším modelem.

2.2.2 Pořízení snímků

Pro účely této diplomové práce jsem pořídil snímky několika různých osob pomocí jednoduchého skript v programovacím prostředí MATLAB. Tento skript ukládá data ve formě hloubkové mapy do dvourozměrného pole o rozměrech 640x480, což je taktéž rozlišení snímků. Pořízený obraz byl zároveň oříznut a to tak, že hloubka menší než 0.7 metru a větší než 0.8 metru byla "vynulována",tzn. jednotně nastavena na hodnotu 0.8 metru. Toto nastavení bylo vybráno pro účely odstranění nežádoucích objektů v pozadí a zároveň jsme tím získali pěkné oříznutí obličeje a odfiltrování krku a těla snímaného subjektu.

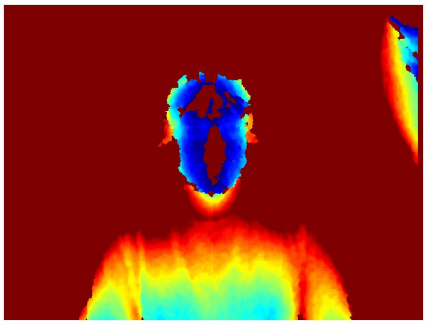


Obrázek 2.4: Původní snímek z Kinectu (hloubková mapa vpravo nahoře)



Obrázek 2.5: Snímek po hloubkovém ořezu

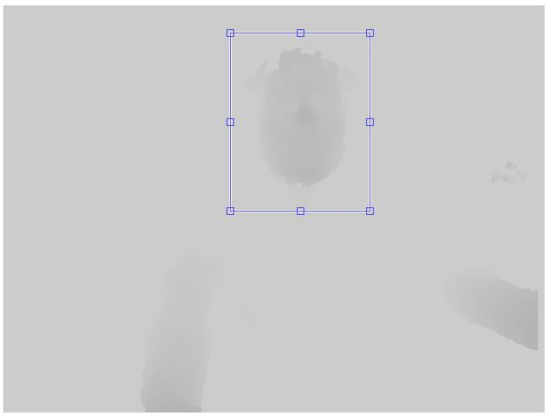
Celkem bylo nasnímáno 9 různých osob, přičemž bylo u každé pořízeno 15 až 20 snímků. Tyto snímky bylo nutné dále probrat a odstranit ty, jenž byly zatíženy příliš velkou chybou. Tato chyba vzniká ze dvou hlavních důvodů - snímaný byl moc blízko nebo naopak moc daleko od Kinectu, v důsledku toho se v obličeji subjektu objeví "nulové plochy" neboli "díry", viz obr. 2.6. Dále byly přednostně vybírány takové obrázky, na kterých se subjekt tváří přirozeně (tzv. zavřená ústa, obě oči otevřené, neutrální výraz tváře).



Obrázek 2.6: Snímek obsahující "díry"

Jak jsme již zmínil výše, obličej subjektu bohužel zabírá minoritní část obrazu, kromě něj se na snímku objevují často také jeho ruce, tělo, člověk sedící vedle něj a mnoho dalších nežádoucích objektů. Proto se bylo potřeba rozhodnout, zda zvolit nějaký druh prohledávání v obrázku, který by nám obličej skenovaného subjektu našel a automaticky ořízl, nebo zda plynule navázat na bakalářskou práci a využít již implementovaného ručního ořezu pomocí MATLABovké funkce *imcrop*. Vhledem k tomu, že ruční ořez se v bakalářské práci osvědčil jako funkční a dostatečně efektivní (a také vzhledem k

tomu, že lokalizace oblasti tváře na snímku není cílem této DP), rozhodl jsem se jej ponechat, mimo jiné taktéž pro pozdější porovnání efektivity obou programů, viz obr. 2.7. Další nespornou výhodou ručního oříznutí je praktická absence možnosti chybného ořezu a následného vyseparování špatné části obrázku. Vzhledem k faktu, že Baselský



Obrázek 2.7: Obraz hloubkové mapy během ořezávání

model nepracuje v metrických délkách, bylo nutné zkorigovat tato měřítka oříznuté části skenu, viz obr. 4.14. Tomuto problému se budu hlouběji věnovat později, viz. kapitola 4.

Obrázek 2.8: Oříznutá část skenu

Kapitola 3

Stávající metody přizpůsobení modelu lidského obličeje podle trojrozměrných dat

Pro další práci bylo stěžejní prostudovat stávající metody přizpůsobení 3D modelu, zvláště metody týkající se přizpůsobení lidského obličeje. S vyšším rozlišením a přesností dnešních 3D skenů se procento chyby prudce snížilo, díky čemuž je lze využít například pro rozpoznávání osob, což v minulosti nebylo zdaleka možné. Výhodou 3D skenovaných dat je oproti 2D barevným datům to, že se rozdílnost osvětlení projeví mnohem méně. Avšak i přes velký pokrok oproti dřívějším letům, 3D skeny stále trpí občasnými chybějícími body. Proto je potřeba, aby metody rozpoznávání byly k těmto "dírám" invariantní, popřípadě je potřeba tyto chybějící oblasti doplnit interpolací. Tímto problémem se zabývá například Laplaceovo vyhlazení (Laplacian smoothing) [15], jehož úspěšnost je však velmi závislá na rozlišení zpracovávaného skenu. Dále je možné zde použít nejrůznější druhy filtrů, jako například řádkový filtr, mediánový filtr atd.

Další problém, který u 3D rozpoznávání (přizpůsobení) tváře nastává je fakt, že používané metody jsou závislé na výrazu tváře. Tímto problémem jsem se nemusel zabývat díky tomu, že lidé na mnou pořízených datech mají neutrální výraz a zároveň Baselský model, který jsem použil, je vytvořen také z dat tváří s neutrálními výrazy. Existuje mnoho metod, které se používají pro přizpůsobování 3D modelu a každá má své výhody i nevýhody, přičemž každá je více či méně vhodná pro daný model. Rád bych zde alespoň některé nejzajímavější zmínil:

3.1 Metoda B. ter Haara a Remco C. Veltkampa

Jedná se o plně automatickou metodu přizpůsobení 3D modelu trojrozměrným datům. [11]. Pro testování byl vytvořen model obsahující původně 953 skenů různých osob s neutrálními výrazy tváře, na které byla později použita PCA metoda a to tak, aby jednotlivé složky byly nezávislé. Jelikož testovaná metoda pracuje pouze se skeny s neutrálními výrazy tváře, nelze ji použít na 3D fitting tváře s jinými výrazy, avšak pozdější testy ukázaly, že si dokáže poradit i s jemnými nuancemi (například lehce přivřené oči, drobný úsměv atd.).

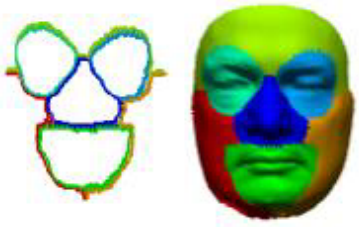
V prvním kroku této metody je hledán nos subjektu na skenu, v druhém celkový

tvar tváře, přičemž tento tvar je získán oříznutím veškerých data, která mají větší euklidovskou vzdálenost od nosu než 100 mm, viz obr. 3.1. Původně se uvažovalo o rozšíření této vzdálenosti, aby byl započten i krk a uši, ale bylo od toho opuštěno.

Obrázek 3.1: Oříznutí tváře podle Euklidovské vzdálenosti od nosu

Na samotné vyčíslení, zda je model kvalitní aproximací dodaného 3D skenu, je v této metodě použita střední kvadratická vzdálenost každého vertexu modelu k nejbližšímu bodu na 3D skenu tváře, přičemž nejsou brány krajní body skenu. Ke každému bodu instance (tváře vytvořené modelem), je nalezen bod na skenu obličeje, který má minimální Euklidovskou vzdálenost.

Pro účel hledání optimálního řešení byl zvolen iterativní downhill walk algoritmus. Při tomto procesu je prohledáván m-dimenzionální prostor, přičemž m je počet komponent, který získali po aplikaci PCA metody (v tomto konkretním případě 99), aby našel instanci modelu s minimální chybou. Bohužel donwhill walk algoritmus se často zasekl v globálním minimu. Tento problém byl vyřešen rozdělením tváře na tři segmenty, které byly hodnoceny nezávisle, přičemž v každé iteraci se měnil pouze jeden z nich. Při dalším testování však vznikly problémy při počátečním zarovnání instance průměrné tváře na 3D sken obličeje, v případě že se od ní hodně lišil. Proto se v první iteraci prohledává pouze prvních 10 komponent a instance se na sken hrubě "namontuje". Později byl algoritmus dále vylepšen a to tak, že tvář byla rozdělena na sedm částí, viz obr. 3.1, díky čemuž je algoritmus nyní schopen generovat i nesymetrickou lidskou tvář.



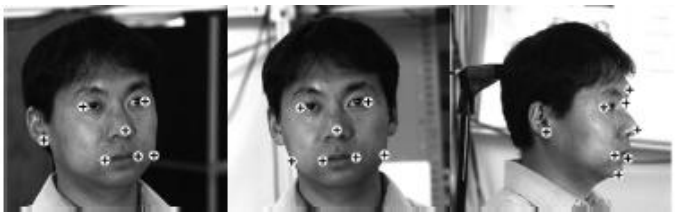
Obrázek 3.2: Rozdělení tváře na sedm segmentů

3.2 Metoda Volkera Blanze a Thomase Vettera

Jedná se o poloautomatickou metodu přizpůsobení 3D modelu 2D datům [5][6], která byl opět testována na databázi, na níž byla aplikována PCA metoda (vzniklo 200 komponentů, z nichž bylo použito prvních 99). Tato metoda se zabývá pouze vykreslením tváří s neutrálními výrazy, stejně jako ta předchozí (v tomto případě však bylo použito 6428

snímků pro získání textury). Jejím cílem bylo zamezit vliv osvětlení a celkově špatných podmínek během skenování na výslednou instanci (například natočení hlavy atd.). Tato metoda kombinuje deformovatelný 3D model obličeje spolu s grafickou počítačovou simulací natočení a osvětlení obličeje. Z každého obrázku algoritmus dokáže automaticky odhadnout tedy nejen tvar, ale také texturu obličeje a všechny další důležité parametry vypovídající o daném skenu. Testováním bylo ověřeno, že všechny možné kombinace natočení, osvětlení atd. jsou touto metodou bez výjimky pokryty. Metoda bere v potaz dokonce i zrcadlové odrazy a stíny, které mají značný vliv na vzhled lidské kůže. Celý přístup je založen na Morphable 3D modelu, který zachycuje specifické vlastnosti tváře. Tyto vlastnosti se program automaticky učí z každého 3D skenu. Kromě samotné textury a tvaru tváře je u každé specifické vlastnosti uložen i údaj o pravděpodobnosti jejího výskytu.

Prvním krokem v tomto algoritmu je vyčíslení kvadratické vzdálenosti mezi tvarem skenu a průměrného obličeje, zároveň vyčíslení této vzdálenosti i pro texturu. Dále je manuálně na sken naneseno šest až osm tzv. featured pointů, díky kterým algoritmus získá informaci o základní pozici, tvaru a barvě obličeje. viz obr. 3.3.



Obrázek 3.3: Rozdělení tváře na sedm segmentů

Obličej byl před začátkem prohledávání opět rozdělen do segmentů, tentokrát do čtyř – oči, ústa, nos a zbytek. Během hledání správné instance program neustále bere v potaz pravděpodobnost výskytu dané vlastnosti. Toto zamezuje některým fatálním chybám. Samotná optimalizační část je vlastně modifikovaná stochastická verze Newtonova algoritmu. Ta se vyhýbá lokálním minimům díky prohledávání vždy větší části obličeje. Optimalizace je prováděna pro každý ze čtyř segmentů odděleně. Ve finále je celý algoritmus velice rychlý a efektivní. Na průměrném počítači trvá zhruba 4,5 minuty.

3.3 Metoda Pouria Mortazaviana, Josefa Kittlera a Williama Christmase

Tato metoda se zabývá problémem přizpůsobení lidské tváře podle dvourozměrného snímku s nízkým rozlišením [14]. Prostudoval jsem ji vzhledem k faktu, že pořízené snímky z Kinectu mají bohužel také nízké rozlišení a proto jsem doufal, že by její znalost mohla býti později užitečná.

Tato metoda pracuje jak s tvarem, tak s texturou obličeje. Objevitelé metody tvrdí, že chybová funkce používaná tradičními metodami přizpůsobování je v tomto případě suboptimální, jelikož tyto funkce neuvažují vliv bodové funkce šíření (point spread

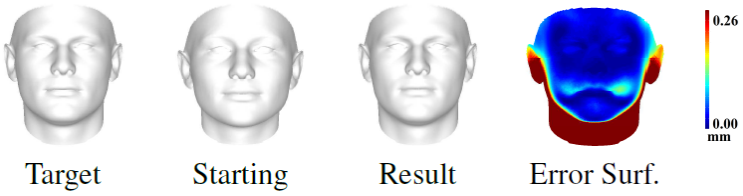
function)[1] daného fotoaparátu (kamery) a implicitně předpokládají hladký obraz. I když má tento předpoklad při vysokém rozlišení jen malý vliv, tento vliv exponenciálně stoupá s poklesem rozlišení.

Z tohoto důvodu byl navržen algoritmu, který zahrnuje jak parametry fotoaparátu, tak i bere v úvahu prostorové začlenění přes jednotlivé pixely, což má za následek mnohem přesnější modelování při nízkém rozlišení snímku. Celý proces navíc nepoužívá častou stochastickou optimalizaci, ale vyhýbá se lokálním minimům pomocí tzv. multiple fitting feature strategie. Ukázalo se, že proces přizpůsobení může být zpřesněn začleněním výše zmíněných parametrů do chybové funkce. Experimentální výsledky potvrdili, že takto upravený algoritmus překonává tradiční algoritmy z hlediska vizuální kvality i z hlediska podobnosti parametrů modelu získaných pro stejnou osobu na snímku s vysokým rozlišením.

3.4 Metoda Cladia Arellana a Rozenna Dahyota

Hlavním cílem během vývoje tohoto algoritmu bylo nalézt plně automatickou metodu přizpůsobení 3D modelu 2D datům bez referenčních bodů [4]. Metoda je založena na Bayesovských rámcích (Bayesian frameworks), ale na rozdíl od jiných metod využívající tyto rámce zohledňuje Gaussovského rozložení pro pravděpodobnosti řešení. Výsledná cenová funkce je optimalizována pomocí algoritmu střední změny (Mean shift algorithm). Konvergence tohoto algoritmu ke globálnímu řešení je vylepšena použitím metody simulovaného žíhání, viz. 3.5.2, které se osvědčilo, vzhledem k faktu, že se ve většině případů úspěšně vyhýbá globálnímu minimu. Metoda pracuj pouze s neutrálními výrazy tváře a pouze s tvarem (tzn. vynechává texturu obličeje).

K testování tohoto algoritmu byl využit, stejně jako v mojí práci, Baselský model, viz. 4.1. Během experimentu byla vygenerována tvář pomocí vektoru náhodných parametrů, přičemž tato tvář byla dále považována za neznámou instanci, jejíž parametry chceme získat. Algoritmus se tedy pokusil najít původní vektor s co nejmenší chybou, přičemž tato chyba byla tradičně počítána jako Euklidovská vzdálenost mezi vygenerovanou a cílovou instancí.



Obrázek 3.4: Přizpůsobení 3D modelu

Získané výsledky, viz obr. 3.4, jsou velmi uspokojivé a metoda se ukázala býti velmi robustní a spolehlivá. Další testování ukázalo, že je algoritmus dostatečně účinný i pro zpracování 2D dat (testování probíhalo ns modelu lidské ruky).

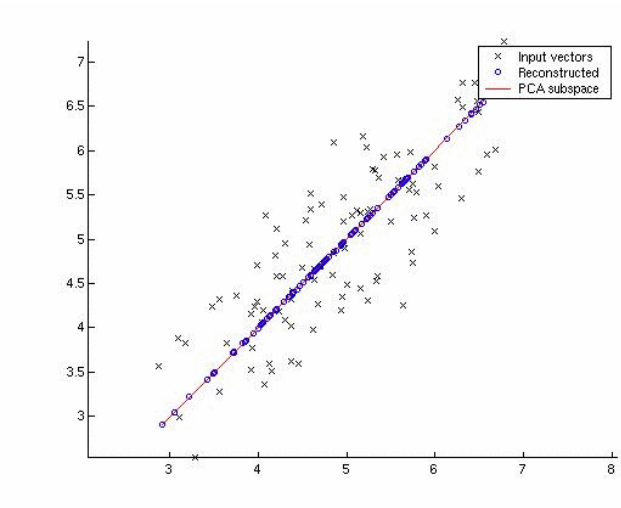
[1] Většina běžných objektivů nejsou dokonalé optické systémy. V důsledku toho jsou vizuální data po průchodu objektivem lehce degradována. Velikost a způsob této degradace popisuje právě tato funkce [1].

3.5 Důležité metody využité v této práci pro zpracování 3D dat

3.5.1 Analýza hlavních komponent (PCA)

Analýza hlavních komponent (Principal Component Analysis) [12], někdy také nazývána metoda extrakce příznaků, je matematická technika, která využívá ortogonální transformaci k přeměně naměřených dat, která mohou být korelovaná na soubor nekorelovaných dat. Tato nekorelovaná data nazýváme hlavní složky (principal components). Jedná se o metodu redukce dimenze dat, která se používá v případě, že máme rozsáhlá data s mnoha neznámými a předpokládáme, že v těchto neznámých existuje nějaká redundance. V tomto případě je redundancí myšlena právě výše zmíněná korelace mezi jednotlivými naměřenými hodnotami. Díky redundanci je možné snížit počet naměřených neznámých na soubor, který bude představovat nejdůležitější naměřenou informaci. Proto je vždy počet hlavních složek nižší nebo roven počtu původních měření. Hledání redukce se provádí, jak již bylo řečeno, pomocí transformace, která je definována, aby vždy první složka měla co nejvyšší odchylku a každá následující co nejvyšší odchylku splňující podmínku ortogonality. V případě, že data navíc splňují vícerozměrné normální rozdělení, je zajištěna jejich nezávislost, v ostatních případech tomu tak není. Bohužel data použitá v práci (Baselský model) tuto podmínku nesplňovala, a proto jsem nemohl zkoumat základní složky obličeje každou zvlášť, což by velice zjednodušilo a urychlilo prohledávání prostoru řešení, vzhledem k faktu, že pokud by se jednotlivé komponenty byly nezávislé, mohli bychom vždy prohledávat pouze jednodimenzionální prostor pro každý parametr zvlášť.

Samotná PCA byla vymyšlena v roce 1901 anglickým matematikem Karlem Pearsonem. Nyní se používá hlavně pro účely průzkumné analýzy dat a pro prediktivní modely. Její nespornou výhodou je její šetrnost k relativnímu měřítku původního souboru dat. Jedná se také o nejjednodušší vícerozměrnou analýzu vlastních vektorů.



Obrázek 3.5: Graf ukazující využití PCA k nalezení 1D reprezentace vstupních 2D dat s minimální rekonstrukční chybou

3.5.2 Simulované žíhání (SA)

Simulované žíhání neboli simulated annealing je jednou z heuristických metod používaných k nalezení optimálního řešení složitých kombinatorických úloh [18]. Tyto úlohy mají většinou natolik rozsáhlý prostor přípustných řešení, že není v rozumném čase možné všechna řešení otestovat a najít nejlepší z nich. Tato metoda má fyzikální základ (žíhání tuhého železa) a inspiruje se tzv. horolezeckým algoritmem, který systematicky prohledává stavový prostor všech řešení a snaží se najít řešení optimální. Jedná se vlastně o gradientní metodu, kdy se směr nejprudšího spádu určí prohledáním okolí. Tento algoritmus zároveň potřebuje pro své fungování funkci, která je schopná ohodnotit jakékoliv řešení patřící právě do prostoru všech přípustných řešení (funkce je nazývána účelová nebo také cenová). Pokud tato funkce neexistuje, nelze daný problém řešit tímto algoritmem. Princip horolezeckého algoritmu lze popsat následujícími kroky:

1. na počátku je náhodně vygenerováno řešení $x_0 \in X$ a položí se rovno řešení výchozímu.
2. v dalším kroku se postupně generuje použitím konečného souboru transformaci k řešení ležící v určitém okolí výchozího řešení $x : x_i \in U(x), i = 1, ..., k$.
3. z těchto řešení je vybráno to, které má z hlediska účelové funkce nejvyšší ohodnocení, tedy nejmenší hodnotu cenové funkce.
4. toto vybrané řešení je prohlášeno za nový střed okolí.
5. body 2, 3 a 4 se neustále opakují, dokud počet kroků nedosáhne předem dané hodnoty, poté algoritmus končí.
6. během výpočtu je zaznamenáváno nejlepší nalezené řešení a to je na konci algoritmu prohlášeno za výsledek.

Nevýhodou tohoto přístupu je, že během celého řešení jsou brány v potaz pouze stejná nebo lepší řešení než bylo řešení předchozí. Díky tomu se často stává, že algoritmus uvízne v lokálním minimu blízko od počáteční vygenerované hodnoty a tudíž již nikdy nedosáhne optimálního řešení. Tento nedostatek je možné odstranit opakovaným spuštěním algoritmu a tím náhodně zvolit jiné počáteční řešení, to je však spojeno s určitou časovou náročností. Dalším problémem, který může nastat, je možnost zacyklení, při kterém se opakovaně vracíme k lokálnímu řešení. Tento problém řeší úspěšně metoda Tabu prohledávání (Tabu Search), která si uchovává historii již vyzkoušených řešení, takže zbývá efektivně vyřešit pouze problém uvíznutí v lokálním minimu.

Tento nedostatek odstraňuje metoda simulovaného žíhání, která využívá stochastické operátory a na rozdíl od horolezeckého algoritmu s jistou pravděpodobností (danou Metropolisovým kritériem, viz rovnice 3.1, 3.2) i horší řešení než výchozí řešení x. Metoda také navíc prohledává celý prostor řešení. Dalším rozdílem je, že se negeneruje okolí a nevybírá nejlepší řešení, ale určitým operátorem se stochasticky transformuje výchozí řešení x na nové x'. Tímto krokem je zajištěno rozptýlení v celém prostoru a nejen v jeho malé části.

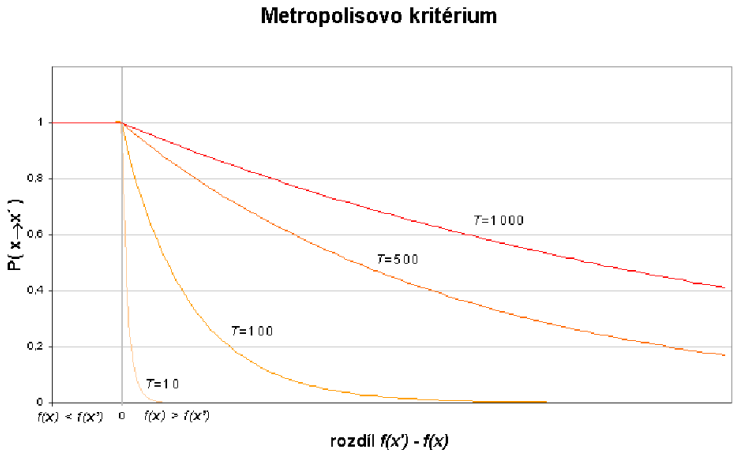
Jak již bylo řečeno, algoritmus je inspirován žíháním tuhého železa, během kterého je vysoká teplota železa na počátku pomalu snižována na teplotu mnohem nižší. Tímto jevem se na začátku osmdesátých let nechali inspirovat pánové Kirkpatrik, Gelatt a Vecchi

z výzkumného centra IBM v USA a nazvali tento nový přístup k hledání globálního minima simulované žíhání. V první fázi vývoje bylo potřeba nahradit fyzikální realizaci numerickou simulací. Inspiraci pro to nalezli v Metropolisově algoritmu. V souvislosti s ním uvažujeme o nějakém stavu systému x. Tento stav se změní na nový stav $x' = O_{pert}(x)$, kde O_{pert} je stochastický operátor poruchy. A právě to, zda bude nový stav akceptován, je vyhodnocováno podle Metropolisova kritéria, které určuje pravděpodobnost nahrazení starého novým.

$$P(x \to x') = 1, f(x') \leq f(x) \tag{3.1}$$

$$P(x \to x') = e^{\frac{f(x')-f(x)}{T}}, f(x') > f(x) \tag{3.2}$$

Parametr T je takzvaná teplota systému. Jak vidíme, tak nový stav x' je akceptován, pokud má stejnou nebo menší funkční hodnotu než původní stav. Pokud je tomu naopak, je akceptován s pravděpodobností určenou právě Metropolisovým kritériem, viz obr. 3.6



Obrázek 3.6: Průběh Metropolisova kritéria v závislosti na rozdílu f(x')>f(x) a nastavené teplotě

Jak je vidět, hodnota parametru T podstatně ovlivňuje hodnotu pravděpodobnosti přijetí nového stavu, který je horší než námi nalezený. Pro velké hodnoty T je tato pravděpodobnost blízká jedné, a proto bude akceptováno téměř každé nové řešení. Naopak pokud se T bude blížit k nule, budou nové stavy akceptovány pouze výjimečně. Simulované žíhání je opakované využití Metropolisova algoritmu s postupným snižováním teploty T. Na počátku, kdy je hodnota T ještě vysoká, jsou algoritmu dovoleny velké skoky napříč celou množinou přípustných řešení. S postupným snižováním T se pravděpodobnost takových skoků snižuje, tudíž lze hovořit o dolaďování nalezeného výsledku.

Je třeba ještě zmínit, co vlastně je operátor O_{pert}. To silně závisí na typu řešené úlohy. Nás nejvíce zajímá případ, kdy je řešení vektor složený z n na sobě nezávislých reálných proměnných (později se bude jednat o vektor příznaků mezi kterými prohledáváme). V tomto případě je modifikována náhodná složka vektoru přičtením náhodné hodnoty. Závěrem je třeba říci, že úspěch této metody silně závisí na zvolení počáteční hodnoty T, která je však případ od případu jiná.

Kapitola 4

Vlastní postup přizpůsobení modelu podle 3D dat

V této kapitole je uveden vlastní postup přizpůsobení modelu lidského obličeje. Nejdříve ze všeho je ale třeba uvést několik údajů ohledně zvoleného animačního modelu. Byl ponechám stejný model jako v BP a to model Univerzity v Baselu.

4.1 Model Univerzity v Baselu

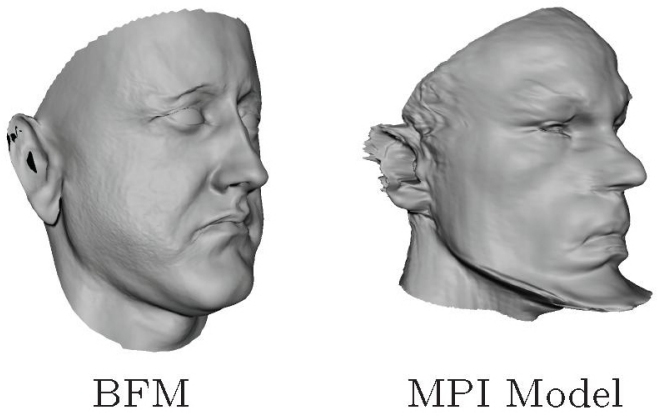
Baselský model (3D Basel Face Model, dále již jen BFM) byl zvolen nejen pro jeho dostupnost, ale také pro jeho kvalitu a příjemné uživatelské prostředí [10]. Další nespornou výhodou je fakt, že je naprogramován ve vývojovém prostředí MATLAB, se kterým jsem měl již určité zkušenosti z dřívějška. Poslední, ale nikoliv nejméně důležitou, výhodou je jeho snadná modifikovatelnost, čehož jsem při práci s ním hojně využíval.

4.1.1 Obecné parametry

Baselský model je psaný ve vývojovém prostředí MATLAB, vytvořený a poskytovaný primárně pro výzkumné a studijní účely [3][16]. Model umí nejen vykreslit různé tvary lidského obličeje, ale dokáže na ně i aplikovat texturu, tudíž je možné vygenerovat jak lidi rozličné hmotnosti a věku, tak i různé pleti.

Původní 3D Morphable modely měly mnoho problémů. Mezi ně patřilo například obtížné rozpoznávání tváří, které nebyly zachyceny z čelního pohledu. Dalším problémovým jevem bylo různé nasvícení tváří trénovacích množin, které se pro tyto programy používaly. Toto a několik dalších věcí činilo pořizování trénovacích množin velmi obtížným. BFM všechny tyto problémy řeší a poskytuje tak uživateli velmi mocný nástroj pro generování a rozpoznávání lidské tváře. BFM, na rozdíl od dřívějších 3D Morphable modelů (dále jen 3DMM), nabízí lepší vykreslení tvarů a vyšší přesnost aplikace textury. Také přináší možnost aplikace stejného 3D modelu obličeje jak na 3D, tak na 2D obrázek, pomocí různých syntetických metod. Další nepostradatelnou výhodou, kterou BFM svému uživateli poskytuje, je možnost vygenerování lidské tváře s libovolným podsvícením, natočením a úhlem. Toto všechno vnáší Baselský model mezi světovou špičku 3DMM. Podle institutu v Baselu existují pouze dva srovnatelné modely, a to model Max-Planckova institutu v Tubingenu (MPI) a model Univerzity Jižní Floridy

(USF). V porovnání s těmito dvěma modely má BFM výhodu ve dvou aspektech. Za prvé, 3D skener (ABW-3D), viz. 2.1.2, používaný pro BFM nabízí vyšší rozlišení a přesnost v kratším čase potřebném pro oskenování obličeje než Cyberware skener, který používají právě dva výše zmíněné modely. Za druhé, díky rozdílné záznamové metodě je potřeba méně korespondující pro samotné vykreslení. Tato výhoda se projeví především při vysokých hodnotách parametrů daného modelu, viz obr. 4.1.

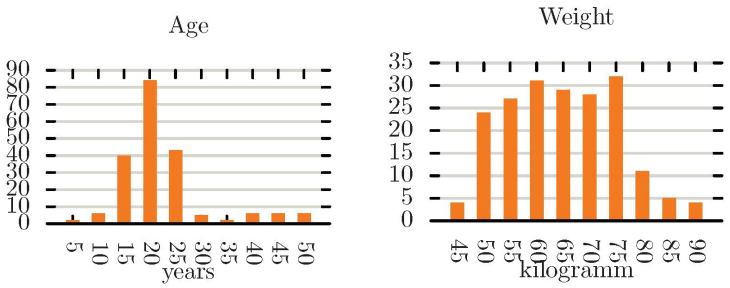


Obrázek 4.1: Porovnání BFM a MPI při extrémních hodnotách parametrů

Stejně jako pro každý jiný model podobného typu bylo pro BFM potřeba vytvořit trénovací množiny. Na rozdíl od mnoha jiných systémů 3DMM bylo rozhodnuto, že trénovací a testovací množina nebude jedna a tatáž. Takové systémy totiž potom mají obtíže s přechodem od jedné množiny k druhé. Trénovací množina obsahovala 3D skeny tváře 100 mužů a 3D skeny tváře 100 žen, povětšinou Evropanů. Muži i ženy byli pro tyto skeny vybíráni tak, aby mezi nimi byla zahrnuta celé škála odchylek a nuancí lidské populace. Věk osob se pohyboval mezi 8 a 62 lety, přičemž průměrný věk byl 24.97 roku, a váha se pohybovala mezi 40 a 123 kilogramy, průměr byl 66.48 kilogramu 4.2. Byly pořízeny tři skeny každé osoby, přičemž byl vybrán ten, kde měl daný člověk nejvíce přirozený výraz. Pro samotné skenování byl použit, jak již bylo výše uvedeno, skener typu ABW-3D, přičemž čas skenování byl nastaven na přibližně 1s. Celý skenovací systém obsahoval 2 projektory a 3 kamery, které snímaly obraz ve třech různých hloubkách. Zároveň jsou během každého skenování pořízeny tři fotky, což zaručuje vysokou přesnost barvu textury obličeje. Nevýhodou tohoto systému je fakt, že jím nelze pořídit snímek očí a vlasů kvůli jejich odrazovým vlastnostem.

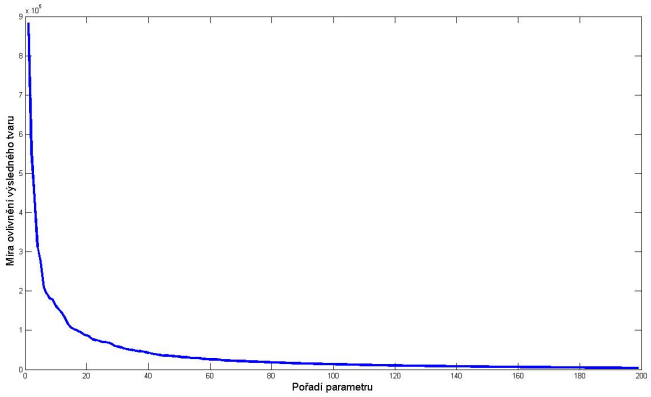
Aby mohla být samotná data ze skenů použita, je potřeba, aby existovala určitá korespondence mezi daty. Proto jsou všechny skeny reparametrizované podle určitých význačných bodů na obličeji (například nos, koutky oči atd.). Pro získání této korespondence byl použit algoritmus optimálního proměnného kroku. Pro zvýšení kvality modelu byly dále ručně přidány mezníky (orientační body) na rty, obočí a uši. Samotná textura je generována z dat ze tří fotografií pořízených během skenu. Pro její zlepšení byly dále manuálně odstraněny z modelu vlasy.

Ve finále je celý model prezentován jako trojúhelníková síť s $m = 53490$ body a se sdílenou topologií, přičemž každý bod má přiřazenou správnou barvu. Na 200 snímků lidských tváří byla aplikována již výše zmíněná metoda PCA, díky níž jsme získali nejen



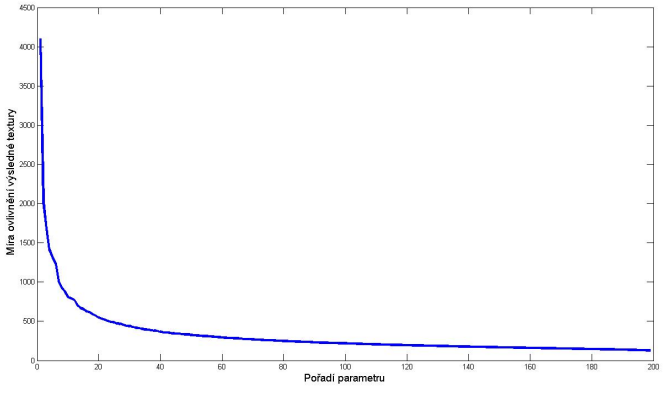
Obrázek 4.2: Grafy stáří a hmotnosti dobrovolníků

podobu "průměrné tváře", ale také 199 parametrů modelu, seřazených podle jejich vlivu na tvar (dalo by se také říci podle důležitosti). Během dalšího zkoumání jsem zjistil, že vliv jednotlivých skenů na tvar obličeje se exponenciálně snižuje ,viz. obr. 4.3, a proto jsem později operoval pouze s několika prvními příznaky.



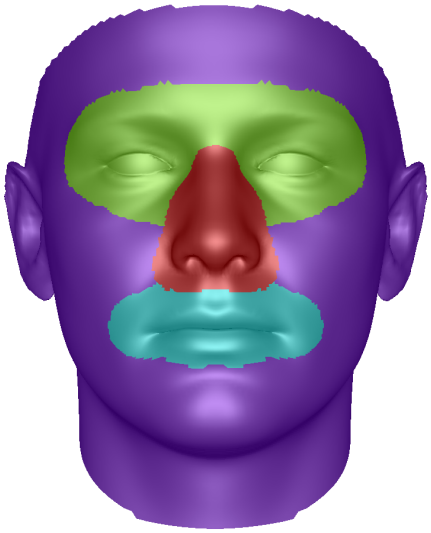
Obrázek 4.3: Vliv parametrů modelu na jeho celkový tvar

PCA metoda byla aplikována stejně tak na textury, přičemž jak můžeme vidět na následujícím grafu ,viz obr. 4.4, opět se jejich vliv na výslednou texturu exponenciálně snižuje. Vzhledem k cíli této práce textura nebyla dále testována.



Obrázek 4.4: Vliv parametrů modelu na výslednou texturu

Pro větší flexibilitu byl model dále rozdělen do čtyř segmentů, viz obr. 4.5, a to nos, ústa, oči a zbytek tváře. Od návrhu tohoto faktu využít po vzoru metody B. Haara a C.



Obrázek 4.5: Rozdělení tváře do čtyř segmentů

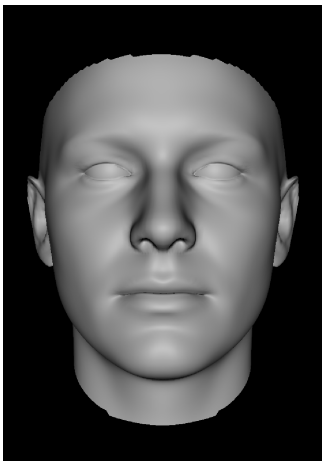
Veltkampa, viz 3.1, bylo nakonec upuštěno, vzhledem k pozdějšímu zjištění, že parametry vzniklé PCA analýzou ovlivňují tvar obličeje jako celek a změny tvaru jednotlivých segmentů tudíž nelze jednoduše separovat.

Model tedy obsahuje několik částí: průměrný tvar a průměrnou texturu tváře, 199 komponent pro tvar a dalších 199 komponent pro texturu tváře, dále síťovou topologii odchylky tvarů a odchylky textury. K modelu je navíc přiloženo několik funkcí naprogramovaných ve vývojovém prostředí MATLAB, které slouží k práci s modelem. Tyto funkce jsou uživatelsky velmi příjemné a lehce modifikovatelné. Pro účely této práce jich několik bylo upraveno a dále použito.

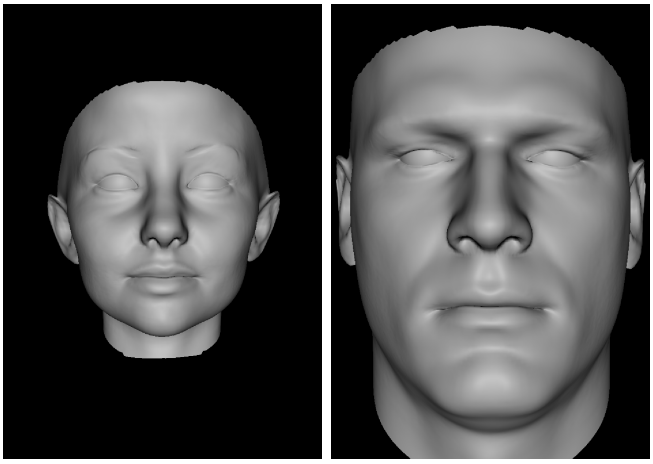
4.1.2 Vliv jednotlivých parametrů na tvar obličeje

Nyní bych pro představu rád uvedl několik snímků, na kterých je demonstrován vliv prvních tří parametrů na celkový tvar výsledné instance lidského obličeje. Vzhledem k faktu, že vliv parametrů exponenciálně klesá, dále jsem pracoval již pouze s prvními deseti parametry, což se ukázalo být jako ideální kompromis mezi možností tvar instance měnit a časovou náročností během jejich zpracování.

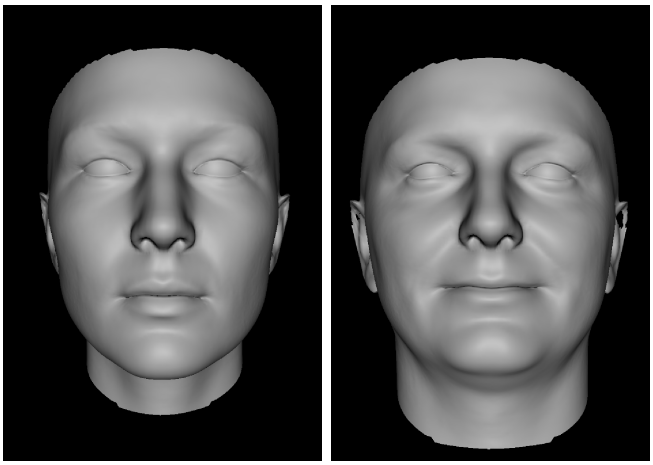
Jak vidíme na těchto snímcích, již na první pohled je poznat, že vliv jednotlivých parametrů klesá, přičemž kladné hodnoty komponent obličej formují spíše do dětského/ženského obličeje, zatímco záporné hodnoty dávají instanci mužné tvary.



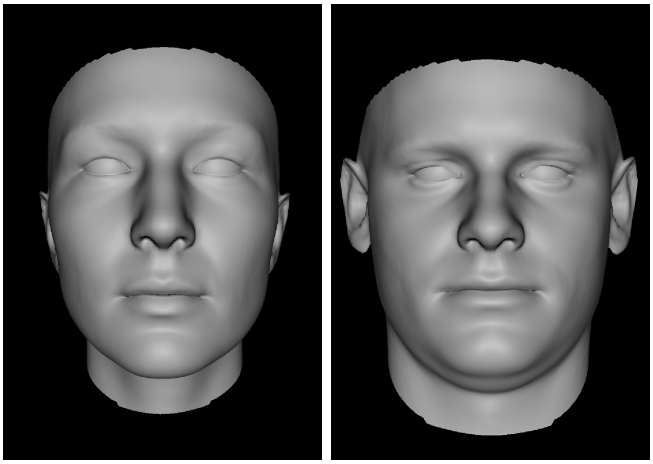
Obrázek 4.6: Průměrný obličej



Obrázek 4.7: Vliv první obličejové komponenty: +5x/-5x



Obrázek 4.8: Vliv druhé obličejové komponenty: +5x/-5x



Obrázek 4.9: Vliv třetí obličejové komponenty: +5x/-5x

4.2 Funkce využité pro práci s modelem

Jak již bylo v předešlé kapitole uvedeno, BFM kromě samotného modelu nabízí i několik funkcí pro práci s tímto modelem. Některé z těchto funkcí se zabývají práci s texturou, ty nebyly dále nijak využity. Další se zabývají jak texturou, tak tvarem, to znamená, že pokud jsem je chtěl dále využít, bylo je potřeba upravit, aby práce s texturou zbytečně nezabírala výpočetní kapacitu. Nyní bych rád uvedl několik funkcí, dodané k modelu, které byly pro účely této práce využity.

1. *load_model* - jedná se o pomocnou funkci, která načte všechny části modelu (viz. 4.1.1) potřebné pro další práci.

2. *coef2object* - funkce, jejíž vstupem jsou koeficienty pro jednotlivé atributy (tzn. vstupem je vektor o 199 složkách, každá z nich představuje jeden z komponentů lidské tváře), výstupem je vektor o 160470 složkách (53 490 bodů x 3, tzv. 1x za každý rozměr), který představuje výsledný tvar obličeje. V této práci ji používám ke generování jednotlivých instancí, které se snažím přizpůsobit dodané hloubkové mapě. Původně tato funkce mohla stejným způsobem (vstupem byl opět vektor o 199 složkách) generovat i texturu obličeje ale vzhledem k tomu, že jsem s texturou nijak nepracoval, byla tato možnost odstraněna.

3. *defrp* - jedná se o funkce, díky které lze přenastavit původní tzv. rendering parametry, ovlivňující vykreslování jednotlivých instancí lidské tváře.

4. *display_face* - úkolem této funkce je vytvořit instanci (obraz) obličeje. Vstupem jsou vektory vygenerované funkcí *coef2object* jak pro tvar, tak pro texturu, dále rendering parametry, které dodává funkce *defrp*. Výsledkem je obraz obličeje s aplikovanou texturou. Generování textury bylo v rámci snížení výpočetní náročnosti opětovně odstraněno. Funkce byla později dále upravována, abychom získali možnost s vygenerovanou instancí i libovolně natáčet a posouvat ji v rámci snímku.

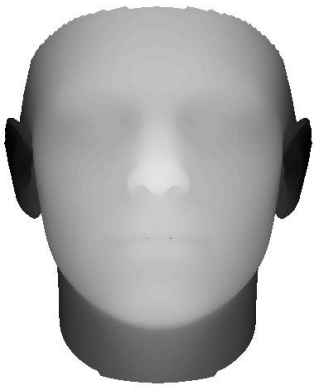
5. *object2coef* - tato funkce pracuje na přesně opačném principu než funkce *coef2object*, neboli vstupem je vektor o 160470 složkách a výsledkem jsou koeficienty pro jednotlivé komponenty obličeje. Vektor však musí mít správné adresování příznaků

(to znamená, že například údaj o poloze špičky nosu musí být vždy na určeném místě, konkrétně u BFM je to bod číslo 8321). Tento fakt však mnou pořízené skeny z Kinectu zdaleka nesplňovaly, proto jsem tuto funkci dále nijak nevyužil. Zajímavostí je, že i tato funkce funguje nejen pro tvar, ale také pro texturu.

4.3 Srovnání 3D dat a modelu

Před samotným algoritmem přizpůsobování bylo třeba převést získané 3D data a BFM do takového tvaru, aby tyto dvě na sobě navzájem nezávislé instance byli vůbec srovnatelné. Zatímco snímky z Kinectu byly reprezentovány hloubkovou mapu ve formě matice 640x480 (po ořezu samozřejmě méně), jednotlivé tváře vygenerované BFM byly uloženy jako vektor o 160470 složkách, které přestavovali 53490 bodů v prostoru tvořící lidský obličej (160470/3 = 53490, tzn. že původní vektor obsahuje prostorové souřadnice každého z těchto bodů). Bohužel nebylo možné využít funkci *object2coef* zmíněnou výše a to kvůli nesplněnému adresování příznaků skenu.

Po několika pokusech bylo zjištěno, že nejrozumnějším řešením bude převést obraz vykreslený pomocí funkce *display_face* na hloubkovou mapu. K tomuto účelu mi posloužila MATLABovská funkce *colormap*. Tato funkce nanese na místa obrázku barvu podle z-tové souřadnice v daném místě, přičemž nejvhodnější pro naše účely byla colormapa v odstínech šedi, která nanese na místa s nejnižší z-tovou souřadnicí (nejvzdálenější body) černou barvu, naopak čím je bod blíž, tím je vybarven světlejší barvou, viz obr. 4.10. Pomocí další MATLABovské funkce *getframe* jsem pak již jednoduše získal reprezentaci ve formě hloubkové mapy.



Obrázek 4.10: Průměrný obličej s aplikovanou colormapou

Bohužel tato metoda s sebou přinesla jistá úskalí a to ve formě faktu, že funkce *colormap* vždy nanese bílou barvu na nejbližší bod a černou na nejvzdálenější nezávisle na tom, jaké tyto hodnoty vzdálenosti doopravdy jsou, tzn. barvy nezávisí na absolutních z-tových hodnotách, ale pouze na relativních. Vzhledem k faktu, že každá instance BFM má například nos v jiné relativní hloubce (tzn. nos má jinou z-tovou souřadnici), vnášela by tato skutečnost do zpracování značnou nepřesnost. Proto je během každého vykreslení přidán do obrázku referenční bod, který má větší z-tovou souřadnici než je možné dosáhnout při rozumných hodnotách protažení obličeje. Dále je do obrázku přidán také další referenční bod, který má naopak menší z-tovou souřadnici než je obvyklé. Tím

je zajištěno to, že colormapa aplikuje na tyto body krajní hodnoty barev a na lidský obličej již příslušné odstíny šedi. Tyto body mají navíc tu výhodu, že nejsou příkazem *getframe* brány v potaz, proto nemusí být dále řešeny.

4.4 Korekce měřítek a úprava rozměrů

Dalším nutným krokem před samotným přizpůsobováním modelu byla korekce měřítek a úprava rozměrů. Po aplikaci colormapy se hodnoty hloubkové mapy BFM pohybovaly mezi 0 a 255 (bylo aplikováno 256 odstínů šedi), zatímco hodnoty hloubkové mapy získané z Kinectu mezi 0.8 a 0.7 (tyto hodnoty přímo odpovídají vzdálenosti snímaného subjektu od senzoru). Naštěstí hodnota 0 přímo korespondovala s hodnotou 0.8 a hodnota 255 s hodnotou 0.7, což celou korekci značně ulehčilo. Přicházely v úvahu dvě varianty: korekce dat z Kinectu nebo korekce instancí BFM. Byla zvolena první možnost z toho důvodu, že během samotného přizpůsobování modelu 3D datům je generováno nesčetně instancí tohoto modelu, kdyby byla zvolena druhá možnost, bylo by nutné provádět korekci pro každou z nich, takto je korekce provedena pro každý snímek pouze jednou, viz obr. 4.11.

Obrázek 4.11: Změna měřítek skenu z Kinectu

Posledním velkým problémem byla rozdílná měřítka jednotlivých matic představujících hloubkovou mapu. Funkce *display_face* zobrazuje BFM v rozlišení 640x480, což by se mohlo zdát býti ideální, vzhledem k tomu, že skeny pořízené Kinectem mají totéž rozlišení. Nesmíme však zapomenout na fakt, že obličej z těchto skenů je dále vyříznut funkcí *imcrop*, viz obr. 2.7,tudíž je finální rozlišení nižší. Naštěstí tento problém řešila funkce *defrp*, která přímo umožňuje nastavit rozlišení, v jakém má být každá instance BFM zobrazena.

Naneštěstí se během ořezávání objevil další problém. Obličej zabírá u BFM průměrně 40% plochy obrázku, při ořezu pomocí funkce *imcrop* tomu tak bohužel není. Pokud totiž například ořízneme sken tak, že obličej zabírá menší plochu, algoritmus se bude domnívat, že se jedná o obličej dítěte a tomu se také bude snažit přizpůsobit jednotlivé parametry modelu. Finální instance pak bude nejspíš mít dětské rysy, navzdory tomu, že subjektem zachyceným na skenu mohl být dospělý člověk. Podobná extrémní situace

může nastat i v opačném případě, kdy algoritmus bude považovat subjekt na skenu nejspíše za velkého muže.

Dalším nedostatkem tohoto postupu je skutečnost, že instance vygenerovaná BFM je vždy perfektně vycentrovaná, zatímco u našeho oříznutého snímku tomu tak, vzhledem k manuálnímu ořezu, být nemusí.

Tyto problémy bylo třeba dále řešit, jelikož by do algoritmu mohly vnášet poměrně velkou chybu. Oba byly později vyřešeny poskytnutím BFM lehké volnosti v osách, natočení a zvětšení během hledání řešení. Tímto se podrobně zabývá podkapitola 4.7.

4.5 Vyhledávací algoritmus parametrů modelu

Před výběrem optimalizačního algoritmu bylo třeba zvolit způsob, jakým bude vyčíslována chyba mezi skenem a BFM. Chybu jsme se rozhodli vyčíslit absolutně následujícím způsobem:

$$\Delta_{abs} = \sum |s_{i,j} - m_{i,j}| \tag{4.1}$$

kde $s_{i,j}$ je bod 3D skenu o souřadnicích i,j a $m_{i,j}$ je odpovídající bod instance BFM.

Pro výpočet této chyby a pro automatické vygenerování instance BFM v každé iteraci byla vytvořena funkce *Vypocetni_funkce*. Jejím vstupem je vektor *coef* hodnot jednotlivých komponent modelu a výstupem je výše uvedená chyba. Tato funkce tedy pracuje následujícím způsobem:

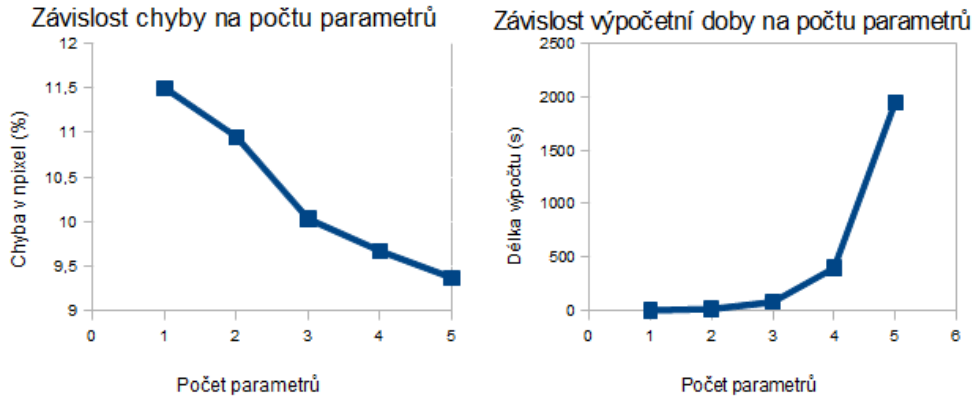
1. vstupní vektor *coef* předložen funkci *coef2object* ke zpracování, výstupem je vektor *shape* s prostorovými souřadnicemi jednotlivých bodů výsledné instance
2. vektor *shape* je použit jako vstup pro funkci *display_face*, výstupem je zobrazení dané instance modelu
3. na zobrazení je aplikována šedotónová colormapa pomocí příkazu *colormap gray*
4. získání maticové reprezentace pomocí MATLABovské funkce *getframe*
5. vypočtení absolutní chyby a navrácení tohoto algoritmu optimalizační funkci

Samotné prohledávání prostoru řešení samozřejmě muselo startovat z nějakého stavu, dále je zřejmé, že prohledávané hodnoty je nutné omezit. Jako startovní stav byl zvolen průměrný obličej, tzn. všechny koeficienty byly nulové. Omezující podmínky byly původně pro každou složku vektoru nastaveny od -2 do 2, ale pozdější testování ukázalo, že rozšířením omezujících podmínek na interval $\langle -3; 3 \rangle$ získáme lepší výsledky.

Před jakýmkoliv dalším postupem (výběr optimalizačního algoritmu, ladění, apod.) jsem tento postu otestoval na algoritmu hrubé síly. Tento algoritmus je samozřejmě velmi neefektivní a extrémně časově náročný, nicméně by odhalil případné nedostatky, které je třeba opravit. Pro testování tohoto algoritmu byl vybrán snímek prakticky bez děr, zároveň byl proveden ořez tak, aby byl obličej subjektu co nejlépe vycentrovaný a zabíral přibližně 40% plochy výsledného snímku.

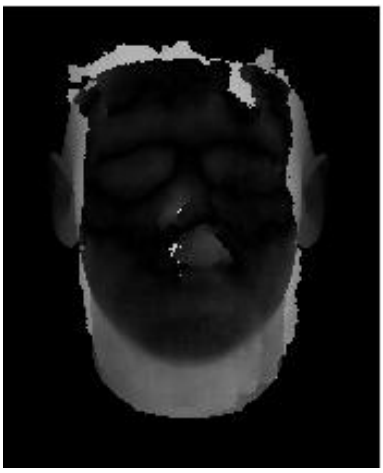
Testování probíhalo, tak, že byla v prvním testu prohledávána pouze první složka vektoru komponent, s každým dalším testováním byla jedna složka přidána navíc. Test

dopadl podle očekávání, zatímco se chyba s přidanými komponentami snižovala, výpočetní náročnost exponenciálně stoupala, viz obr. 4.12.



Obrázek 4.12: Výsledky řešení pomocí metody hrubé síly

Metoda hrubé síly odhalila další problém, se kterým původně nebylo počítáno. Tento nedostatek přímo plyne z rozdílnosti skenů a BFM. Zatímco skeny obsahují vlasy, BFM nikoliv, a zatímco na skenech obvykle nejsou zachyceny ani uši ani krk (krk se na některých snímcích objevuje, avšak pouze jeho nepatrná část), BFM obě tyto části lidského obličeje zřetelně vykresluje. Díky této skutečnosti se bohužel optimalizační algoritmus pokouší "napasovat" krk instance na bradu skenu a uši na bok hlavy, viz obr. 4.13, čímž vzniká velká nepřesnost a finální řešení nemá s původní předlohou prakticky nic společného. Tyto problémy bylo třeba před jakýmkoliv dalším postupem odstranit.

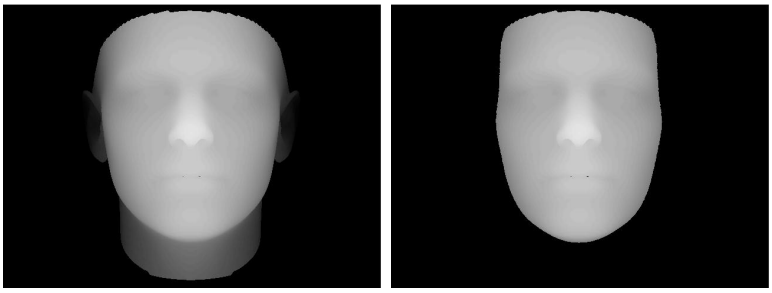


Obrázek 4.13: Výsledné rezidua vzniklá odečtením instance BFM od 3D snímku

4.6 Oříznutí obličeje

Jak již bylo řečeno, metoda hrubé síly nám odhalila určité nedostatky, které bylo třeba odstranit. Hlavním důvodem byla rozdílnost skenu a BFM, proto bylo nutné se těchto rozdílů (vlasy, uši, krk) nějak zbavit.

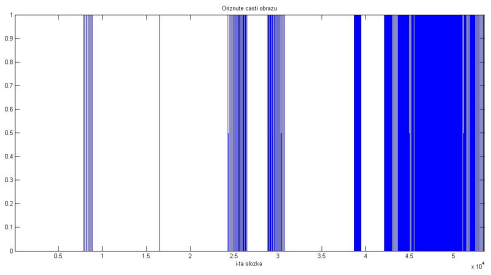
Po vzoru B. Haara a R. Veltkampa, viz kap. 3.2, jsem zamýšlel obličej vytvořený BFM oříznout v závislosti na vzdálenosti od špičky nosu. Jelikož jsem však neznal přesné měřítko, zvolil jsem podmínku pro ořez tak, že se brala v úvahu pouze vzdálenost ve smyslu z-tové souřadnice od bodu, který ji měl nejvyšší (špička nosu). Hraniční hodnotu jsem nalezl experimentálně. Výsledné oříznutí fungovalo tak, že všechny hodnoty nad hraniční byly ponechány, zatímco ostatní vynulovány. Tímto postupem jsem získal poměrně pěkný ořez obličeje bez krk a uší, viz obr. 4.14.



Obrázek 4.14: Průměrná tvář před ořezem a po něm

Ač by se mohlo zdát, že je tento postup bezchybný a vyřešil tak výše uvedené problémy, není tomu tak. Problém nastává, když tento ořez implementujeme pro každou nově vygenerovanou instanci během prohledávání prostoru řešení. Díky ořezu je totiž pokaždé vykreslován jiný počet (nenulových) bodů (pro každou instanci BFM se nachází jiný počet bodů pod mezní hodnotou ořezu). Tento fakt ztěžuje optimalizačnímu algoritmu jeho práci tím, že je algoritmus nucen pracovat v každé iteraci za trochu jiných podmínek. To bezesporu vnáší do finálního řešení chybu, která nebude nikterak velká, nicméně už drobné rozdíly při zobrazení lidského obličeje (délka nosu, vzdálenost očí, apod.) kompletně mění celkový dojem z tváře člověka.

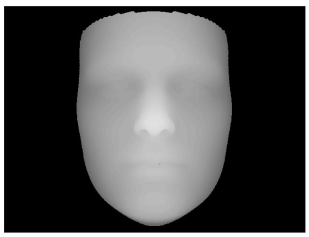
Proto jsem se rozhodl metodu ořezu upravit. Základní myšlenka byla stejná a to ta, že body pod určitou hranicí by měly být nulové, abychom odstranili krk a uši, nicméně počet vynulovaných bodů by měl být v každém kroku prohledávání stejný. To bylo vyřešeno následujícím způsobem: Před začátkem optimalizace byla vygenerována průměrná tvář a nalezeny všechny body této instance, které mají z-tovou souřadnici nižší než je experimentálně nastavená hodnota (byly nalezeny souřadnice těchto bodů ve vektoru *shape*), viz obr. 4.15.



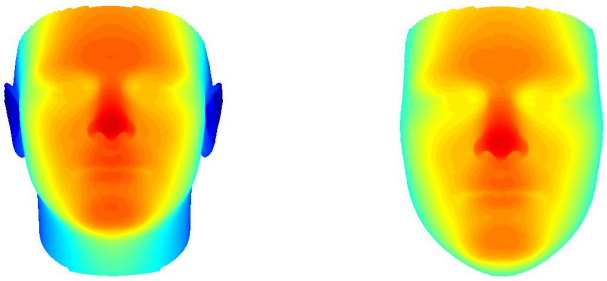
Obrázek 4.15: Souřadnice bodů s nižší z-tovou souřadnicí než práh

Tyto body jsou uloženy a nezávisle na tom, jakou z-tovou souřadnici mají v dále vygenerovaných instancích BFM, jsou vždy vynulovány. Stejně tak jakékoliv body, které

hranici překročí až během prohledávání, vynulovány nejsou, ale jsou jim ponechány původní hodnoty. Tím je zajištěno, že algoritmus má vždy stejné podmínky během každé iterace a zároveň je tím získán velmi podobný ořez,viz obr. 4.16. Robustnost ořezu byla dále ověřena na BFM s náhodně vygenerovanými koeficienty, viz obr. 4.17.
Výše uvedený postup poměrně elegantně vyřešil část problémů vzniklých rozdílností



Obrázek 4.16: Výsledek nové metody ořezu tváře



Obrázek 4.17: Výsledek nové metody ořezu tváře

skenu a modelu. Bohužel model se od skenu stále lišil tím, že neměl vlasy, zatímco Kinect je poměrně často zachytil. Automatické řešení tohoto problému se nám bohužel objevit nepodařilo, jelikož na získaných datech nelze nijak jednoduše oddělit vlasy a čelo snímaného subjektu. To znamená, že odstranění vlasů by se muselo buď řešit manuálně nebo by na pořízení snímků bylo nutné použít jiný typ skeneru. Ani jedna z těchto variant nebyla akceptovatelná, proto se dále snažíme vytvořit algoritmus dostatečně robustní na to, aby si s těmito nedostatky dokázal bez dalších zásahů poradit.

4.7 Vyhledávací algoritmus polohy

Oříznuté snímky pořízené Kinectem s sebou přináší mnoho úskalí. Jedním z nich je fakt,že snímek není nikdy perfektně vycentrovaný, ač se o to sebevíc pokoušíme. Řešením by mohlo být rovnoměrné dooříznutí okrajových částí, avšak to by nám pomohlo pouze ve vodorovné linii, ve svislé nikoliv, jelikož neexistuje žádný záchytný bod, o který bychom se mohli opřít. Proto je třeba dát prohledávání prostoru řešení určitou relativně malou volnost v pohybu modelu, jak ve směru osy X tak ve směru osy Y. Dalším problémem je natočení samotného obličeje, ať už se jedná o rotaci kolem jakékoliv z os, či o jejich kombinaci. Rotace kolem osy Z by se dala poměrně jednoduše, například určením pozice koutků očí a následné rotace o příslušný úhel tak, aby byly ve stejné výšce, nicméně pro ostatní rotace opět neexistuje žádný záchytný bod. Z těchto důvodů

je opět nutné do vyhledávacího algoritmu implementovat volnost v natočení ve všech třech osách.

Prvotním nápadem bylo rozšířit původní vyhledávací algoritmus o několik dalších parametrů a dát mu tudíž volnost jak v pohybu s modelem ve směru osy X a Y, tak volnost v natočení kolem všech tří základních os v prostoru. Toto řešení by bylo určitě správné, nicméně výpočetní doba vyhledávacích algoritmů prudce stoupá s každým novým uvažovaným parametrem, v našem případě by těchto parametrů bylo pět (původně zamýšlený počet, ve finále byl rozšířen na sedm). Proto jsme se rozhodli hledání správného řešení rozdělit na dvě 'nezávislé' části. Nezávislé v tom smyslu, že v první fázi bude hledána pouze správná poloha tváře a to se základní (průměrnou, neutrální) tváří BFM a ve druhé fázi, po nalezení přibližné polohy, budou hledány samotné parametry BFM. Během tohoto řešení může sice dojít k drobným nepřesnostem, vzhledem k tomu, že je měněna poloha pouze průměrné tváře, nicméně následné testování ukázalo, že snížení výpočetní doby tyto nepřesnosti naprosto převáží.

4.7.1 Natočení

Natočení BFM je implementováno pomocí obecné matice rotace, kterou je přenásobena matice tvaru obličeje (*shape*) ještě před jeho vykreslením, tzn. rotujeme se samotným modelem. Tato operace byla implementována přímo do funkce *display_face* a její vstupní parametry tedy byly rozšířeny hodnoty natočení v jednotlivých osách.

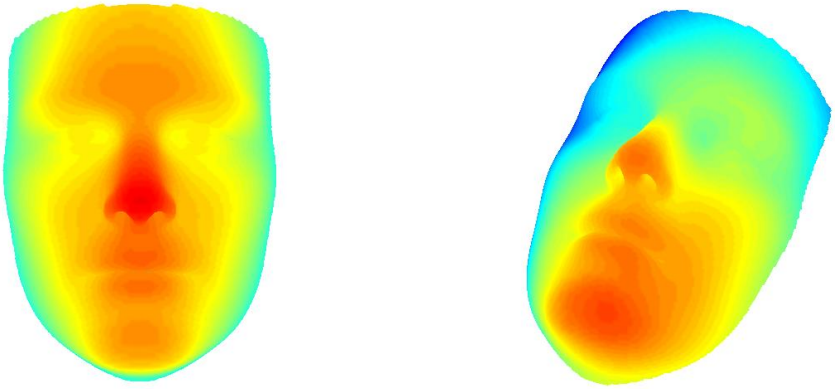
Druhou možností bylo rotovat s pohledem, ze kterého je již vygenerovaný model snímán, avšak námi zvolená možnost je výpočetně rychlejší (díky velice rychlému násobení matic v MATLABu) a v případě potřeby drobných úprav také programátorsky přívětivější. Další nespornou výhodou první možnosti je její snazší srozumitelnost pro někoho, kdo se setká s tímto algoritmem v budoucnu poprvé. Zde uvádím obecné matice rotace pro jednotlivé osy:

$$R(\omega_x) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\omega_x) & sin(\omega_x) \\ 0 & -sin(\omega_x) & cos(\omega_x) \end{bmatrix} \tag{4.2}$$

$$R(\omega_y) = \begin{bmatrix} cos(\omega_y) & 0 & -sin(\omega_y) \\ 0 & 1 & 0 \\ sin(\omega_y) & 0 & cos(\omega_y) \end{bmatrix} \tag{4.3}$$

$$R(\omega_z) = \begin{bmatrix} cos(\omega_z) & sin(\omega_z) & 0 \\ -sin(\omega_z) & cos(\omega_z) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.4}$$

Tímto krokem jsem získali poměrně šikovný nástroj, který nám dovoluje vygenerovanou instanci BFM libovolně natáčet ve všech třech osách, viz obr. 4.18. Zároveň se zde projevuje užitečnost přepracované metody ořezu, která nám ořezává přesně ty části modelu, které chceme, nezávisle na tom, jakou mají body z-tovou souřadnici po rotaci.

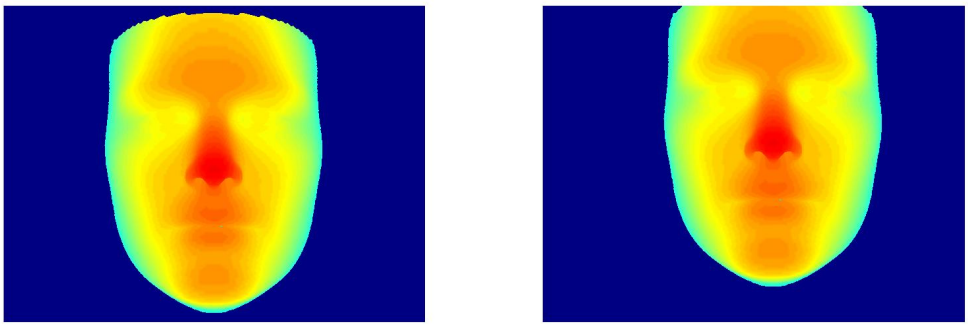


Obrázek 4.18: Natočení tváře

4.7.2 Posunutí v osách X a Y

Původně bylo v plánu implementovat posunutí podobným způsobem jako rotace, tzn. posunout model ještě před jeho vykreslením, nicméně taková to implementace by potřebovala přeprogramovat celou vykreslovací funkci. Ta byla totiž původně vytvořena tak, aby se vygenerovaný obličej vždy perfektně vycentroval.

Proto byla zvolena metoda posunu pohledu na již vygenerovaný BFM. K tomuto účelu byla využita MATLABovská funkce *Position*. Funkce *Position* má celkem čtyři vstupní parametry: První parametr funkci říká, o kolik má pohled na graf posunou v horizontálním směru, zatímco druhý mluví o směru vertikálním, viz obr. 4.19. Zbylé dva parametry jsme nevyužili, jelikož jejich hodnoty mění velikost grafu (okénka, kterým se na graf díváme), opětovně v horizontálním a vertikálním směru. Tento posun byl taktéž implementován do funkce *display_face* stejně jako rotace.

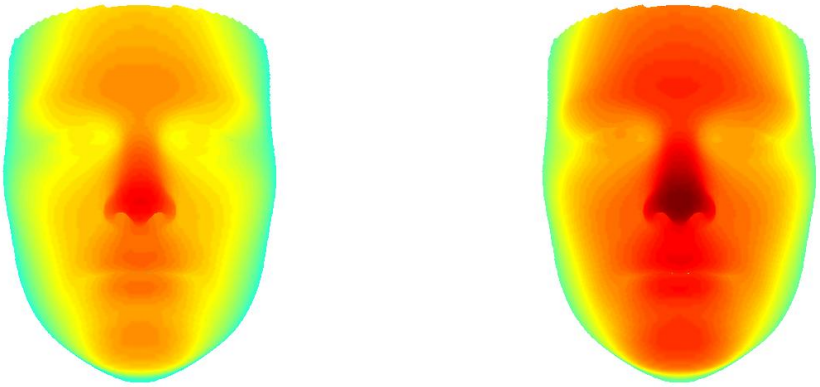


Obrázek 4.19: Posunutí tváře (kvůli názornosti i s ohodnoceným pozadím)

4.7.3 Posunutí v ose Z

Pro posunutí v ose Z jsem bohužel v MATLABu neobjevil žádnou využitelnou funkce (například nějaký posun vykreslovaného objektu do grafu ve směru osy Z nebo posun pohledu blíže k vykreslovanému objektu), proto bylo třeba vymyslet vlastní alternativu řešení.

Ta se naštěstí poměrně nabízela a to v podobě námi přidaných referenčních bodů, viz kap. 4.3. Mluvíc o těchto bodech, je nutné podotknout, že jejich přidání do obrazu bylo třeba provádět až po rotaci modelu. Kdyby byly tyto akce uskutečněny v opačném pořadí, referenční body by nám taktéž rotovaly a jejich výsledný efekt na srovnání měřítek jednotlivých instancí by minul účinkem. Nicméně tyto body byly dále využity k posunu modelu podél osy X a to tak, že z-tová souřadnice prvního referenčního bodu byla přenásobena koeficientem posunu. Tím bylo docíleno změny měřítka ve smyslu osy Z, která zároveň simuluje posun tváře směrem "dopředu/dozadu", viz obr. 4.20.



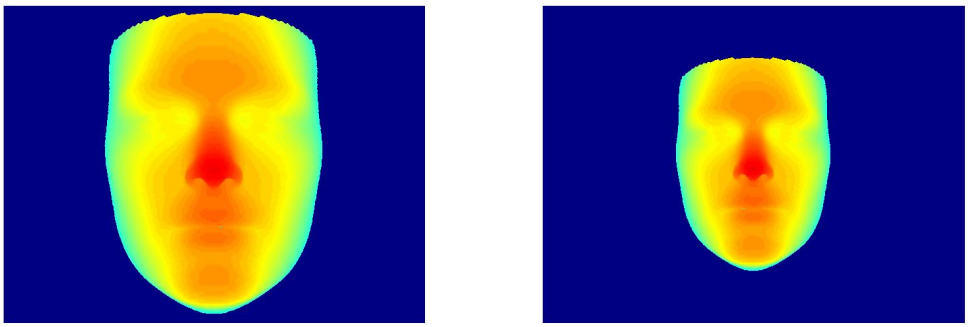
Obrázek 4.20: Posunutí tváře ve směru osy Z "dopředu"

4.8 Změna velikosti obličeje

Posledním problémem, který bylo třeba vyřešit, je velikost plochy, kterou zabírá vygenerovaná tvář v celkovém obrazu. Jak již bylo řečeno v kap. 4.4, vzhledem k využití funkce *imcrop* vzniká problém s tím, že průměrný obličej instance BFM zabírá zhruba 40% obrazu, zatímco při manuálním ořezu skenu tomu tak být vůbec nemusí.

Prvním nápadem bylo opětovně využít funkci *Position*, viz kap. 4.7.2. Změna velikosti okna by nám pomohla změnit poměr jaký tvář na snímku zabírá a kdybychom velikost okna měnili v obou osách ve stejně, zůstal by zároveň i zachován poměr stran. Bohužel by se tím ale změnilo rozlišení obrázku, což by nám velice zkomplikovalo další počínání, proto byla tato varianta zamítnuta.

Tato překážka byla tedy nakonec překonána pomocí MATLABovské funkce *PlotBoxAspectRatio*, ta totiž mění velikost ideálního zobrazovacího okna a tím plně splňuje naše požadavky. Tato funkce nám tedy umožňuje měnit část obrazu, jaký zabírá lidský obličej beze změny celkového rozlišení nebo parametrů modelu, což ji činí ideální pro vyřešení tohoto problému, viz obr. 4.21.



Obrázek 4.21: "Zmenšení" obličeje (kvůli názornosti opět i s ohodnoceným pozadím)

4.9 Optimalizační algoritmy

Nyní již zbývalo jen najít dostatečně efektivní optimalizační algoritmus, který by prohledával prostor přípustných řešení. MATLAB poskytuje pro takovéto případ velkou škálu prohledávacích funkcí, které jsou v dokumentaci velmi pěkně rozebrány. Pro náš problém přicházely v úvahu dva algoritmy: jeden z algoritmů typu omezené nelineární minimalizace nebo simulované žíhání (SA), viz kap. 3.5.2. Z nelineární minimalizace byl shledán jako nejvhodnější algoritmus SQP[9] (Sequential quadratic programming - Sekvenční kvadratické programování). Oba výše uvedené algoritmy byly otestovány, viz kap. 5.1.

V obou případech byla jako startovní bod (bod, ze kterého začíná prohledávání prostoru přípustných řešení) zvolena instance průměrné tváře (tedy všechny koeficienty byly nastaveny na hodnotu 0) a hraniční podmínky byly nastaveny stejně jako u metody hrubé síly, tedy interval $\langle -3; 3 \rangle$. Tentýž optimalizační algoritmus byl použit jak na hledání parametrů BFM, tak na hledání správné polohy, přičemž startovní polohové parametry byly taktéž nulové. Posledním nastavením, kterým jsme se zabývaly, byly zastavovací podmínky optimalizace. Možností bylo hned několik: klesnutí chyby pod určitou hranici, překročení maximálního počtu iterací a nebo v případě nelineární kombinace pokles hodnoty gradientu pod určitou hodnotu. Všechny možnosti byly řádně otestovány, více v následující kapitole.

Kapitola 5

Testování navrženého postupu přizpůsobení modelu

V této kapitole bych rád uvedl výsledky několika testů, které byly provedeny pro postup přizpůsobení modelu podle 3D dat uvedený v předešlé kapitole. Testování nejprve probíhalo bez možnosti pohybu modelu, která byla implementována až později na základě výsledků z první části.

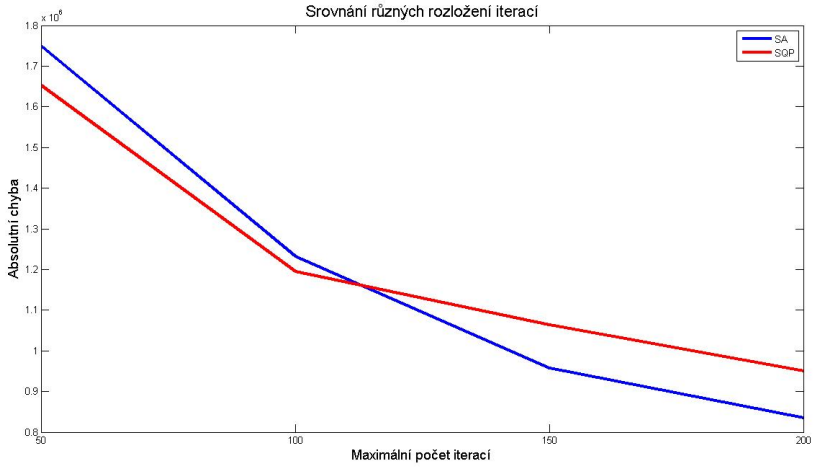
5.1 Srovnání optimalizačních algoritmů

Během těchto testů jsem se pokoušel posoudit, který optimalizační algoritmus by byl nejvhodnější pro řešení našeho problému. Jak již bylo řečeno v předešlé kapitole (4.9), byly testovány dva algoritmy: SQP a SA.

Oba byly testovány pro stejné podmínky: Bylo bráno v úvahu pouze prvních deset parametrů modelu, zastavovací podmínka pro absolutní chybu byla taktéž nastavena na stejnou hodnotu. Je nutné poznamenat, že zastavovací podmínku minimální derivační odchylky bylo třeba u algoritmu SQP zvýšit oproti původnímu nastavení, jelikož s ním SQP považoval funkci za neklesající a za optimální řešení označil startovní bod.

Tyto testy probíhaly na uměle vygenerovaných datech, tzn. jako cílovou instanci jsem zvolil obličej vygenerovaný pomocí náhodných parametrů. Tyto parametry pak samozřejmě během vyhledávání nebyly algoritmu nijak poskytnuty. Na obrázku 5.1 uvádím výsledky z několika testování.

Jak vidíme, SQP poskytuje lepší výsledky pro menší počet iterací, zatímco při vyšším počtu dosahuje lepších výsledků SA. Zároveň je třeba podotknout, že SA je mnohem méně náchylný k uvíznutí v lokálních minimech. Co se týče časové náročnosti, je SQP zhruba o 10% rychlejší. Vzhledem k získaným výsledkům, byl pro všechna další testování používán pouze SA algoritmus, který, ač se ukázal býti lehce výpočetně náročnější, poskytoval průměrně lepší výsledky co se týče jak chyby absolutní, tak subjektivně posuzované podobnosti cílové a nalezené tváře.



Obrázek 5.1: Graf porovnání optimalizačních algoritmů

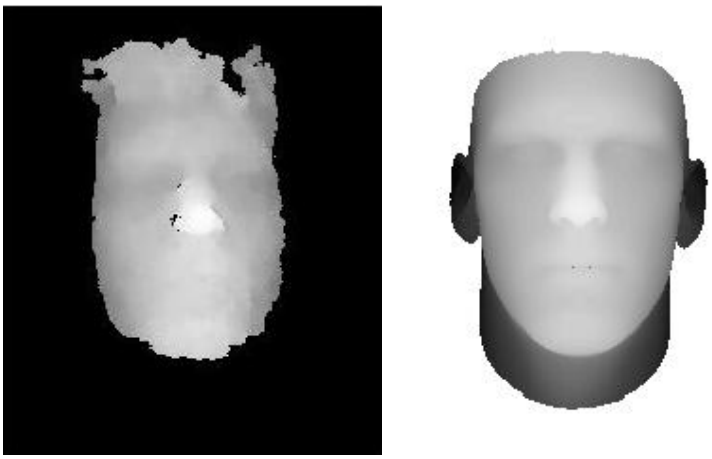
Obrázek 5.2: Výsledky testování SA algoritmu pro 1000 iterací

Obrázek 5.3: Vzniklá rezidua

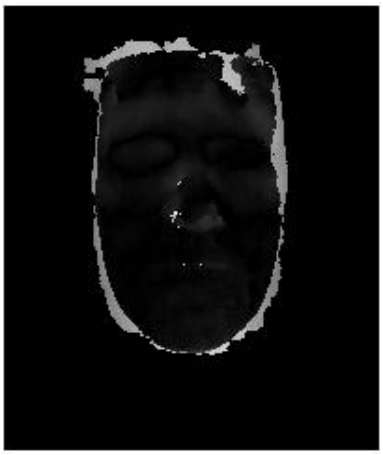
5.2 Testování SA na reálných datech

Nyní jsme přikročili k testování tohoto algoritmu na reálných datech, abychom zjistili, zda je dostatečně robustní i pro ně. Pro testování byl vybrán co nejkvalitnější sken dané osoby v tom smyslu, aby byl pokud možno bez děr, co nejlépe vycentrovaný a obličej zabíral zhruba 40% plochy. Opětovně byl prohledáván prostor prvních 10 komponent

modelu, přičemž maximální počet iterací byl nastaven na hodnotu 500.



Obrázek 5.4: Původní sken (vlevo) a výsledná instance BFM (vpravo)



Obrázek 5.5: Chyba mezi skenem a modelem (rezidua vzniklá odečtením modelu od skenu)

Jak vidíme na obrázku 5.5, chyba vznikla hlavně na okrajích obličeje, další větší chyba se objevila v okolí nosu a očí. Tyto chyby vznikly hlavně díky stacionaritě modelu, jelikož během těchto pokusů BFM ještě nemělo volnost v pohybu. Celkově řešení není zdaleka tak kvalitní, jakého jsme dosáhli u umělých dat. Chyba plyne ale nejen ze stacionarity, ale také z rozdílnosti skenu a modelu, nižšího rozlišení a z občasných děr na snímku.

5.2.1 Relativní chyba

Dalším nedostatkem, kterým bylo třeba se zabývat je absence jakéhokoliv nástroje pro porovnání výsledků dvou různých testování. Při pokusech s uměle vygenerovanými daty jsme se s tímto problémem nesetkali, jelikož zobrazení těchto dat vždy bylo v rozlišení 640x480, tudíž mohla být porovnávána absolutní chyba. V tomto případě tomu tak

bohužel není, vzhledem k rozdílnosti ořezů jednotlivých snímků. Proto byla dodefinována relativní chyba na pixel následujícím vztahem:

$$\delta_r = \frac{\Delta_{abs}}{A \cdot B} \cdot \frac{100}{255} \tag{5.1}$$

kde Δ_{abs} je chyba vypočtená rovnicí 4.1, A je šířka zobrazení v pixelech a B je výška zobrazení v pixelech. Číslo sto v čitateli druhého zlomku je koeficient převedení chyby na procenta a 255 je maximální rozdíl, kterého můžeme u dvou bodů v oblasti jasu dosáhnout.

Tímto výpočtem jsme získali univerzální nástroj pro porovnání výsledků z různých testování, ať už se jednalo o pokusy s vygenerovanými či reálnými daty.

Výpočtem jsem dodatečně zjistili, že δ_r dosažená na cvičných datech se průměrně pohybuje okolo 1% na pixel, zatímco u reálných dat je to okolo 5.5%. To by mohlo na první pohled vypadat jako dobrý výsledek, ale je potřeba si uvědomit několik věcí. Zhruba 30% obrázku zabírají místa, které v žádném případě nemohou způsobit jakoukoliv chybu (jedná se především o okraje obrázku, kam lidská tvář nikdy nezasahuje). Kromě toho musíme vzít v úvahu fakt, že pracujeme s lidským obličejem a už drobné nuance, například jiná vzdálenost očí, délka nosu apod., kompletně mění celkový dojem z tváře člověka.

5.3 Testování algoritmu vyhledávání polohy

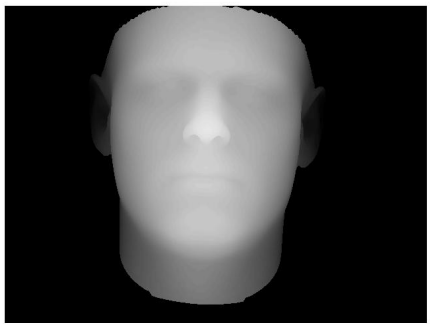
Ještě před samotným vyhledáváním bylo třeba vytvořit novou hodnotící funkci pro vyhledávání polohy. Finální varianta této hodnotící funkce se příliš neliší od původní pro parametr BFM. Na rozdíl od ní ale není v každé iteraci prohledáván prostor vektoru příznaků BFM, nýbrž polohový vektor, který je definován jako uspořádaná šestice: rotace kolem osy X, rotace kolem osy Y, rotace kolem osy Z, posunutí podél osy X, posunutí podél osy Y, posunutí podél osy Z. Optimalizace je opětovně prováděna algoritmem simulovaného žíhání. Nad celým procesem prohledávání bylo provedeno několik testování, které se týkaly jak dosavadní efektivity, tak možností vylepšení.

5.3.1 Testování efektivity

Prvním testováním bylo zjišťováno nakolik velký přínos bude mít rozšíření oproti původnímu stavu. Pomocí prvních deseti náhodně vygenerovaných parametrů BFM byla vytvořena neznámá tvář, viz obr. 5.6, kterou jsme úmyslně lehce natočily, aby byl simulován reálný stav, ve kterém snímaný člověk nikdy není v perfektní rovině.

Nejprve jsem se pokusili nalézt správné řešení, aniž bychom připustili volnost v pohybu a natočení modelu (prohledávání bylo úmyslně ukončeno po 1000 iteracích). Tím jsem simuloval dosavadní situaci během hledání optimálního řešení pro námi pořízené snímky, u kterých jsme tiše předpokládali, že snímek dotyčné osoby je perfektně vycentrovaný a bez natočení.

Výsledkem testu je průměrná **chyba 3,4% na pixel**, což je vzhledem k tomu, že se jedná o cvičná data, opravdu velká chyba. Další testování potvrdilo zřejmý fakt, že čím

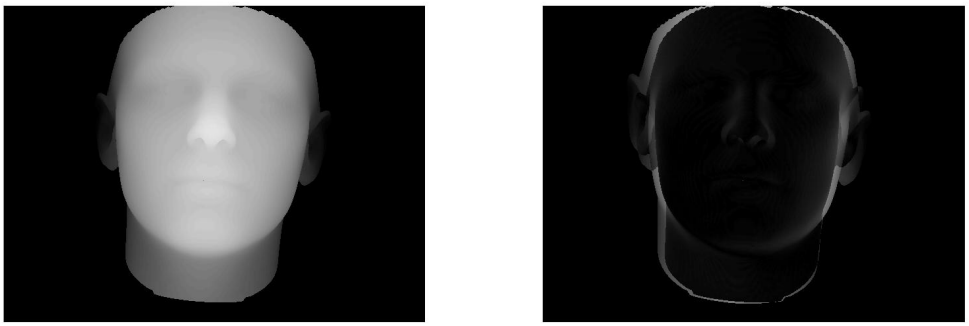


Obrázek 5.6: Původní náhodně vygenerovaný model

bude natočení/posun větší, tím bude mít algoritmus bez volnosti pohybu větší problémy nalézt správné parametry BFM a chyba bude prudce vzrůstat.

V dalším kroku bylo proto testováno prohledávání s možností drobného pohybu modelu. Testování bylo rozděleno do dvou částí. V první bylo hledáno správné řešení pouze pro polohový vektor, koeficienty BFM byly nastaveny na nulu. Správná poloha byla tedy hledána pomocí posunu/natočení průměrného obličeje. V druhé části se s polohovým vektorem nijak nemanipulovalo, ale byly měněny pouze parametry BFM. Obě části byly ukončeny po 500 iteracích.

Již po první části optimalizace se podařilo chybu snížit na **1.7% na pixel**, což znamená, že za poloviční počet iterací se nám podařilo najít řešení s poloviční chybou než v předchozím pokusu, viz obr. 5.7.



Obrázek 5.7: Výsledky po první části prohledávání (předlohou byla umělá data, parametry modelu zafixovány,hledána pouze správná poloha) (nalezená instance + vzniklá rezidua)

V druhé části pokusu byla chyba opětovně snížena, tentokrát na **1.4% na pixel**, což je zhruba 40% finální chyby z testování bez možnosti pohybu, viz obr. 5.8.

Dosažené výsledky však neodpovídaly mým představám, proto jsem se rozhodl v pokusu pokračovat a pokusil jsem se o další upřesnění polohy, tentokrát však nebylo manipulováno s průměrnou tváří, nýbrž s výslednou tváři z předchozí optimalizace, navíc startovní bod pro vektor polohy nebyl již nulový vektor, nýbrž vektor nalezený dříve. Chyba byla opětovně snížena, tentokrát na **1.1% na pixel**. Optimalizace dále

Obrázek 5.8: Výsledky po druhé části prohledávání

pokračovala opětovným prohledáváním vektoru příznaků BFM doprovázena dalším poklesem chyby, tentokrát na **0.9% na pixel**, viz obr 5.9.

Obrázek 5.9: Finální výsledky experimentu

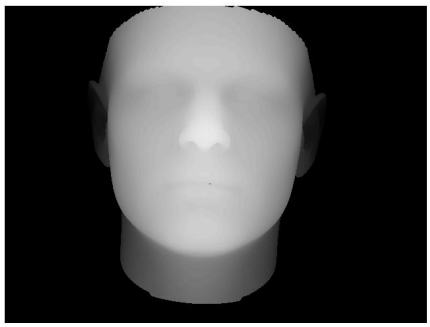
Testování nám ukázalo nakolik je drobná volnost v pohybu pro model důležitá. Původní chybu se nám podařilo snížit zhruba na čtvrtinu.

5.3.2 Testování vhodného rozdělení

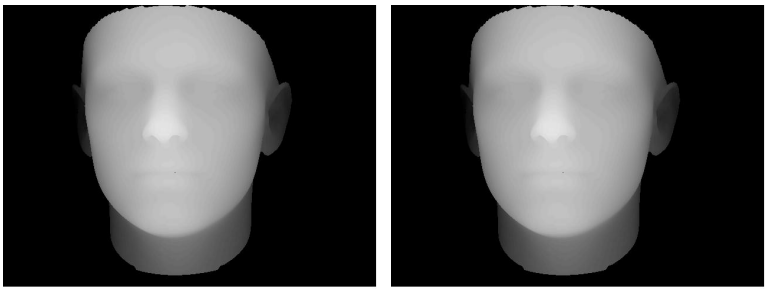
V dalším testování jsme se pokusili zjistit, zda by bylo efektivnější prohledávat v poloze pouze jednou s více iteracemi a poté pouze jednou v prostoru parametrů BFM, či zda by bylo lepší prohledávání rozdělit například na osminy a vždy po určitém počtu iterací přejít do druhé fáze a využít k dalšímu postupu nalezené mezivýsledky. Jako vzor byla opětovně vygenerována náhodná tvář, viz obr. 5.10.

Nejprve jsme se tedy pokusili nalézt správné řešení bez mezikroků, v každé fázi bylo optimalizačnímu algoritmu poskytnu 500 iterací. Po první fázi byla **2.61% na pixel**, po druhé už jen **1.87% na pixel**.

Získaná chyba je poměrně uspokojivá, nicméně jak je vidět na obrázku 5.11, algoritmu se nepodařilo nalézt správné natočení a postup bez mezikroků mu nedal šanci tuto chybu napravit. Proto byla otestována další varianta: Přepínáno mezi fázemi prohledávání bylo po 250 iteracích. Celkový počet iterací byl zachován (každá fáze tedy proběhla celkem dvakrát).

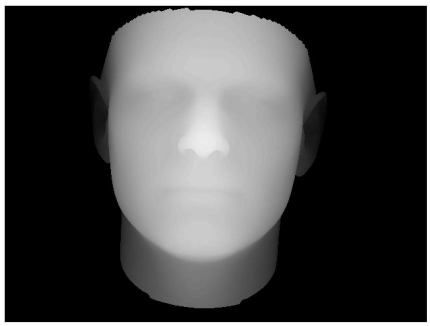


Obrázek 5.10: Náhodně vygenerovaná tvář použitá jako vzor



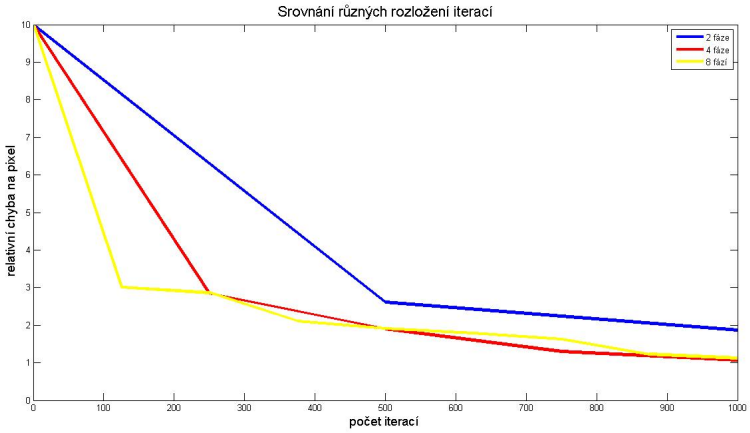
Obrázek 5.11: Výsledky prvního testování (nalezená poloha, finální instance)

Chyba na pixel se postupně snižovala a to tak, že po první fázi byla **2.84%**, po druhé **1.90%**, po třetí **1.30%** a nakonec po čtvrté **1.07%**. Výsledkem tohoto experimentu je tedy zjištění, že po stejném počtu iterací máme téměř poloviční chybu jen tím, že jsme optimalizaci rozdělili do více podfází, viz obr. 5.12.



Obrázek 5.12: Finální výsledky optimalizace

Další testování ukázalo, že dělení na ještě menší úseky by již nebylo nijak plodné a že 250 iterací na fázi je ideální minimum v námi používaném prohledávacím algoritmu, viz obr. 5.13 .

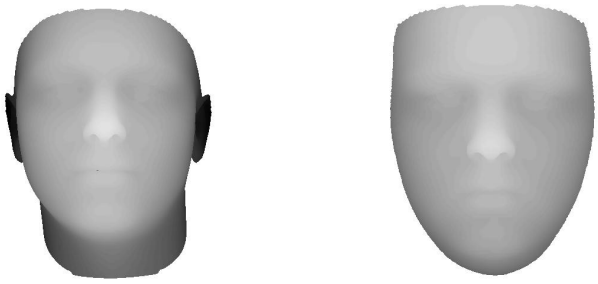


Obrázek 5.13: Porovnání efektivity různých dělení fází

5.4 Testování důležitosti ořezu

Testy v této kapitole mají ukázat, nakolik je důležitý ořez BFM i přes implementaci možnosti změny měřítka. Mohlo by se zdát, že díky této alternativě, kterou model má, by mohl být ořez zcela odstraněn a model by mohl rozdílnost skenu a BFM překlenout právě pomocí změny měřítka, bohužel testy ukázaly, že tomu tak není.

Během těchto testů byla opět vygenerována náhodná tvář bez ořezu, která byla dále považována za naši cílovou instanci. Dále se algoritmus pokusil přizpůsobit této tváři model, který byl ale oříznut.



Obrázek 5.14: Vygenerovaná tvář + nalezená instance

Obrázek 5.15: Vzniklá rezidua

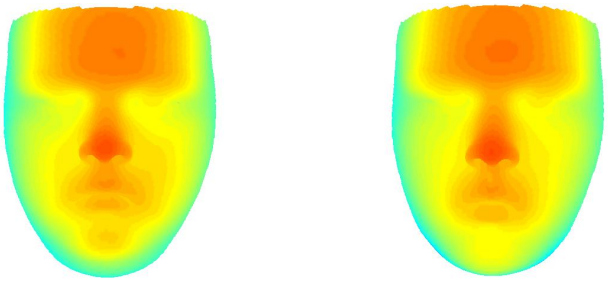
Jak vidíme na výše uvedených obrázcích (5.14, 5.15), ač má model možnost svoje

měřítko měnit, bohužel optimalizace vydává do lokálního minima, a snaží se chybu odstranit extrémním protažením ořízntého obličeje a "napasovat" tedy bradu na krk předlohy. Toto testování nám dokázalo, že se bez ořezu obličeje neobejdeme.

5.5 Finální testování na vygenerovaných datech

Poslední testování před testováním na reálných datech mělo ověřit jak všechny výše uvedené možnosti, které optimalizační algoritmus má, fungují dohromady. Opětovně byla vygenerována náhodným natočením, náhodným posunutím a náhodně změněným měřítkem. Optimalizace byla rozdělena na čtyři části, přičemž v první a třetí byl prohledáván polohový vektor, zatímco ve druhé a čtvrté vektor parametrů BFM. Na každou fázi bylo využito 250 iterací, celkový počet iterací byl tedy standardně tisíc.

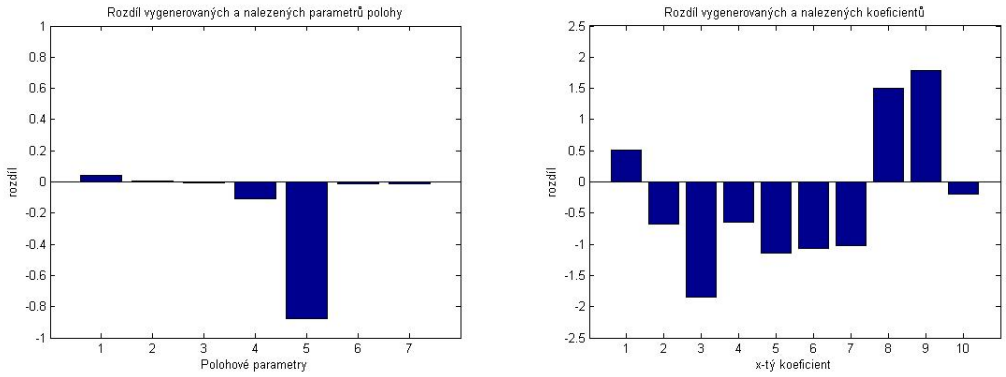
Optimalizace startovala s nulovým vektorem jak polohy, tak parametrů modelu. V druhé fázi optimalizace byla samozřejmě brána v potaz poloha nalezená v části první. Ve třetí fázi byl opětovně prohledáván prostor řešení vektoru polohy s tím, že startovní bod byla nalezená poloha z fáze jedna a omezující podmínky se taktéž odvíjely od této polohy. Bylo samozřejmě manipulováno s instancí, která byla označena za výslednou na konci fáze 2. Ve fázi čtyři byla opětovně hledána nejlepší instance BFM, tentokrát se zafixovanou polohou získanou ve třetí fázi optimalizace. Jako startovní bod byl zvolen vektor parametrů získaný ve druhé fázi a omezující podmínky se opět odvíjely od něj. Zde přikládám výsledky, tentokrát pro větší názornost v barevných odstínech:



Obrázek 5.16: Vygenerovaná tvář + nalezená instance

Obrázek 5.17: Vzniklá rezidua

V tomto konkrétním případě byla výsledná chyba na pixel rovna 0.77%, přičemž u ostatních podobných testování se chyba pohybovala mezi hodnotami 0.1% a 0.9%. Jak vidíme, obě tváře si jsou velmi podobná a vzniklá rezidua jsou poměrně zanedbatelná.



Obrázek 5.18: Grafy rozdílu koeficientů (poloha, BFM)

Na grafech 5.18 je vidět rozdíl mezi nalezenými koeficienty jak polohy (v pořadí: rotace kolem osy X, rotace kolem osy Y, rotace kolem osy Z, posunutí podél osy X, posunutí podél osy Y, posunutí podél osy Z, změna měřítka), tak BFM. Tento rozdíl je u polohy prakticky nulový, u parametrů BFM jsou vidět určité nepřesnosti, avšak to je způsobeno tím, že každý komponent ovlivňuje více vlastností obličeje a proto velmi podobné tváře můžeme získat v určitém okolí cílové instance.

5.6 Testování na reálných datech

Nyní, když jsme si ověřili funkčnost na cvičných datech, nám nic nebránilo otestovat metodu přizpůsobování modelu na reálných 3D datech. Testování ukázala, že dosažené výsledky jsou v průměru o 2% na pixel lepší než tomu bylo před implementací volnosti pohybu modelu. Průměrná chyba se tedy pohybovala okolo 3.5% na pixel, což vzhledem k tomu, že byly používány i snímky s různým natočením tváře je velký pokrok.

Výsledky však zdaleka stále nedosahují kvality výsledků z pokusů na cvičných datech. Tento jev bychom mohli přisoudit nejen rozdílnosti skenů a BFM, ale také nízké kvalitě snímků. Na snímcích se často objevují díry, ke kterým je SA sice poměrně invariantní, nicméně přeci jen se do celkové chyby promítnou.

Je třeba zmínit, že u minoritního podílu testování bohužel chyba snížena nebyla, jelikož optimalizační funkce na počátku prohledávání špatně posoudila polohu tváře subjektu na snímku, dostala se do globálního minima a už se z tohoto bodu leckdy nedokázala vrátit ke správnému řešení. Na druhou stranu je ale nutno podotknout, že u snímků, u kterých by bylo dříve řešení kvůli natočení tváře nenalezitelné, byla často poloha optimalizačním algoritmem velmi přesně sledována.

Zde uvádím výsledky jednoho z testů, viz obr. 5.19 a obr. 5.20, kdy algoritmus úplně přesně nedokázal nalézt správnou polohu a přesto chybu na pixel snížil na 3%. Lze si na něm povšimnout zbývajících nedostatků naší metody, které přímo vyplývají z toho, že snímky z Kinectu obsahují vlasy. Optimalizační algoritmus se pak často, aby snížil chybu, snaží BFM nepřirozeně protáhnout.

Ve finále se tedy stal optimalizační algoritmus mnohem robustnější a ač chyba, se kterou nalézal řešení, nebyla ideální, zaznamenal v této oblasti razantní zlepšení (zhruba o 40%). I přes veškeré tyto nedostatky, kterými optimalizační metoda stále trpěla jsme se rozhodl pokusit se pořízená data experimentálně klasifikovat a tak posoudit, zda jsou

Obrázek 5.19: Výsledky testování na reálných datech (sken, nalezená instance)

Obrázek 5.20: Výsledky testování na reálných datech (rezidua)

již výsledky natolik přesné, nebo alespoň si navzájem natolik podobné pro stejnou osobu. Problémy klasifikace se zabývají následující kapitoly.

Kapitola 6

Klasifikace a klasifikátory

Rozpoznávání či klasifikace je přirozená pro každého člověka [17]. Člověk rozpoznává předměty ve svém okolí a jedná v relacích s nimi. I v oblasti rozpoznávání můžeme pozorovat tendenci o vytvoření stroje, který tuto činnost provádí automaticky bez účasti člověka. Takovému stroji se říká klasifikátor. Klasifikace tedy spočívá v zařazování předmětů do jednotlivých tříd. Každý předmět musí být nejprve vhodně popsán, přičemž tento popis může být jak kvantitativní tak i kvalitativní. V případě, že se jedná o dostatečně kvalitní popis, nazýváme ho obraz a dále mluvíme o tzv. rozpoznávání obrazů (pattern recognition). Při návrhu automatického rozpoznávání obrazů musíme řešit dva základní problémy:

1. Problém vhodného popisu předmětu: Předmět, který chceme rozpoznat, musí být vhodně popsán. Ideální popis je tvořen pomocí tzv. příznaků daného předmětu - elementárních vlastností, které mají číselný charakter. Vektor těchto příznaků se pak nazývá obraz předmětu.

2. Problém rozpoznávání nebo také problém návrhu klasifikátoru: Problém nalezením dělících hranic mezi jednotlivými třídami, do kterých má být klasifikovaný obraz zařazen. Rozdělující hranice jsou definovány tzv. diskriminačními (rozhodovacími) funkcemi, podle jejichž hodnot provede klasifikátor automatické přiřazení obrazu do příslušné skupiny.

6.1 Problém vhodného popisu předmětu

Tímto problémem jsme se naštěstí nemuseli příliš zabývat, jelikož jeho řešení se nám přímo nabízelo v podobě jednotlivých parametrů BFM. Tyto parametry splňovaly jak předpoklad, že se jedná o elementární vlastnosti, tak i předpoklad číselného charakteru. Je vhodné zmínit, že vektor polohy nebyl během klasifikace ze zřejmých důvodů nijak uvažován (tvář může být jakkoliv natočena, nezávisle na konkrétní osobě). Klasifikace námi nalezených řešení se tedy odehrávala nad desetisložkovým vektorem příznaků BFM(optimalizace prohledávala pouze prvních deset příznaků BFM).

6.2 Rozhodovací funkce

Pro některé typy úloh nelze jednoznačně posoudit, do které třídy příznaků posuzovaný obraz patří. Proto existuje několik přístupů (typů rozhodovacích funkcí), jak v těchto případech nalézt správné řešení. Zde uvádím několik nejdůležitějších:

6.2.1 Pravděpodobnostní diskriminační funkce

Pravděpodobnostní diskriminační funkce využívá pro správnou klasifikaci obrazu x Bayesovo kritérium v následujícím vztahu:

$$g'_r(x) = p(x|\omega_r) \cdot P(\omega_r);\ r = 1, 2, ..., R \tag{6.1}$$

kde $p(x|\omega_r)$ je podmíněná hustota pravděpodobnosti obrazu x ze třídy ω_r a $P(\omega_r)$ je apriorní pravděpodobnost výskytu obrazů náležících třídě ω_r.

Klasifikátor pracující podle tohoto vztahu se nazývá Bayesův klasifikátor a pro jeho konstrukci je třeba znát apriorní pravděpodobnost pro každou třídu, bohužel tuto znalost prakticky nikdy nemáme. Rozhodnutí, do které třídy je pak neznámý obraz x zařazen, se provede výběrem maximální hodnoty napříč Bayesovi kritérii jednotlivých tříd.

6.2.2 Lineární diskriminační funkce

Tato funkce by se měla využívat, pokud všechny třídy vykazují stejnou kovarianční matici a zároveň podléhají normální rozložení. Vzhledem k velmi výhodným analytickým vlastnostem těchto funkcí se však využívají i v případech, kdy výše uvedené podmínky splněny nejsou. Přesto v případech, kdy jsou obrazy jednotlivých tříd dobře distribuované neboli vytvářejí kompaktní shluky, které jsou lineárně separabilní, toto zjednodušení vyhovuje dostatečně. Při klasifikaci do dvou tříd (dichotomii) se využívá následující rozhodovací funkce:

$$g(x) = q_0 + \sum_{i=1}^{n} q_i x_i \tag{6.2}$$

kde q_i je váha funkce pro jednotlivé složky vektoru x a q_o je práh funkce. Pro $g(x) > 0$ bude patřit x do první třídy, pro $g(x) < 0$ bude patřit do třídy druhé.

6.2.3 Klasifikace podle minimální vzdálenosti

Funkce minimální vzdálenosti se využívá v případě, že příznaky obrazů podléhají normálnímu rozložení, a apriorní pravděpodobnosti všech tříd jsou stejné. Před klasifikací je nutné určit střední hodnotu (střed shluku) každé třídy. Samotná funkce je definována následujícím vztahem:

$$g_r(x) = \|x - \mu_r\|^2 \tag{6.3}$$

kde $\|x - \mu_r\|^2$ je kvadrát Euklidovské vzdálenosti mezi neznámými x a střední hodnotou r-té třídy (typickým představitelem r-té třídy). Klasifikátor zařadí neznámý obraz x do té třídy, pro kterou bude $g_r(x)$ nabývat minimální hodnoty. Dobrých výsledků dosáhneme, když budou střední hodnoty jednotlivých tříd dostatečně vzdálené a shluky

dostatečně kompaktní. Klasifikátor podle minimální vzdálenosti je velmi často nasazován i v případech, kdy není jistá podmínka stejných apriorních pravděpodobností jednotlivých tříd kvůli jeho jednoduchosti a malé náročnosti na paměť.

6.2.4 Klasifikace podle nejbližšího a k-nejbližšího souseda

Klasifikace podle nejbližšího souseda probíhá podle pravidla vyjádřeného vztahem:

$$\omega_r = argmin\|x - \mu_{rs}\| = argmin[d(x - \mu_{rs})] \tag{6.4}$$

kde μ_{rs} je s-tý zástupce ze vzorových zástupců třídy r. Obraz x je tedy zařazen do té třídy, jejíž některý vzorový zástupce má od obrazu x nejmenší vzdálenost. V případě k-nejbližších sousedů je pro každou třídu vždy bráno k zástupců a jejich vzdálenosti sečteny. Obraz x je pak zařazen do třídy, jejíž suma těchto vzdáleností je nejnižší. Klasifikátor využívající tyto funkce má tu výhodu, že při dostatečně rozsáhlé trénovací množině se tvar dělících funkcí pro jednotlivé třídy "blíží" Bayesovskému klasifikátoru. Naproti tomu se prudce zvyšují nároky na paměť, jelikož je nutné si celou trénovací množinu pamatovat.

Kapitola 7

Vlastní postup klasifikace získaných dat

V této kapitole je popsán mnou vybraný postup klasifikace zpracovaných 3D dat i získané výsledky. Nad vybranými klasifikátory jsem provedl testování jak s umělými tak s reálnými daty. Jak již bylo zmíněno v kapitole 2.2.2 bylo pořízeno patnáct až dvacet snímků 9 rozličných osob. Tyto snímky byly vyselektovány a bylo ponecháno deset až patnáct snímků na osobu. Během trénování klasifikátorů bylo vždy prvních sedm snímků použito jako vzor a zbylé snímky klasifikovány. Tento postup se však aplikoval pouze na prvních osm osob, devátá osoba se ponechala stranou a byla předložena klasifikátoru aniž by ji kdykoliv předtím viděl. Testovali jsme celkem tři typy klasifikátorů.

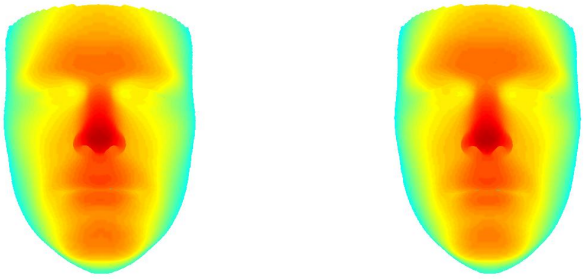
7.1 Klasifikátor podle minimální vzdálenosti

Jako první byl otestován klasifikátor pracující podle standardního kritéria minimální vzdálenosti, viz kap. 6.2.3. Klasifikátor byl vybrán nejen pro jeho jednoduchou implementaci (námi vytvořený skript v MATLABu *Klasifikator*), ale také kvůli předpokladu, že finální instance BFM budou mít podobné parametry pro stejné vzorové osoby. Zároveň jsme doufali, že střední hodnoty tříd budou pro rozličné osoby od sebe dostatečně vzdálené. Tato vzdálenost by měla být tím větší, čím méně si jsou osoby podobné.

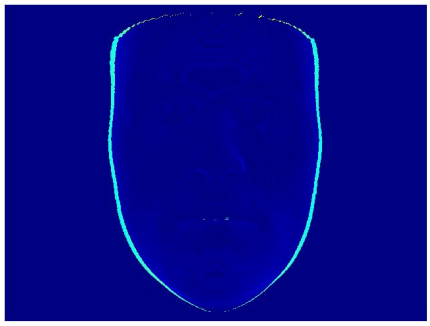
7.1.1 Testování na umělých datech

Umělá data byla vytvořena tak, že bylo nejprve vygenerováno 8 odlišných tváří. Pro každou tuto tvář bylo dále vygenerováno devět velmi podobných (toho jsme docílili tak, že jsme lehce zašuměli BFM parametry původní tváře). Tváře byly samozřejmě vygenerovány s náhodným natočením a posunem.

V dalším kroku jsme, stejně jako v kap. 5.5, umělá data použili jako vzor pro námi vyvinutou metodu přizpůsobení modelu 3D datům. Tyto výsledky byly uloženy a prvních sedm u každé osoby bylo použito k vypočtení střední hodnoty dané třídy. Ač se parametry výsledných instancí leckdy lehce lišily, střední hodnoty jednotlivých tříd byly velmi podobné původním datům, viz obr. 7.1. Daný obrázek je opět pro názornost vykreslen v barevných odstínech a bez původního posunu, jelikož střed shluků byl počítán se základní polohou. Jak vidíme na obr 7.2, rezidua jsou minimální.

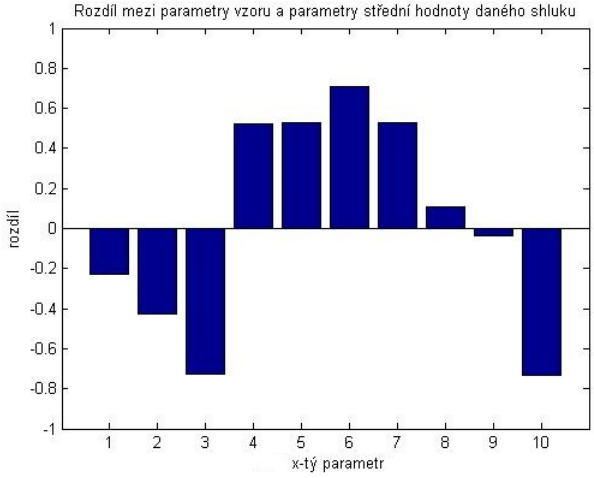


Obrázek 7.1: Srovnání vzoru a střední hodnoty dané třídy



Obrázek 7.2: Vzniklá rezidua

Dále přikládám graf rozdílu mezi koeficienty vzoru a středu shluku, viz obr. 7.3. V neposlední řadě je třeba zmínit, že chyba na pixel se pohybovala v průměru okolo 0.8% (průměrná chyba mezi vzory a středy shluků).



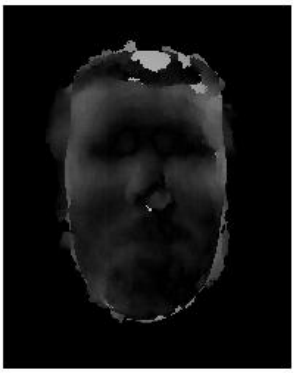
Obrázek 7.3: Rozdíl mezi nalezenými a původními parametry modelu

Klasifikátoru byla poté dodána zbylá data k posouzení, který zařadil posuzovaný vektor příznaků do správné třídy s pravděpodobností **100%**. Tento výsledek byl velmi povzbudivý a proto jsme ihned přistoupili k experimentální klasifikaci reálných dat.

7.1.2 Testování na reálných datech

Jak již bylo v této kapitole řečeno, nejprve byly optimalizační metodou nalezeny finální instance BFM pro každý snímek, poté bylo prvních sedm nalezených instancí od každé osoby použito k výpočtu středů shluků. Zde přikládám opět porovnání střední hodnoty třídy a jednoho ze skenů dané osoby:

Obrázek 7.4: Srovnání vzorového skenu a střední hodnoty dané třídy



Obrázek 7.5: Vzniklá rezidua

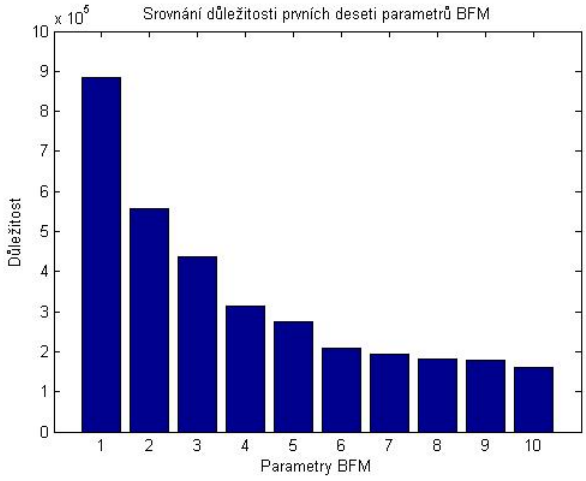
Jak vidíme, rezidua jsou v tomto případě větší než u umělých dat. Velká chyba se objevuje v místech vlasů a nově také vousů (dotyčná osoba na skenu měla během snímání plnovous). Model se tvář opětovně snažil nesmyslně protáhnout v důsledku přítomnosti vlasů na skenu. Průměrná chyba mezi střední hodnotou dané třídy a skeny se pohybovala kolem 4% na pixel. Tato hodnota je vyšší než u testování v kapitole 5.6, důvodem je úmyslné zařazení méně kvalitních snímků, tak aby byla simulována situace z reálného prostředí, kde osoba, která chce být rozpoznána nebude dodržovat perfektní vzdálenost od čidla, či nebude držet hlavu vždy v perfektní rovině, viz obr. 7.6.

Obrázek 7.6: Snímek s úmyslným náklonem hlavy a malými dírami

Klasifikátor byl otestován nejprve na datech, která mu nikdy předtím nebyla poskytnuta. Bohužel výsledkem byla úspěšnost pouze **22.5%**. Proto jsem se pokusil klasifikovat i data, která byla využita k hledání středních hodnot tříd. Úspěšnost rozpoznání těchto dat byla přibližně 64%. Z tohoto faktu lze vyvodit, že náš předpoklad, že klasifikovaná data budou tvořit kompaktní shluky, je pravděpodobně mylný a tudíž námi použitý klasifikátor pro ně není vhodný.

7.2 Modifikovaný klasifikátor podle minimální vzdálenosti

V této kapitole je popsána modifikace klasifikátoru podle minimální vzdálenost z kapitoly předchozí. Je třeba si uvědomit, že jednotlivé parametry BFM nejsou stejně důležité, viz obr. 7.7 a toho jsem se rozhodl během modifikace využít.



Obrázek 7.7: Váha prvních deseti parametrů BFM

Původní klasifikátor byl tedy poměrně jednoduše modifikován a to tak, že během výpočtu vzdálenosti byla vzdálenost v daném rozměru vždy ještě vynásobena příslušnou váhou parametru.

$$g_r(x) = \sum_{i=1}^{10}(\omega_i \cdot [x_i - \mu_{r,i}]^2) \tag{7.1}$$

Klasifikovaný obraz byl opět zařazen do té třídy, pro níž vyšla vzdálenost minimální.

7.2.1 Testování na umělých datech

Testování na umělých datech probíhalo prakticky identicky jako u předchozího typu klasifikátoru. Tentokrát však již nebyla data znovu zpracovávána, ale byly využity výsledky z předchozí optimalizace. Úspěšnost klasifikátoru byla zachována a to **100%**.

7.2.2 Testování na reálných datech

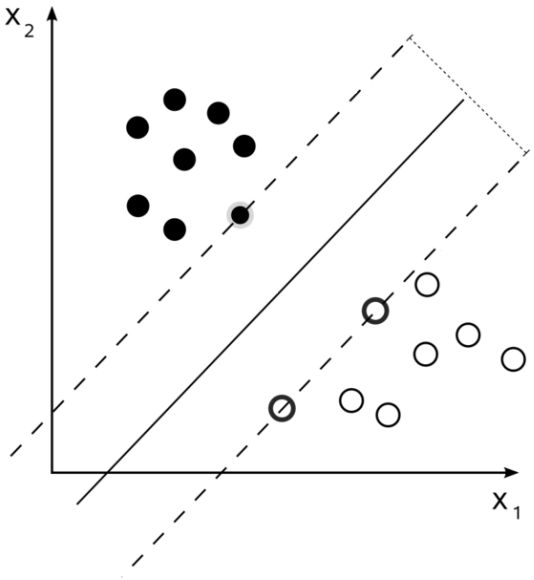
Testování probíhalo tentokrát bez první fáze, tedy bez hledání správné projekce modelu (byly využity výsledky z předchozího pokusu). Střední hodnoty tříd byly opět spočteny z prvních sedmi instancí u každé osoby a zbylé byly použity jako testovací množina.

Úspěšnost tohoto klasifikátoru byla **32.5%**, zaznamenali jsme tedy zlepšení zhruba o 10%. Dále byla předložena klasifikátoru data, která již použil k nalezení středů shluků. Tato data správně zařadil v 67% případů. Jak vidíme, modifikace klasifikátoru s sebou přinesla zlepšení, nicméně finální úspěšnost je stále poměrně nízká. To, že klasifikátor zařadil pouze dvě třetiny dat využitých na trénování správně, nám ukázalo, že data opravdu netvoří kompaktní shluky a proto pro ně není klasifikátor založený na minimální vzdálenosti vhodný.

7.3 SVM klasifikátor

SVM (Support vector machine) klasifikátor patří do třídy binárních (zařazuje obrazy pouze do dvou tříd) klasifikátorů [7][8]. Trénování klasifikátoru probíhá ve formě učení s učitelem. To znamená, že data, která mu předložíme musí v sobě nést i údaj, zda patří do první či druhé třídy. Z těchto údajů klasifikátor poté vytvoří model, podle kterého hodnotí všechny další obrazy. SVM model je znázornění cvičných dat v prostoru, mapovaných tak, že příklady odlišných tříd jsou od sebe odděleny nadrovinou tak širokou, jak jen to je možné. Nové obrazy jsou pak mapovány do stejného prostoru a následně zařazeny do třídy podle toho, na jaké straně nadroviny skončí, viz obr. 7.8. Existují jak lineární, tak nelineární typy klasifikátoru a odlišují se od sebe právě tím, jakým typem nadroviny od sebe oddělují jednotlivé třídy.

Nespornou výhodou tohoto klasifikátoru bylo, že již byl v MATLABu implementován, čehož jsme také využili (funkce *svmtrain* a *svmclassify*). Bohužel SMV klasifikátor je klasifikátor binární, zatímco my jsme měli tříd osm. Proto bylo třeba zvolit vhodnou klasifikační strategii. Jako nejrozumnější a nejsnáze implementovatelná byla shledána strategie *vítěz bere vše* (winner-takes-all strategy), která patří mezi strategie typu *jeden proti všem* (one-versus-all). Tato strategie funguje tak, že každá třída má vlastní klasifikátor. Ten je natrénován s tím, že trénovací obrazy jsou mu dodávány



Obrázek 7.8: Lineární dělicí nadrovina SVM klasifikátoru

pouze s údajem, zda do dané třídy patří či nepatří (odtud one-versus-all). Klasifikovaný obraz je pak posouzen všemi klasifikátory zvlášť a je zařazen do té třídy, pro níž je vzdálenost od hraniční nadroviny největší (ve směru do středu dané třídy).

Bohužel ani jedna z výše zmíněných funkcí poskytnutých MATLABem nám tento údaj neposkytla, proto bylo třeba najít vhodnou alternativu. Naštěstí jsem mezi nezdokumentovanými funkce MATLABu nalezl funkci *svmdecision*, jejíž výstupem byla hledaná vzdálenost.

Teď již zbývalo pouze zvolit vhodný typ dělicích nadrovin. Celkem byly otestovány typy tři. Testování proběhlo rovnou na reálných datech, jelikož po předešlé zkušenosti s umělými daty nemělo další testování žádný přínos. Data opět nebyla znovu zpracovávána, ale byly použity výsledné instance získané před testováním klasifikátoru podle minimální vzdálenosti.

7.3.1 Lineární

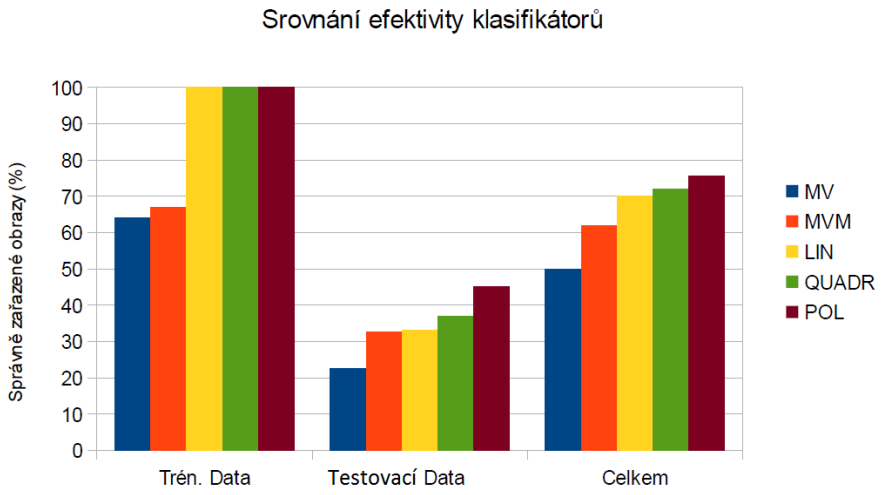
Nejdříve bylo testováno lineární dělení jednotlivých tříd. Klasifikátor rozpoznal data, která využil k trénování, s úspěšností 100%, zatímco nová data, která k trénování využita nebyla, správně přiřadil ve **33%** případů. Jak vidíme, úspěšnost rozpoznání trénovacích dat prudce vzrostla, bohužel u ostatních dat zůstala přibližně stejná.

7.3.2 Kvadratické

Kvadratické dělení ponechalo úspěšnost rozpoznání trénovacích dat na hodnotě 100% a zároveň šance na správné přiřazení ostatních dat vzrostla přibližně o 4%, tedy na hodnotu **37%**.

7.3.3 Polynomické

Polynomické dělící nadroviny se ukázaly býti jako zdaleka nejefektivnější. Trénovací data byla zařazena do správné třídy vždy a obrazy, které byly klasifikátoru předloženy poprvé, byly správně zařazeny ve **45%** případů. To je nárůst na dvojnásobek oproti testům s klasifikátorem podle minimální vzdálenosti. Vzhledem k poměrně malé trénovací množině můžeme předpokládat, že s jejím zvětšením, by došlo i ke zvýšení efektivity klasifikátoru.



Obrázek 7.9: Srovnání efektivity jednotlivých klasifikátorů

7.3.4 Klasifikace neznámé osoby

Vzhledem k efektivitě prvních dvou testovaných klasifikátorů by tento experiment postrádal smysl, nicméně SVM klasifikátor se ukázal býti jako poměrně úspěšný a proto jsme mu zkusili předložit data deváté osoby, která nebyla do trénování klasifikátoru nijak zapojena.

Klasifikace neznámé osoby byla nejprve testována na umělých datech. Klasifikátor poznal se 100% úspěšností, že daný obraz nepatří do žádné třídy. Během testování na reálných datech bylo klasifikátoru předloženo celkem 12 zpracovaných snímků deváté osoby. U čtyř z nich rozpoznal, že se nejedná o žádnou z předchozích osob o nezařadil ji do žádná třídy. Zbylých osm snímků klasifikoval jako členy některé z osmi tříd.

7.4 Experiment s referenčními body

Ač bylo při změně klasifikátoru zaznamenáno signifikantní zlepšení, stále jsem s výsledky nebyl spokojen. Proto jsem se pokusil zlepšit optimalizační metodu, ve které jsem kromě nízké kvality snímků viděl největší nedostatky. Po vzoru metody Blanze a Vettera, viz kap. 3.2, jsem se rozhodl nanést na každý sken několik referenčních bodů a tím zlepšit následné přizpůsobení modelu. Původně bylo zamýšleno nanášet pět referenčních bodů na každý snímek a to na špičku nosu, koutky očí a koutky úst. Bohužel kvůli nízké kvalitě

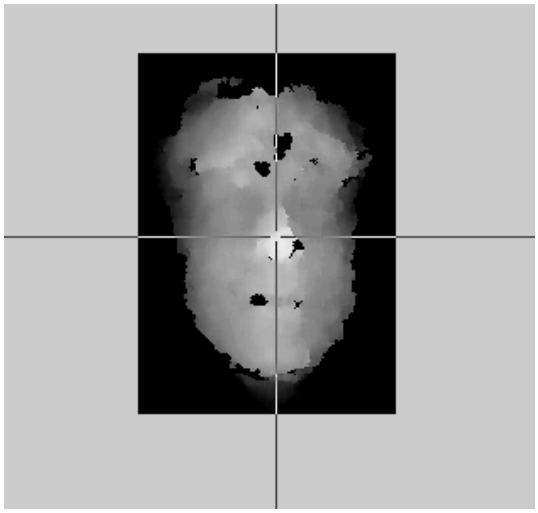
snímků bylo často nemožné označit správnou polohu úst a proto byl počet zredukován na tři.

Nanášení probíhalo u nových snímků rovnou během jejich ořezu, u starých bylo provedeno dodatečně. Celá myšlenka referenčních bodů fungovala tak, že jsme získali souřadnice tří výše zmíněných míst a kromě klasické chyby jsme v každé iteraci optimalizace chybu navýšili o rozdíl z-tových souřadnic v těchto místech vynásobený určitým váhovým koeficientem. K získání souřadnic referenčních bodů jsme použili MATLABovskou funkci *ginput*, jejíž výstupem je matice s x-ovými a y-ovými souřadnicemi nakliknutých bodů, viz obr. 7.10.

Váhový koeficient byl experimentálně určen a to tak, že bod špičky nosu měl váhu ω_1 jako 25% všech bodů snímku a každý koutek očí měl váhu ω_2 jako 10% všech bodů snímku. Referenční body byly uvažovány jak při hledání správných parametrů BFM, tak při prohledávání prostoru polohy modelu. Tato úprava by nám měla zkvalitnit nalezené výsledky a zároveň prakticky nezvýšit výpočetní náročnost. Jedinou nevýhodou je nutnost delší předpřípravy jednotlivých skenů. Zde uvádím rovnici výpočtu chyby:

$$\Delta_{abs} = \sum |s_{i,j} - m_{i,j}| + \omega_1 \cdot |s_{r,s} - m_{r,s}| + \omega_2 \cdot |s_{u,v} - m_{u,v}| + \omega_2 \cdot |s_{w,z} - m_{w,z}| \quad (7.2)$$

kde *r,s* jsou souřadnice špičky nosu, *u,v* jsou souřadnice pravého očního koutku a *w,z* souřadnice levého očního koutku.



Obrázek 7.10: Funkce ginput

Testování bohužel přineslo rozporuplné výsledky. U některých skenů byla chyba na pixel snížena, u některých vyšla zhruba stejná a u některých se naopak chyba zvětšila a to pravděpodobně u těch snímků, kde se přesně nepodařilo označit koutek oka. Celkově se průměrná chyba na pixel snížila zhruba jen o 0.2%, což vzhledem ke zvýšeným nárokům na manuální zásah není obhajitelný přínos.

Kapitola 8

Zhodnocení výsledků

8.1 Výsledky pořízení kolekce dat

V této oblasti má práce největší nedostatky. Velkou předností snímků z Kinectu je jejich lehká a rychlá pořiditelnost. Další nespornou výhodou je jejich snadná zpracovatelnost a poměrně nízké nároky na paměť. Bohužel tím výčet výhod končí.

Verze Kinectu, kterou jsme použili, nedokáže zachytit předměty na menší vzdálenost než 70 cm. S přihlédnutí na celkové rozlišení snímku 640x480 nám kvůli tomu bohužel na obličej skenovaného subjektu připadá pouze zhruba 200x200 bodů. Toto rozlišení není ideální a vzhledem k tomu, že námi vybraný model byl původně koncipován na rozlišení právě 640x480, způsobuje to během hledání vhodné reprezentace časté nepřesnosti. Tento problém by se bohužel nedal vyřešit jinak než výběrem jiného skeneru, například nové verze Kinectu, kterou se nám pro tuto práci ale nepodařilo získat.

Dalším problémem, který s sebou snímky z Kinectu přináší jsou občasné díry. Tento problém však není nikterak velký, jelikož SA je k dírám na snímku poměrně invariantní. Jediná chvíle, kdy tento nedostatek nabyl na váze, byla během pokusu s referenčními body. Když byla díra na skenu přímo v místě referenčního bodu, vnášelo to do optimalizace neúnosnou chybu. Pokud bychom tuto metodu chtěli dále používat, bylo by nutné aplikovat na jednotlivé snímky například mediánový filtr.

Obrázek 8.1: Ilustrační příklad zachycených dlouhých vlasů

Poslední, ale neméně důležitou nevýhodou snímků z Kinectu byla prakticky nepřítomnost krku a uší, zatímco model tyto části obličeje měl. To jsme naštěstí vyřešili pomocí ořezu dat v hloubce. Co však vyřešeno nebylo, je přítomnost vlasů a vousů na skenech z Kinectu, zatímco BFM je vymodelovat nedokáže. Tento rozdíl vnášel do zpracování obrovskou chybu, přičemž kromě manuálního odstranění, neexistovala žádná rozumná možnost, jak se vlasů a vousů na snímku zbavit. Největší problém nastával tehdy, pokud dotyčná osoba měla dlouhé rozpuštěné vlasy, viz obr 8.1.

Celkově se tedy zařízení Kinect neprokázalo jako nejrozumnější volba a pro další testování by bylo přínosem pokusit se kolekci dat pořídit například jeho novou verzí, která dokáže snímat objektu už na vzdálenost 30 cm bez ztráty přesnosti.

8.2 Výsledky optimalizace

Samotná optimalizace prošla během této práce velkými změnami. Vybraný optimalizační algoritmus - simulované žíhání se ukázal býti dostatečně robustním jak pro umělá, tak pro reálná data. V této oblasti nemá algoritmus větší nedostatky. Do budoucna by se dalo uvažovat o hledání hrubého řešení právě pomocí SA, přičemž k prohledávání jeho okolí by mohl posloužit nějaký jiný, více metodický algoritmus.

Nový způsob ořezu pracoval také naprosto spolehlivě a s přihlédnutím na poté přidanou možnost pohybovat s instancemi modelu se tato inovace ukázala i jako naprosto nezbytná.

Největším vylepšením v oblasti optimalizace byla zajisté implementace hledání správné polohy modelu, která prokázala dostatečnou efektivitu ve všech testováních. SA algoritmus se ukázal vhodný pro řešení i tohoto problému a hledal správné výsledky s velkou přesností. Nejvhodnější alternativou během hledání správného řešení se ukázalo býti rozdělení prohledávání do čtyř částí, kde v první a třetí je hledána správná poloha modelu, zatímco ve druhé a čtvrté je prohledáván prostor parametrů BFM.

Experiment s referenčními body bohužel nepřinesl žádné větší zlepšení, nicméně to je způsobeno mimo jiné i nižší kvalitou skenů. Pokud by se nám tuto kvalitu podařilo zvýšit (například při použití jiného skeneru), referenční body by pravděpodobně více přispívaly k nalezení ideální instance BFM.

Celkově bych optimalizační část hodnotil veskrze pozitivně. Drobná vylepšení by mohl zaznamenat v budoucnu například samotný optimalizační algoritmus, ale zároveň si myslím, že tyto nedostatky jsou minoritní záležitostí.

8.3 Výsledky klasifikace

Pokusy s prvními dvěma typy klasifikátorů nám ukázaly, že naše teorie, že výsledná data budou tvořit pro stejné osoby kompaktní shluky, je mylná. Nicméně modifikovaný klasifikátor podle minimální vzdálenosti neposkytoval úplně chybné výsledky, proto se dá očekávat, že se vzrůstající kvalitou skenů a tudíž i vzrůstající kvalitou jejich zpracování, by mohl být poměrně účinný.

SVM klasifikátor dokázal zhruba s poloviční úspěšností rozpoznat neznámá data a zařadit je do správné třídy. Data použitá pro jeho natrénování rozpoznal se stoprocentní

úspěšností z čehož lze vyvodit, že se vzrůstající velikostí trénovací množiny by vzrostla i úspěšnost klasifikace neznámých dat.

Vybraný klasifikátor tedy přesně splňuje naše požadavky a v budoucnu bych pro zvýšení úspěšnosti klasifikace navrhoval zvětšení trénovací množiny.

Kapitola 9

Závěr

Tato diplomová práce popisuje nejčastější metody pořizování obrazových 3D dat lidského obličeje. Na základě těchto poznatků byl vybrán vhodný přístroj, konkrétně Kinect Xbox360, pomocí kterého byla pořízena má vlastní kolekce dat. Bylo oskenováno celkem devět osob, přičemž se jednalo o pět mužů a čtyři ženy. Všichni snímaní lidé byli běloši v letech mezi 17 a 55 lety. Pro každou osobu bylo pořízeno celkem 15 až 20 snímků, které byly následně vyselektovány a bylo ponecháno 10 až 15 snímků pro každou osobu.

Dále jsou v práci uvedeny vybrané metody přizpůsobování 3D modelu podle trojrozměrných dat, ze kterých bylo později čerpáno při návrhu vlastního postupu pro vybraný animační model. Tímto modelem byl zvolen model vyvinutý Univerzitou v Baselu, který poskytoval nejen celou škálu možností při práci s modelem, ale byl zároveň lehce modifikovatelný.

Úkolem práce tedy bylo vytvořit algoritmus, který by pro snímky z Kinectu dokázal najít optimální instanci Baselského modelu. Snímky bylo nejprve nutné oříznout, jelikož minimální snímací vzdálenost vybraného zařízení činila 70 cm, a tak bylo na skenech zachyceno kromě lidské tváře i mnoho nežádoucích objektů. Dalším krokem bylo nalezení vhodné reprezentace, ve které by se instance modelu a skeny daly srovnat. Ta byla získána pomocí funkce *colormap*, která byla aplikována na model a díky tomu získána jeho hloubková mapa. Dále bylo nutné vyřešit vzájemnou rozdílnost skenů a modelu, jelikož sken obsahoval vlasy a vousy, zatímco model nikoliv, a naproti tomu model obsahoval krk a uši, zatímco ty na snímcích zachyceny nebyly. Problém se podařilo vyřešit alespoň částečně hloubkovým ořezem modelu, díky kterému z něj byly uši a krk odstraněny. Bohužel vlasy a vousy by bylo nutné odstranit na každém skenu manuálně, od čehož bylo upuštěno.

Po překlenutí hlavních problémů bylo otestováno několik optimalizačních algoritmů, přičemž jako nejvhodnější se ukázal být algoritmus simulovaného žíhání. Bohužel bylo třeba dále řešit fakt, že skeny nejsou perfektně vycentrované a subjekt na nich není nikdy zachycen v perfektní rovině. Proto byla dodána modelu volnost jak v posunu, tak natočení. Následně byla taktéž přidána možnost drobné změny měřítka modelu kvůli skutečnosti, že po rozdílných ořezech stejný obličej nezabíral stejnou plochu. Během optimalizace pak byly prohledávány parametry modelu, ale také tyto polohové parametry. Nejvíce se osvědčilo tato prohledávání od sebe oddělit (vzhledem k výpočetní náročnosti) a hledat řešení celkem ve čtyřech oddělených částech. V první a třetí části byla hledána správná poloha modelu, zatímco ve druhé a čtvrté parametry BFM.

V poslední části práce jsem se zabýval klasifikací zpracovaných dat. Je zde mimo jiné uvedeno několik základních typů klasifikátorů, přičemž klasifikátory, které byly použity během testování jsou rozebrány detailněji. Pro klasifikaci našich dat byly použity celkem tři typy klasifikátorů - klasifikátor podle minimální vzdálenosti, modifikovaný klasifikátor podle minimální vzdálenosti a SVM klasifikátor. Nejlepších výsledků dosahoval právě poslední jmenovaný, který neznámá data správně přiřadil zhruba v polovině případů.

Výsledky práce jsou poměrně uspokojivé, nicméně kdybychom námi vyvinutý algoritmus chtěli využít pro rozpoznávání osob, bylo by třeba buď zvolit pro získávání dat jiný skener, a nebo nalézt metodu, které automaticky odstraní u každého snímaného subjektu vlasy a vousy.

Literatura

[1] The Joy of Visual Perception. 2002.
URL `http://www.yorku.ca/eye/toc.htm`

[2] Xbox Kinect 360. 2007.
URL `http://www.xbox.com`

[3] Basel Face Model. date.
URL `http://faces.cs.unibas.ch/bfm/main.php`

[4] ARELLANO, C.; DAHYOT, R.: Shape model fitting algorithm without point correspondence. 2012.

[5] BLANZ, V.; VETTER, T.: A Morphable Model for the synthesis of 3D Face. 2005.

[6] BLANZ, V.; VETTER, T.: Face Recognition based on fitting a 3D Morphable Model. Srpen 2003.

[7] BURGES, C.: A Tutorial on Support Vector Machines for Pattern Recognition. 1998.

[8] FAIG, J.; SVOBODA, L.; ŽOLDÁK, M.: Support Vector Machine. 2001.

[9] GILL, P.; WONG, E.: *Sequential quadratic programming methods*. 2007.

[10] GRUBER, I.: Rekonstrukce povrchu lidské tváře za pomoci 3D modelu. 2011.

[11] HAAR, F.; VELTKAMP, R.: 3D Model Fitting for recognition. 2008.

[12] HARLEY, R.; ZISSERMAN, A.: *Multiple View Geometry in computer vision*. 2010.

[13] LANGMAJER, J.: 3D rekonstrukce povrchu lidské tváře z několika nekalibrovaných snímků. 2010.

[14] MORTAZAVIAN, P.; KITTLER, J.; CHRISTMAS, W.: 3D Morphable Model Fitting For Low-Resolution Facial Images. 2010.

[15] PARKE, F.; WATERS, I.: *Computer Facial Animation*. 2008.

[16] PAYSAN, P.; KNOTHE, R.; AMBERG, B.; aj.: *A 3D Face Model for pose and llumination invariant face recognition*. 2009.

[17] PSUTKA, J.: *Učící se systémy a klasifikátory*. 2011.

[18] VAŠÍČEK, Z.: Simulované žíhání. 2002.

Seznam obrázků

Příloha

Obsah přiloženého CD

Na přiloženém nosiči je k nalezení tato práce ve formátu PDF spolu se všemi potřebnými soubory pro spuštění algoritmu, včetně všech snímků, pořízených zařízením Kinect použitých ke klasifikaci. Dále obsahuje databázi příznaků Baselského modelu a všechny tři typy klasifikátorů.