Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Extrakce statického modelu lidského obličeje ze stereo/multiview videa

Plzeň 2020 Jakub Vašta

Místo této strany bude zadání práce.

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 29. července 2020

Jakub Vašta

Abstract

In this work, a tool for human face static model extraction from stereo/multiview video is designed. It is based on photogrammetry, followed by rigid and non-rigid registration. For this purpose, a toolset is created in order to automatize this task as much as possible.

Abstrakt

V této práci je navržen nástroj pro extrakci statického modelu lidského obličeje ze stereo/multiview videa, který je založený na fotogrammetrii, následované rigidní a nerigidní registrací. Za tímto účelem byla vytvořena sada nástrojů, jež tento proces co možná nejvíce automatizují.

Obsah

1 Úvod

V dnešní době jsou počítačové modely reprezentující objekty reálného světa využívány v mnoha oblastech, jako je například herní a filmový průmysl. Jedním způsobem, jak takový model získat je najmout umělce, který požadovaný objekt vymodeluje. Tento přístup je však časově velmi náročný, stojí mnoho peněz a realističnost modelu je výrazně ovlivněna schopnostmi jeho tvůrce. Dalším způsobem je získat počítačový model pomocí fotogrammetrie, nebo 3D skenerů. Dokážeme-li tento proces automatizovat, tak eliminujeme většinu negativ ručního modelování. Právě takový přístup, automatizace extrakce modelu založená na fotogrammetrii, bude náplní této práce. Konkrétně se jedná o extrakci statického modelu lidského obličeje ze stereo/multiview videa. Osoba stojící modelem bude během snímání otáčet hlavou, aby bylo zaznamenáno co možná nejvíce oblastí.

První část procesu, zabývající se fotogrammetrií, je již dobře známou oblastí, a proto existuje mnoho nástrojů, které je možné použít. Z tohoto faktu však také plyne, že samotné zvolení vhodného a kvalitního softwaru může být poměrně zdlouhavý proces. Máme-li k dispozici kvalitní fotogrammetrický nástroj, tak lze pouze s použitím snímacího zařízení rekonstruovat téměř libovolný objekt reálného světa. Pokud však není objekt statický, tak jistě nevystačíme s jedním snímacím zařízením, jelikož by si kvůli pohybu objektu dané snímky neodpovídaly.

Jak je možné nahlédnout, lidský obličej nelze považovat za statický, byť jen kvůli pohybu mimických svalů a mrkání. Navíc námi zvolený přístup vyžaduje otáčení hlavy osoby stojící modelem. Na základě těchto poznatků následuje po fotogrammetrii druhá část procesu zabývající se registrací již rekonstruovaných modelů, která má v případě rigidní registrace řešit pohyb hlavy a v případě nerigidní nechtěný pohyb osoby stojící modelem, jako je již zmíněné mrkání a různé další pohyby obličeje.

Práce je členěna do několika logických celků. Nejprve bude čtenář uveden do problematiky fotogrammetrie, následovanou teorií zabývající se rigidní a neregidní registrací. Dále bude navržen a popsán nástroj, který celý tento proces extrakce statického modelu lidského obličeje ze stereo/multiview videa co možná nejvíce automatizuje. V poslední části budou uvedeny výsledky a možná vylepšení celého procesu.

2 Fotogrammetrie

Fotogrammetrie je obor zabývající se rekonstrukcí scény (tvarů a pozice objektů v prostoru) z jejích projekcí, resp. snímků. Alternativním přístupem je využití 3D skenů, které bývá často přesnější. Přístup pomocí fotogrammetrie však na rozdíl od 3D skenování vyžaduje pouze použití snímacích zařízení (nejčastěji fotoaparátu/ů), která jsou v dnešní době snadno dostupná, oproti stále velmi drahým 3D skenerům. V obou případech je samozřejmostí nutnost zapojení softwaru, který nasnímaná data zpracuje a provede samotnou rekonstrukci.

Fotogrammetrie se dělí na dva hlavní směry. Prvním druhem je fotogrammetrie letecká (dálková), která se nejčastěji využívá pro mapování ve středním měřítku [8]. Druhým typem je fotogrammetrie blízká, jež se používá v řadě odvětví, např. v herním průmyslu při získávání modelů. Obvykle hovoříme o blízké fotogrammetrii v případě, že je vzdálenost snímané scény od snímacího zařízení přibližně do 100 metrů [8]. Další dělení je založeno na počtu snímků scény (resp. objektu), které jsou potřeba na rekonstrukci, a to fotogrammetrie na základě jednoho, nebo více snímků. V této práci bude použita a diskutována blízká fotogrammetrie s použitím více snímků, resp. stereo fotogrammetrie.

2.1 Základy fotogrammetrie

Nejprve je uveden matematický model dírkové (pinhole) kamery, viz rovnice 2.1, která je definována jako kamera s malou, a pouze jednou, závěrkou, která neobsahuje žádné čočky. Jedná se tedy o jednoduchý model, který nedokáže modelovat typ kamery nazývaný rybí oko [19] [18].

$$w \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \boldsymbol{K} \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.1}$$

Matice $\boldsymbol{P} = \boldsymbol{K} \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \end{bmatrix}$ se nazývá matice kamery (projekční matice) a obsahuje jak vnitřní, tak vnější parametry. Tato matice mapuje body ze scény na projekční rovinu. Parametr w je škálovací koeficient. Vnější parametry

jsou $\boldsymbol{R}$ a $\boldsymbol{t}$, tzn. rotace a translace kamery. Vnitřní parametry, viz rovnice 2.2, jsou parametry kamery. [19]

$$\boldsymbol{K} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.2}$$

kde $\boldsymbol{f}$ je ohnisková vzdálenost v pixelech, přičemž $\boldsymbol{f} = \left[\frac{F}{p_x}, \frac{F}{p_y}\right]$, kde F udává ohniskovou vzdálenost v reálných jednotkách (převážně v milimetrech) a $\boldsymbol{p}$ je velikost pixelu v reálných jednotkách, s je koeficient zkosení a $\boldsymbol{c}$ je optické centrum (principal point). [19]

Jak je možné si všimnout, tak matice kamery neuvažuje distorzi, a to z důvodu, že model byl definován bez použití optických čoček, které tuto distorzi způsobují. Budou uvedeny dvě možné distorze. První, takzvaná radiální distorze, je způsobena větším zahnutím paprsků s rostoucí vzdáleností od optického centra čočky. Druhá, tangenciální distorze, vzniká v případech, kdy čočka a senzor kamery nejsou rovnoběžně zarovnány. Pro matematické vyjádření těchto jevů viz [19]. Zkreslení způsobené optickými čočkami je možné odstranit za pomoci nasnímaní jistých vzorů z různých úhlů. Takovými vzory jsou např. šachovnice nebo množina kruhů.

Nyní se vrátíme k problematice fotogrammetrie. Konkrétně se budeme zabývat úkolem, jak ze snímků jedné scény (resp. objektu), které jsou pořízeny z různých úhlů, zjistit pozici nasnímaného objektu v reálném světě. Pokud bychom znali matici kamery (resp. vnější a vnitřní parametry kamery) a zároveň byli schopni nalézt korespondence mezi jednotlivými snímky, tak bychom mohli jednoduše pomocí triangulace tuto informaci vypočítat. Jak je však možno nahlédnout, požadavek na pozici kamery v prostoru (vnější parametry) je poměrně výrazně omezující a bylo by vhodné mít prostředky, jak tuto informace zjistit pouze ze znalosti snímků. Díky epipolární geometrii je však požadovanou informaci možné vypočítat, viz [10] [9]. Zbývá tedy najít korespondence mezi jednotlivými snímky. Za tímto účelem jsou využívány deskriptory snímků, viz [24].

2.2 Fotogrammetrický software

V současnosti existuje mnoho nástrojů zaměřených na fotogrammetrii. Tyto programy se liší jak v dostupnosti (tzn. volně dostupné a placené), tak v zaměření. Pro tuto práci je zásadní, aby se dal daný nástroj používat z pří-

kazové řádky, popř. aby se dal snadno skriptovat a bylo jej možno využívat zdarma. Dále bude uveden seznam dostupných a zvažovaných fotogrammetrických nástrojů.

2.2.1 Dostupný fotogrammetrický software

3DF Zephyr je fotogrammetrický nástroj, který lze získat v několika variantách, přičemž pouze jedna, *3DF Zephyr Free*, je zcela zdarma. Tato varianta umožňuje úplnou 3D rekonstrukci s limitem maximálně padesáti snímků použitých pro rekonstrukci a omezením pouze pro jedno *Nvidia GPU*. Bohužel nejde tento software používat z příkazové řádky a ani možnosti skriptování nejsou ideální, tzn. aplikace je určena pro používání v jejím grafickém rozhraní. Práce s tímto programem je poměrně snadná a umožňuje speciální typ rekonstrukce ze snímků zachycujících lidskou hlavu. Tento nástroj se osvědčil v jiných projektech zabývajících se podobnou problematikou, viz [22].

Colmap je fotogrammetrický nástroj, který je volně dostupný na *GitHubu* a je licencovaný pod BSD licencí. Umožňuje rekonstrukci jak za pomoci příkazové řádky, tak v rámci grafického rozhraní. Tento projekt vznikl jako práce výzkumných pracovníků, a tudíž je možné poměrně rozsáhle nastavit různé parametry procesu rekonstrukce. Program pracuje především s mračnem bodů, ale lze docílit i exportu do trojúhelníkového modelu. Oproti konkurenčním nástrojům musí být pro úpravy výsledného modelu použit jiný nástroj, např. *Meshlab*.

Meshroom je fotogrammetrický nástroj, který je volně dostupný na *GitHubu* pod licencí MPL2. Umožňuje rekonstrukci jak za pomoci příkazové řádky, tak v grafickém rozhraní. Oba přístupy jsou poměrně snadné na zprovoznění. V rámci grafického prostředí lze vytvářet vlastní graf zpracování za pomoci předem definovaných funkčních bloků. Tato funkcionalita umožňuje velmi snadné experimentování s různým nastavením programu. *Meshroom* je pouze grafická nadstavba nad frameworkem *AliceVision* a umožňuje velmi snadno zjistit, a to i v případě vlastního sestavení grafu zpracování, jak stejný proces spustit z příkazové řádky. Tento software umožňuje i využití předpřipravené funkcionality zaměřené na úpravu získaných trojúhelníkových modelů.

Agisoft Metashape je placený fotogrammetrický nástroj, který umožňuje rekonstrukci nasnímaných objektů za pomoci profesionálního grafického

rozhraní nebo skriptů vytvořených v jazyce *Python*. Tento software nabízí rozsáhlou sadu nástrojů pro editaci mračna bodů před samotnou tvorbou trojúhelníkového modelu.

Samozřejmě existuje více možností, jak volně dostupných pod různými licencemi, např. *MicMac*, *OpenMVG* atd., tak placených, např. *AutoDesk ReCap*, *Photomodeler*, *iWitnessPRO* atd.

2.2.2 Zvolený fotogrammetrický software

Ze zadání práce vyplývá, že je zapotřebí zvolit takový nástroj, který lze ovládat z příkazové řádky, nebo skriptovat nějakým jiným způsobem, který není závislý na interakci s uživatelem. Jinými slovy, ovládání pomocí grafického rozhraní není pro potřeby tohoto projektu dostačující. Z tohoto důvodu není vhodné použití programu *3DF Zephyr*, který je v této oblasti nedostatečný a značně horší než ostatní nástroje. Dalším velmi restriktivním omezením je rozpočet projektu, který se snaží cílit na co největší dostupnost vzhledem k nákladům nutným na zprovoznění celého procesu. Na základě toho byly zamítnuty placené programy, jako např. *Agisoft Metashape*. Vzhledem k snadnosti použití (jak v grafickém rozhraní, tak pomocí příkazové řádky) a možnosti velkých vlastních úprav celého procesu rekonstrukce, které umožňují snadné testování různých nastavení, byl zvolen program *Meshroom*. Na tomto místě je vhodné zmínit, že nejsou žádná další omezení na použití jiného nástroje a je tedy možné, v případě potřeby, vyměnit zvolený program pro fotogrammetrii za jiný a pokusit se tím vylepšit tuto část procesu, potažmo výsledek celého procesu.

Meshroom

Meshroom, resp. *AliceVision*, je projekt, který vychází ze spolupráce mezi výzkumem a průmyslovou činností, proto jsou v tomto softwaru implementovány výkonné algoritmy na základě současného stavu výzkumu, které jsou robustní a dosahují potřebné kvality, která je vyžadována pro průmyslové použití. Nejedná se tedy o triviální proces a problematiku, a proto jsou v této sekci popsány základní stavební kameny tohoto nástroje.

Extrakce význačných rysů (natural feature extraction) je jedna z prvních fází, která si klade za cíl nalezení význačných rysů v dostupných snímcích. Význačným rysem ve snímku rozumíme skupiny pixelů, které jsou do jisté míry rozlišitelné od ostatních skupin reprezentujících jiné význačné rysy.

Dalším požadavkem, který bychom chtěli od takových rysů ve snímcích, je jejich invariance vzhledem k pohybu kamery (tzn. rotace a translace), změně měřítka a do jisté míry i změně osvětlení.

Metod pro získání takových význačných rysů je několik, např. SIFT a AKAZE. Dále bude uveden základní princip první ze zmíněných metod, tzn. SIFT (scale-invariant feature transform), pro větší podrobnosti viz [17] [2].

Nejprve probíhá detekce scale-space extrémů (scale-space extrema detection), která hledá přes všechny naškálované snímky a vrací kandidáty, kteří splňují podmínky invariance vzhledem k měřítku a orientaci. Následuje lokalizace klíčových bodů (keypoint localization), která podrobně prozkoumá kandidáty z předchozího bodu. Poté proběhne přiřazení orientace (orientation assignment), které každému bodu přiřadí jednu nebo více orientací na základě gradientu. Nakonec je každému klíčovému bodu přiřazen deskriptor (keypoint descriptor). [17]

Po extrakci následuje filtrování, jelikož tato metoda produkuje značné množství významných rysů. [2]

Srovnání snímků (image matching) je další fází, která usiluje o nalezení shodné části nasnímaného objektu, resp. scény, na různých snímcích, a to na základě již vypočtených význačných rysech. Jelikož použití význačných rysů při jemném detailu snímků je velmi nákladné, tak se využívá slovníkových stromů (vocabulary tree). [2]

Srovnání význačných rysů (feature matching) je třetí fáze, v které je snaha o nalezení shodných význačných rysů ve dvojicích snímků. Z prvního snímku jsou získány význačné rysy, ke kterým jsou vždy nalezeni dva nejlepší kandidáti ze snímku druhého. Následně je vybrán z této dvojice ten lepší. Vychází se tedy z předpokladu, že je vždy pouze jedna shoda s kandidátem (takový předpoklad neumožňuje struktury se vzorem, ale obecně se prokázal jako robustní). Tento krok je výpočetně velmi náročný, tudíž není vhodné používat algoritmy typu hrubá síla, ale s výhodou se využívá Approximate Nearest Neighbor [2]. Následně jsou tyto dvojice využity při geometrickém filtrování za využití frameworku zvaného RANSAC (RANdom SAmple Consensus), který využívá epipolární geometrii. [2]

Struktura z pohybu (structure from motion) je čtvrtá a velmi důležitá fáze. Cílem tohoto kroku je najít z již získaných informací (z předchozích kroků) mračno bodů reprezentující nasnímaný objekt, resp. scénu, pozici

a vnitřní kalibraci kamer. Tato fáze je iterativní. Nejprve je zvolen počáteční pár snímků, z kterých se vypočítá fundamentální matice, viz [9] [10] (zásadní krok pro výsledek celé fáze). Nyní, když je známa pozice kamer, lze pomocí triangularizace určit pozici bodů v prostoru. Dále je přidán další snímek, který má nejlepší shodu s již hotovou částí a je nalezena pozice kamery tohoto snímku, tak aby potvrdila co nejvíce shod. Následuje upravení všech parametrů kamer i samotných rekonstruovaných bodů s případným odstraněním špatných výsledků. Dokud je možné lokalizovat další pohledy, resp. snímky, tak se pokračuje v iteracích. [2]

Určení hloubkových map (depth maps estimation) je fáze, v které se zjišťuje hloubka jednotlivých pixelů na základě kroku předchozího (struktura z pohybu). V *Meshroom*, resp. *AliceVision*, je použito Semi-Global Matching (SGM). Tento přístup vychází z dat vybraných z N nejlepších, popř. nejbližších, kamer (pro větší detaily viz [2]). Existuje mnoho alternativních přístupů, např. Block Matching nebo ADCensus. [2]

Tvorba trojúhelníkového modelu (meshing) je předposlední fáze, ve které se vytváří trojúhelníková síť, resp. model, ze spojených hloubkových map z kroku předchozího (určení hloubkových map), uložených v octree. Posléze proběhne 3D Dalaunayho tetrahedronizace, následovaná dalšími kroky, jako je komplexní hlasovací procedura (complex voting procedure) a Graph Cut Max-Flow pro získání povrchu. Na konci této části ještě probíhá Laplaceovské filtrování (pro více informací viz [2]). [2]

Texturování (texturing) je posledním krokem, v kterém je získáno UV mapování i pro body, které nemají přiřazeny žádné UV. [2]

2.3 Snímací zařízení pro fotogrammetrii

Volba snímacího zařízení, ať už ve formě fotoaparátu nebo kamery, je pro kvalitu rekonstrukce velmi důležitá. Z nekvalitního snímku není z principu možné provést tak kvalitní rekonstrukci, jako ze snímku kvalitního, při použití stejných technik fotogrammetrie, jelikož již ve vstupních datech chybí velké množství informací vzhledem ke kvalitnímu snímku. Proto se v této sekci budeme zabývat důležitými parametry snímacích zařízení, které ovlivňují kvalitu snímku.

2.3.1 Důležité parametry snímacích zařízení

Mezi důležité vlastnosti snímacích zařízení lze řadit mnoho parametrů, přičemž některé mají vyšší dopad na kvalitu než jiné. Následuje výčet právě takových parametrů.

Rozlišení snímače

Prvním důležitým parametrem je rozlišení snímače. Velmi zjednodušeně lze říct, že čím je větší rozlišení, tím je i lepší schopnost zachytit detail snímaného objektu. Samozřejmě to v realitě není tak jednoduché a velmi závisí na fyzické velikosti snímače. V dnešní době mají i levnější přístroje poměrně dobré rozlišení v řádech desítek megapixelů.

Velikost snímače

Dalším důležitým parametrem je fyzická velikost snímače, která přímo souvisí s velikostí odstupu signálu od šumu a dynamickým rozsahem, tzn. přímo určuje schopnost snímacího zařízení zachytit jemný detail. Obecně lze říct, že čím je větší fyzická velikost snímače, tím je zařízení lepší. V tomto ohledu excelují full-frame snímače.

Světelnost objektivu

Třetím parametrem, který je důležitý pro kvalitu snímku, je světelnost objektivu. Světelnost objektivu je charakteristika, která vyjadřuje poměr velikosti čočky vzhledem k ohniskové vzdálenosti. Tato vlastnost má souvislost s množství světla, které je objektiv schopen využít, resp. kolik světla dopadá na snímač. Tento rys je velmi důležitý v případě, že snímáme i za horších světelných podmínek.

Citlivost ISO

Citlivost snímače je dalším parametrem, který ovlivňuje výsledný snímek. Čím je větší citlivost snímače, tím je nutné méně světla dopadajícího na čip, ovšem za cenu narůstajícího šumu ve snímku. Opět lze nahlédnout, že je tato vlastnost důležitá zejména při snímání za horších světelných podmínek. Obecně lze říct, že chceme snímat s co nejnižší hodnotou ISO. Kvalitní snímací zařízení mají velký rozsah použitelných ISO hladin.

Ohnisková vzdálenost

Tento parametr, ohnisková vzdálenost, je důležitý z hlediska zorného úhlu snímacího zařízení, který je určen na základě poměru velikosti snímače ku ohniskové vzdálenosti. Objektivy se podle ohniskové vzdálenosti rozlišují na širokoúhlé, které mají krátkou ohniskovou vzdálenost, normální, jež nejvíce odpovídají lidskému oku, a teleobjektivy s velkou ohniskovou vzdáleností, které mají schopnost přiblížit vzdálený objekt.

Snímací frekvence

U fotoaparátu je snímací frekvence většinou poměrně nezajímavá informace, naopak u kamer velmi důležitý parametr. Jak již napovídá název, tak se jedná o počet snímků za sekundu, tzn. čím větší snímací frekvence, tím větší hladkost (návaznost) pohybu, který snímáme.

Lze říci, že určité parametry jsou více důležité u kamery a jiné u fotoaparátu. Také jsou některé parametry, které jsou více či méně důležité podle účelu snímacího zařízení. Samozřejmě s vyšší kvalitou snímacích zařízení výrazně roste i jejich cena.

2.3.2 Důležité vlastnosti pro fotogrammetrii

Pro fotogrammetrii je důležité zachytit co možná největší množství informace snímaného objektu a zároveň mít takových snímků pokud možno co nejvíce a to zároveň pod různými úhly. Z těchto důvodů je poměrně zřejmé, že jsou parametry jako rozlišení a fyzická velikost snímače velmi důležité pro kvalitní rekonstrukci nasnímaného objektu. Naopak parametry jako citlivost snímače, světelnost objektivu a zoom jsou téměř zanedbatelné, jelikož ze zadání úlohy, která je v této práci řešena, vyplývá, že se bude snímat ve studiu, kde je možné docílit velmi dobrého osvětlení scény a zároveň lze kameru fyzicky téměř libovolně přibližovat nebo oddalovat od snímaného objektu.

Pro kvalitní výsledky fotogrammetrie je dobré znát další parametry snímacího zařízení jako např. již zmíněnou ohniskovou vzdálenost, optické centrum (principal point) a distorzi optických čoček. Dalším parametrem, který je vhodné při práci se snímkem vzít v úvahu je, zda se jedná o globální nebo rolling závěrku (shutter). Pro účely fotogrammetrie chceme získat přístup k snímku, který není dále nijak upravován, jelikož všechny tyto úpravy mohou vést ke ztrátě původní informace. Takové úpravy často probíhají za účelem vylepšení vjemu ze snímku pro lidské oko.

2.3.3 Vybrané zařízení

Lze si všimnou, že zmíněné požadavky, konkrétně kvalita jednotlivých snímků a jejich množství pod různými úhly, jdou v jistém smyslu proti sobě. Jedná se zejména o rozpočet. Jinými slovy, je zapotřebí nakoupit větší počet snímacích zařízení, tzn. nehledáme jedno špičkové, ale zároveň velmi drahé snímací zařízení. Z těchto důvodů bylo upuštěno od fotoaparátů. Ze zadání úlohy vyplývá, že je zapotřebí natáčet video sekvenci, a proto v úvahu připadali dvě varianty: průmyslové a osobní kamery. První zmíněné, průmyslové kamery, mají mnoho kategorií od cenově dostupných až po velmi drahé modely. Tyto kamery mají vzhledem ke zmíněným vlastnostem, viz sekce 2.3.1, ideální parametry. Bohužel vybudování infrastruktury pro zhruba 10 takových kamer by nebyl zcela triviální úkol, a proto bylo od tohoto typu kamer upuštěno. Zbývá tedy poslední typ kamer, a to standardní kamery pro osobní použití. Jak lze snadno nahlédnout, tak tyto kamery jsou snadno dostupné. Jejich nasazení a použití je téměř triviální. Z uvedených důvodů byl tento druh snímacího zařízení vybrán, byť jsou tyto vlastnosti částečně na úkor kvality snímku.

Konkrétně je v této práci použito *GoPro HERO7 Silver*, pro parametry této kamery viz specifikace výrobce. Toto zařízení bylo vybráno na základě ceny, tzn. jsme schopni v daném rozpočtu nakoupit 10 takových zařízení. Mezi dobré vlastnosti této kamery patří natáčení videa ve 4K s 30 FPS, nebo natáčení ve Full HD s 60 FPS, zvukové ovládání, možnost stahování videa z kamery přes Wi-Fi, ovládání pomocí mobilního zařízení a neautomatický zoom. Mezi horší vlastnosti patří zejména velká distorze optické čočky (fisheye), stabilizace obrazu a fyzická velikost snímače. Na tomto místě je vhodné zmínit, že v případě většího rozpočtu, či jiných možností je možné použité zařízení nahradit za lepší a tím vylepšit výsledky fotogrammetrie, potažmo celého procesu jako takového.

3 Sjednocení modelů

Mějme množinu modelů, které reprezentují objekt zájmu, v případě této práce se jedná o lidský obličej, získané např. pomocí fotogrammetrie, viz sekce 2. Dalším logickým krokem je spojení těchto částí do jednoho, pokud možno co nejlepšího, modelu. Zmíněné části nemusejí vždy reprezentovat celý objekt zájmu, naopak ve většině případů obsahují pouze část většího celku, popř. se liší kvalita jednotlivých částí. V takovém případě je potřeba jednotlivé části na sebe registrovat. Řešením tohoto úkolu se zabývá mnoho prací, a tudíž existuje mnoho různých řešení. Obecně se registrace modelů dělí na registraci rigidní (tuhou), viz sekce 3.1, a registraci nerigidní, viz sekce 3.2, která do jisté míry umožňuje lokální deformaci modelu, na rozdíl od rigidní.

Posledním krokem je z takto zarovnaných modelů získat pouze jeden model, který bude pokud možno co nejlépe reprezentovat dobré vlastnosti jednotlivých částí. Tato problematika je v oblasti počítačové grafiky velmi dobře známa, jelikož se využívá pro získávání povrchů z naskenovaných dat, při remeshingu nebo při pouhé snaze o zaplnění děr v modelu. V sekci 3.3 bude uvedena metoda nazývaná Poisson Surface Reconstruction.

3.1 Rigidní registrace

Úkol rigidní registrace lze formulovat následovně: Nechť máme dva modely zdrojový S a cílový T, které jsou si navzájem podobné ve smyslu zachycení určité části stejného tvaru. Hledáme rotaci $\boldsymbol{R}$ a translaci $\boldsymbol{t}$ takovou, že po aplikaci této rotace a translace na zdrojový model (S), bude tento model mít co nejmenší chybu, dle zvolené metriky, vzhledem k cílovému modelu (T). Taková rotace $\boldsymbol{R}$ je vždy rigidní (to však neplatí u obecné transformace), tzn. nesmí deformovat tvar modelu (isometrie zachovávající orientaci). [11]

Rigidní registraci lze dělit na lokální (jemnou) a globální (hrubou). Nejvíce rozšířený zástupce lokální registrace se nazývá Iterative Closest Point (ICP), viz sekce 3.1.1. Lokální registrace vychází z předpokladu, že jsou si zdrojový (S) a cílový (T) model blízko. Naopak globální registrace nemá žádné předpoklady o počáteční poloze žádného z modelů. Existuje mnoho metod pro globální registraci, např. 4PCS [1], RANSAC [11], Fast Global Registration [25], metody založené na evolučních algoritmech [5] [6] atd. V následující části, viz sekce 3.1.2, bude uveden obecný postup pro RANSAC metody. [23] [11]

3.1.1 Iterative Closest Point (ICP)

ICP je iterační metoda založená na minimalizaci vzdálenosti zdrojového (S) a cílového (T) modelu ve smyslu nejmenších čtverců, viz rovnice 3.1. V případě dobrého prvotního odhadu rigidní transformace nebo jsou-li modely dostatečně blízko tato metoda velmi rychle konverguje [11]. Nyní bude stručně uveden základní popis tohoto algoritmu.

1. Nalezení nejbližšího bodu na cílovém modelu (T) pro každý vrchol zdrojového modelu (S), popř. pro zvolenou podmnožinu vrcholů. Tato vzdálenost je nejčastěji chápána jako eukleidovská. Následuje ořezání nevhodných bodů, např. pokud je dvojice nejbližších bodů ve větší vzdálenosti než zadaný práh. [23]

2. Stanovení vah pro jednotlivé body. Standardně je nastavena váha pro všechny korespondence na jednotkovou hodnotu, tzn. $\omega_{ij} = 1$, ale je možné váhy přenastavit tak, aby byly zvýhodněny body s vyžadovanými vlastnostmi. Jedna z možností je brát v úvahu vzájemný úhel, který svírají normály v korespondujících bodech, tzn. $\omega_{ij} = \boldsymbol{n}_i \boldsymbol{n}_j$. [23]

3. Samotný výpočet rotace $\boldsymbol{R}$ a translace $\boldsymbol{t}$. Konkrétně se jedná o minimalizaci vzdáleností korespondujících bodů a to ve smyslu nejmenších čtverců, viz rovnice 3.1

$$E(\boldsymbol{R}, \boldsymbol{t}) = \sum_{i=1}^{N_S} \omega_{ij} ||\boldsymbol{x_i} - (\boldsymbol{R c_j} + \boldsymbol{t})||^2, \tag{3.1}$$

   kde $\boldsymbol{R}$ je hledaná rotace, $\boldsymbol{t}$ je hledaná translace, N_S udává počet vrcholů zdrojového modelu (S), ω_{ij} udává váhu jednotlivých korespondencí, $\boldsymbol{x}_i$ je vrchol zdrojového modelu (S) a $\boldsymbol{c}_i$ značí bod korespondence na cílovém modelu (T) pro vrchol $\boldsymbol{x}_i$. [23]

4. Transformace zdrojového modelu (S) na základě nalezené rotace $\boldsymbol{R}$ a translace $\boldsymbol{t}$. [23]

5. Pokračuje se, dokud není dosaženo konvergence. Také je možné zavést maximální počet iterací, popř. pokud je chyba $E(\boldsymbol{R}, \boldsymbol{t})$ menší než je požadováno, tak algoritmus skončí. V ostatních případech se pokračuje v další iteraci. [23]

Existuje množství způsobů, jak algoritmus ICP urychlit a také mnoho postupů, které jsou z této základní podoby odvozeny [23].

3.1.2 RANSAC surface registration

Následuje popis základní podoby metod založených na RANSAC. Pro více podrobností viz [11].

1. Výběr, pokud možno uniformní, podmnožiny vrcholů zdrojového modelu (S). V těchto zvolených bodech jsou stanoveny hodnoty dvou hlavních křivostí κ_1, κ_2 a hlavní směr křivosti e_1 (first principal direction). [11]

2. Výběr, pokud možno uniformní, podmnožiny vrcholů cílového modelu (T). Pro každý prvek této množiny je na základě podobnosti zvolena nejlepší korespondence z podmnožiny vrcholů zdrojového modelu (S), který byl nalezen v kroku prvním. [11]

3. Nalezení rigidních transformací, tzn. rotace $\boldsymbol{R}$ a translace $\boldsymbol{t}$, a jejich ohodnocení kvality. Odstranění transformací se špatnou kvalitou. [11]

4. Množina transformací, které jsou uvažovány jako kandidáti, je chápána jako množina bodů v prostoru SE(3), ve kterém je následně vyhledáno místo s největší hustotou a to je poté prohlášeno za výslednou rigidní transformaci. [11]

Pro nalezení místa s největší hustotu je zapotřebí definovat metriku $d(\boldsymbol{T}_1, \boldsymbol{T}_2)$, kde $\boldsymbol{T}_i$ je uvažovaná transformace. Jak je možné nahlédnout, tak takových metrik existuje celá řada, viz [11], a jejich vhodnost, resp. nevhodnost, závisí na zadaných podmínkách. [11]

3.2 Nerigidní registrace

Úkol nerigidní (netuhé) registrace je téměř stejný jako v případě registrace rigidní (tuhé), viz sekce 3.1. Jediným rozdílem, jak již název napovídá, je požadavek na hledanou transformaci $\boldsymbol{A}$ (již se nejedná o pouhou rotaci $\boldsymbol{R}$, viz sekce 3.1), která nemusí být rigidní, tzn. nemusí být isometrií zachovávající orientaci. Naopak může do jisté míry lokálně deformovat zdrojový model (S). Stále je však žádoucí, aby byla transformace co možná nejvíce rigidní. [4] [16]

Nerigidní registrace má uplatnění především v případech, kdy je snaha o srovnání modelů, které zachycují stejný objekt, resp. část stejného objektu, s jistou mírou překrývajících se částí, který se v průběhu času deformuje. Jako příklad může posloužit sekvence modelů lidského obličeje z různých úhlů, získané pomocí fotogrammetrie. Takové modely budou do jisté míry

stejné, ale nelze zajistit, že se člověk, který stál modelem, mezi prvním a posledním (nebo i následujícím) snímkem nepohnul, a to žádným způsobem.

V následující části bude podrobně uvedena jedna z existujících metod, která se snaží problém nerigidní registrace řešit. Tento přístup je rozdělen na dvě části: hrubé zarovnání, viz sekce 3.2.1, a jemné zarovnání, viz sekce 3.2.2. [4] [16]

3.2.1 Coarse alignment

Cílem hrubého zarovnání je nalezení transformace $(\boldsymbol{A}_i, \boldsymbol{b}_i)$ pro každý vrchol trojúhelníkového modelu, za účelem minimalizování vzdálenosti mezi zdrojovým (S) a cílovým (T) modelem. Tento popsaný vztah je zachycen v rovnici 3.4. Na rozdíl od registrace tuhé (rigidní) není na transformaci kladena podmínka tuhosti (rigidity), přesto i v tomto případě je vyžadována jistá míra tuhosti hledané transformace.

Hrubé zarovnání je založeno na minimalizaci definované energie zapsané v rovnici 3.2, která se skládá ze tří energií (E_{fit} v rovnici 3.3, E_{rigid} v rovnici 3.5 a E_{smooth} v rovnici 3.6). Každá z těchto energií reprezentuje vyžadovanou vlastnost hledané transformace. Těmto energiím jsou dále přiřazeny váhy (α_{fit}, α_{reg} a α_{smooth}), které vyjadřují důležitost jednotlivých energií. Tyto váhy se mohou v jednotlivých krocích optimalizace lišit. [4] [16]

$$E_{total} = \alpha_{fit}E_{fit} + \alpha_{reg}(E_{rigid} + \alpha_{smooth}E_{smooth}) \tag{3.2}$$

Deformační graf

Hrubé zarovnání se nepočítá nad celým modelem, ale nad modelem zjednodušeným, který budeme nazývat deformační graf. Takový graf je získán na základě geodetické vzdálenosti a sestává se z uzlů, které jsou podmnožinou vrcholů původního modelu. Deformační graf musí splňovat následující vlastnosti:

1. Uzel deformačního grafu je vrchol z původního modelu
2. Vzdálenost mezi uzly deformačního grafu není menší než δ (ideálně je rovna δ).
3. Uzel deformačního grafu zná své 4 nejbližší uzly deformačního grafu a vzdálenosti k nim
4. Uzel deformačního grafu reprezentuje všechny vrcholy původního modelu v okolí velikosti δ

5. Každý vrchol trojúhelníkového modelu zná své 4 nejbližší uzly deformačního grafu a jejich vzdálenost.

Geodetická vzdálenost je využita proto, aby bylo zajištěno, že body budou blízko sebe ve smyslu polohy na povrchu modelu. Eukleidovská vzdálenost, byť je oproti geodetické vzdálenosti mnohem méně výpočetně náročná, není tento požadavek v jistých případech schopná splnit, a proto není vhodnou metrikou. Pro výpočet geodetické vzdálenosti lze využít metodu Fast Marching [15]. [4] [16]

Korespondence

Minimalizace energií je definována vzhledem ke korespondencím mezi zdrojovým (S) a cílovým (T) modelem. Korespondencí bodu $\boldsymbol{x}_i$ nazýváme takový bod $\boldsymbol{c}_i$ ze zdrojového modelu (S), který je získán jako průsečík paprsku vyslaném ve směru normály v bodě $\boldsymbol{x}_i$ s povrchem cílového modelu (T). Dále můžeme hodnotit korespondence dle jejich kvality, např. na základě vzdálenosti nebo úhlu, který svírají normály v korespondujících bodech. Špatné korespondence je vhodné při optimalizaci nevyužívat.

Energie

Cílem optimalizace je pro každý uzel deformačního grafu vypočítat afinní transformaci definovanou maticí $\boldsymbol{A}$ s rozměry 3×3 a vektorem $\boldsymbol{b}$ s rozměry 3×1. Transformace nalezená pro uzly grafu je následně přenesena na původní vrcholy zdrojového modelu (S) pomocí lineárního blendingu a to na základě čtyř nejbližších uzlů deformačního grafu. Každý uzel má stanovenou váhu [4] [16]

$$\omega_{ij} = \frac{1}{\sum_{j \in N(i)} \omega_{ij}} \left(\frac{1 - d(\boldsymbol{x}_i, \boldsymbol{x}_j)}{d_{max}} \right)^2,$$

kde $N(i)$ je množina 4 nejbližších uzlů deformačního grafu, $d_{max} = max[d(\boldsymbol{x}_i, \boldsymbol{x}_j)]$ pro všechny $j \in N(i)$ a $d(\cdot, \cdot)$ reprezentuje geodetickou vzdálenost mezi dvěma body.

Energie E_{fit} v rovnici 3.3 vyjadřuje požadavky známé jako point-to-point a point-to-plane. Jinými slovy zachycuje vzdálenost bodů, které spolu korespondují a zda bod leží v rovině definované korespondujícím bodem.

$$E_{fit} = \sum_{i=1}^{m} (\alpha_{point} ||\widetilde{\boldsymbol{x}}_i - \boldsymbol{c}_i||^2 + \alpha_{plane} (\boldsymbol{n}_i^T \cdot (\widetilde{\boldsymbol{x}}_{\boldsymbol{i}} - \boldsymbol{c}_{\boldsymbol{i}}))^2), \tag{3.3}$$

kde m je počet vrcholů zdrojového modelu (S), $\boldsymbol{c}_i$ je bod z cílového modelu (T) korespondující s bodem $\boldsymbol{x}_i$ (bod ze zdrojového modelu (S)), $\boldsymbol{n}_i^T$ je normála v korespondujícím bodě $\boldsymbol{c}_i$ a $\widetilde{\boldsymbol{x}}_i$ získáme následovně. [4] [16]

$$\widetilde{\boldsymbol{x}}_i = \frac{1}{\sum_{j \in N(i)} \omega_{ij}} \cdot \sum_{j \in N(i)} \omega_{ij}(\boldsymbol{A}(\boldsymbol{x}_i - \boldsymbol{x}_j) + \boldsymbol{x}_j + \boldsymbol{b}_j), \tag{3.4}$$

kde $N(i)$ je množina 4 nejbližších uzlů deformačního grafu, ω_{ij} udává váhu pro daný sousední uzel, $\boldsymbol{A}$ je hledaná matice rotace a $\boldsymbol{b}$ je hledaný vektor translace. [4] [16]

Energie E_{rigid} v rovnici 3.5 zachycuje požadavek co největší rigidity (tuhosti) transformace. Tato tuhost je zachycena jednotlivými členy této energie, tzn. skalární součin sloupců matice by měl být nulový a velikost vektoru daného příslušným sloupcem matice by měla být jednotková, jedná-li se o transformaci rigidní.

$$E_{rigid} = \sum_{i=1}^{n}((\boldsymbol{a}_{i,1}^T \cdot \boldsymbol{a}_{i,2}^T)^2 + (\boldsymbol{a}_{i,1}^T \cdot \boldsymbol{a}_{i,3}^T)^2 + (\boldsymbol{a}_{i,2}^T \cdot \boldsymbol{a}_{i,3}^T)^2 + \\ + (1 - ||\boldsymbol{a}_{i,1}||)^2 + (1 - ||\boldsymbol{a}_{i,2}||)^2 + (1 - ||\boldsymbol{a}_{i,3}||)^2), \tag{3.5}$$

kde značení $\boldsymbol{a}_{i,x}^T$ vyjadřuje x-tý sloupec matice $\boldsymbol{A}$ a n je počet uzlů deformačního grafu. [4] [16]

Energie E_{smooth} v rovnici 3.6 vyjadřuje požadavek na jakousi hladkost výsledného modelu po transformaci. Hladkost je v této energii zachycena podobností transformací pro sousední uzly deformačního grafu.

$$E_{smooth} = \sum_{i=1}^{n} \sum_{j \in N(i)} ||\boldsymbol{A}_i(\boldsymbol{x}_j - \boldsymbol{x}_i) + \boldsymbol{x}_i + \boldsymbol{b}_i - (\boldsymbol{x}_j + \boldsymbol{b}_j)||^2, \tag{3.6}$$

kde n je počet uzlů deformačního grafu, $N(i)$ je množina 4 nejbližších uzlů deformačního grafu, $\boldsymbol{A}$ je hledaná matice rotace a $\boldsymbol{b}$ je hledaný vektor translace. [4] [16]

Minimalizace energie probíhá na základě standardního Guass-Newtonova algoritmu, viz sekce 3.2.1. Nejprve jsou vždy určeny korespondence mezi zdrojovým (S) a cílovým (T) modelem, následované dalším krokem optimalizace. Nastavení vah je navrženo následovně: $\alpha_{fit} = 0.1$, $\alpha_{reg} = 1000$, $\alpha_{smooth} = 0.1$, $\alpha_{point} = 0.1$ a $\alpha_{plane} = 1$ [4] [16]. Po dosažení konvergence je postupně regularizační člen uvolňován, tzn. $\alpha_{reg}^{(i+1)} = 0.1\alpha_{reg}^{(i)}$ dokud

platí $\alpha_{reg} \geq 1$ [4] [16]. Samozřejmě je možné omezit i počet iterací Gauss-Newtonova algoritmu. Základní odhad transformace každého uzlu je matice identity pro rotaci a nulový vektor translace. [4] [16]

Gauss-Newtonův algoritmus

Gauss-Newtonův algoritmus [7] [21] vychází z Newtonovy metody pro nalezení minima dané funkce a je používán při řešení nelineárního problému nejmenších čtverců. Tuto metodu lze použít pouze při minimalizaci součtu kvadrátů reziduí, viz rovnice 3.7, přičemž reziduum je reprezentováno funkcí více proměnných.

Gauss-Newtonův algoritmus iteračním přístupem minimalizuje sumu kvadrátů reziduí

$$S(\boldsymbol{\beta}) = \sum_{i=1}^{m} r_i^2(\boldsymbol{\beta}), \tag{3.7}$$

kde m je počet reziduí, n udává počet proměnných, tzn. $\boldsymbol{r} = (r_1(\boldsymbol{\beta}), \ldots, r_m(\boldsymbol{\beta}))$ a $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_n)$. Systém musí splňovat podmínku $m \geq n$.

Jak již bylo zmíněno, algoritmus pracuje v iteracích. Na základě prvního odhadu $\boldsymbol{\beta}^{(0)}$ vypočteme další hodnoty dle rovnice 3.8.

$$\boldsymbol{\beta}^{(s+1)} = \boldsymbol{\beta}^{(s)} - \alpha\boldsymbol{x} \tag{3.8}$$

$$(\boldsymbol{J}_r^T \boldsymbol{J}_r)\boldsymbol{x} = \boldsymbol{J}_r^T \boldsymbol{r}(\boldsymbol{\beta}^{(s)}), \tag{3.9}$$

kde prvky Jacobiho matice vypočteme dle rovnice 3.10, v případě že vektory $\boldsymbol{r}$ a $\boldsymbol{\beta}$ jsou ve sloupcové notaci.

$$(\boldsymbol{J}_r)_{ij} = \frac{\partial r_i(\boldsymbol{\beta}^{(s)})}{\partial \beta_j} \tag{3.10}$$

Výpočet končí jeli dosaženo konvergence, popř. po zadaném počtu iterací, dosažení přijatelné chyby nebo při identifikaci dostatečně malé změny mezi iteracemi. Je nutné si uvědomit, že tento algoritmus nezaručuje nalezení globálního optima z libovolného počátečního odhadu.

3.2.2 Fine scale alignment

Jemné zarovnání probíhá po hrubém a má za úkol zajistit zarovnání jemných detailů (vysokofrekvenčních geometrický detailů) zdrojového modelu (S) na cílový model (T). Za tímto účelem je sestrojena energie E_{total}, viz rovnice 3.11, která se snaží vynutit minimalizaci vzdálenosti korespondujících bodů (viz sekce 3.2.1) rovnicí 3.12 a zároveň co nejmenší rozdíl mezi vektory přemístění $\boldsymbol{d}$ sousedících bodů rovnicí 3.13. Na rozdíl od hrubého zarovnání je tato energie počítána nad všemi vrcholy zdrojového modelu (S). [4] [16]

$$E_{tot} = E_{fit} + E_{reg} \tag{3.11}$$

$$E_{fit} = \sum_{i \in V} ||\boldsymbol{x}_i + \boldsymbol{d}_i - \boldsymbol{c}_i||^2, \tag{3.12}$$

kde $\boldsymbol{x}_i$ je bod ze zdrojového modelu (S), $\boldsymbol{c}_i$ je bod z cílového modelu (T) korespondující s bodem $\boldsymbol{x}_i$ a $\boldsymbol{d}_i$ je hledaný vektor přemístění pro bod $\boldsymbol{x}_i$. [4] [16]

$$E_{reg} = \sum_{j \in N(i)} ||\boldsymbol{d}_i - \boldsymbol{d}_j||^2, \tag{3.13}$$

kde $N(i)$ je množina vrcholů sousedících (incidenčních) s vrcholem $\boldsymbol{x}_i$, $\boldsymbol{d}_i$ je hledaný vektor přemístění pro vrchol $\boldsymbol{x}_i$ zdrojového modelu (S) a $\boldsymbol{d}_j$ je vektor přemístění sousedního vrcholu. [4] [16]

Je žádoucí, aby byl zvolen práh $t = \sigma * BoundingBox$, který vyjadřuje maximální přípustnou vzdálenost nalezené korespondence. To znamená, že pokud $||\vec{x_i}, \vec{c_i}|| > t$, pak pro tento bod je $E_{fit} = 0$. Autoři navrhují nastavit parametr σ na hodnotu 0.1 [4] [16].

Jak je možné nahlédnout, rovnice 3.11 zachycuje lineární soustavu rovnic, kterou je možno řešit například metodou nejmenších čtverců, viz rovnice 3.14.

$$\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{b} \tag{3.14}$$

3.3 Poissonovská rekonstrukce

Poissonovská rekonstrukce [14] slouží k získání trojúhelníkového modelu z mračna bodů s normálami, které může být získáno například pomocí skenování, v průběhu rekonstrukce nebo z již existujícího modelu. Tento postup vytváří uzavřené (watertight) povrchy a je velmi odolný vůči přítomnosti

šumu ve vstupních datech. Tento přístup může být také použit pro zaplnění děr v již existujícím modelu, popř. k remeshingu. [14]

Poissonovská rekonstrukce je jako mnoho jiných metod řešících tento problém založena na použití implicitních funkcí. Konkrétně je vypočtena indikační funkce χ, která je definována následovně. [14]

$$\chi = \begin{cases} 0 \text{ pro body ležící uvnitř modelu} \\ 1 \text{ pro body ležící vně modelu} \end{cases} \tag{3.15}$$

Lze pozorovat, že gradient indikační funkce je roven 0 téměř všude, kromě bodů blízkých povrchu, kde je roven dovnitř směřujícím normálám požadovaného povrchu. Tato vlastnost částečně vychází z charakteru indikační funkce, která je téměř všude, kromě hledaného povrchu, konstantní. Je možné nahlédnout, že mračno bodů s normálami, které reprezentuje vzorky hledaného povrchu, lze chápat jako gradient indikační funkce hledaného povrchu. Jinými slovy, lze problém formulovat jako výpočet skalární funkce χ, jejíž gradient co možná nejlépe aproximuje mračno bodů s normálami, tzn. vektorové pole. Pokud na takto formulovaný problém navíc aplikujeme operátor divergence, tak získáme v matematice dobře známý Poissonovský problém (viz rovnice 3.16), tzn. výpočet skalární funkce χ jejíž Laplacián je roven divergenci vektorového pole definovaného pomocí mračna bodů s normálami. [14]

$$\Delta\chi \equiv \nabla \cdot \nabla\chi = \nabla \cdot \vec{V} \tag{3.16}$$

Převedení zadaného problému na Poissonvský má mnoho výhod, a to například existenci mnoha robustních a efektivních algoritmů na řešení právě Poissonovského problému, jelikož tento problém zasahuje do mnoha disciplín. Dále je výhodná vlastnost, že oproti jiným přístupům využívajících implicitní funkce není tato metoda lokální, a tudíž nepotřebuje dělení prostoru s následným kombinováním výsledků, ale je globální, tzn. bere v úvahu všechny zadané body. Tato skutečnost vede k tvorbě velmi hladkých povrchů, jako např. při použití RBF, a je velmi tolerantní k šumu ve vstupních datech. [14]

Naopak se zdá, že výsledek je někdy až přílíš vyhlazen. Z tohoto důvodu byla vytvořena alternativní metoda nazvaná Screened Poisson Reconstruction, která tuto někdy nechtěnou vlastnost potlačuje, více podrobností [13].

4 Proces extrakce statického modelu lidského obličeje ze stereo/multiview videa

Tato část práce se bude zabývat konkrétními postupy použitými při procesu extrakce statického modelu lidského obličeje ze stereo/multiview videa a jejich implementací. Zvolený přístup vychází z teoretických poznatků popsaných v sekcích 2 a 3. Dále bude také představen návrh nástroje pro automatizaci celého procesu a jeho jednotlivé části.

Hlavní programovací jazyk použitý pro implementaci procesu extrakce je *Python* a to hned z několika důvodů. Jedním z hlavních argumentů byla nutnost použití již existujícího řešení pro fotogrammetrii, viz sekce 2, které bude voláno pomocí konzole, nebo jiným způsobem naskriptováno. Z tohoto požadavku je možné vidět, že konkrétní stavební bloky procesu jsou implementovány v programovacích jazycích zvolenými autory daného nástroje a vyvstává tedy nutnost jejich spojení do jednoho celku za pomoci skriptů. K tomuto účelu je *Python* velmi vhodný. Další výhodou programovacího jazyku *Python* je dostupnost mnoha modulů řešících jak zpracování videa, např. *opencv*, tak i řadu dalších běžně známých oblastí počítačové grafiky, např. *open3d* pro vizualizace a lokální registraci.

Celý proces byl navržen pro operační systém *Windows*. Port nástroje na *Linux* by mohl narazit pouze na nedostupnost některých komponent pro tento operační systém, jelikož zbytek programu byl implementován v *Pythonu*, který je na operačním systému *Linux* samozřejmostí.

4.1 Obecný popis procesu

Program je konzolová aplikace, která se spouští s jedním parametrem, konfiguračním souborem, viz listing 1. Celý proces extrakce statického modelu lidského obličeje ze stereo/multiview videa se sestává z několika hlavních stavebních bloků, konkrétně se jedná o získání video záznamů, sekce 4.2, zpracování videa, sekce 4.3, rekonstrukce, sekce 4.4, registrace, sekce 4.5, a remeshing, sekce 4.6. Většina těchto bloků je dále členěna do modulů, které vykonávají specifické úlohy (podrobněji popsáno v příslušných sekcích).

```
python main.py --config-file path_to_config_file
```

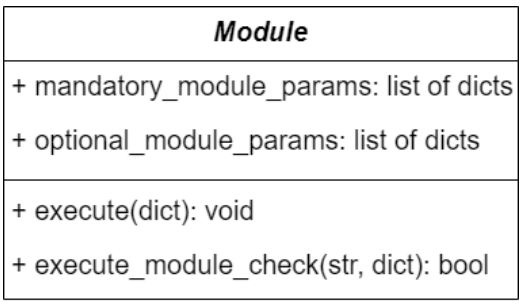
Listing 1: Spouštění nástroje

4.1.1 Popis modulu

Každý modul staví na základní abstraktní třídě, viz diagram 4.1. Tato modulární architektura byla zvolena s cílem snadné rozšiřitelnosti programu. Také výměna starého modulu za nový je v takto navrženém nástroji velmi jednoduchý úkon.

Jak je možné vidět z diagramu 4.1, každý modul musí obsahovat dvě funkce:

- **execute** spouští výpočet modulu a závisí na slovníku, který drží konfiguraci modulu.
- **execute_module_check** slouží ke kontrole vstupních parametrů. Vyžaduje dva argumenty: cestu ke konfiguračnímu souboru a opět slovník obsahující konfiguraci modulu. Tato funkce je standardně zavolána hned po spuštění programu, aby byly chyby v konfiguraci objeveny co nejdříve.



Obrázek 4.1: Diagram abstraktní třídy modulu

Dále by měl každý modul obsahovat dvě proměnné: pro povinné parametry modulu `mandatory_module_params` a `optional_module_params` pro volitelné parametry modulu, viz listing 2. Nastavení těchto proměnných definuje očekávané parametry pro běh modulu a odráží se v konfiguračním souboru, viz sekce 4.1.2.

```
mandatory_module_params = [
    {
        'name': param_name,
        'check' : [
            ...
        ]
    }
]

optional_module_params = [
    {
        'name': param_name,
        'value': param_default_value,
        'check' : [
            ...
        ]
    }
]
```

Listing 2: Definice konfigurace modulu

Jak lze z ukázky konfigurace, listing 2, nahlédnout, tak položka `name` udává jméno jak povinných, tak nepovinných parametrů modulu. Dále pole `check` obsahuje předdefinovanou sadu klíčových slov. Tato klíčová slova vyjadřují vlastnosti parametrů, které musí splňovat, nebo akci preprocesoru. Pole `check` je opět shodné pro oba typy parametrů. Jedinou výjimkou je proměnná `value`, jež udává výchozí hodnotu pro volitelný parametr v případě, že není hodnota uvedena explicitně uživatelem v konfiguračním souboru.

Předdefinované kontroly parametrů slouží ke kontrole definovaných parametrů každého modulu. Aby nebylo nutné psát opakující se kontroly stále znovu, tak byl vytvořen `enum` obsahující základní testy, např. zda cesta vede k souboru nebo složce, kontrola jejich existence, nebo kontrola datových typů. Tento `enum` s názvem `Constraints` lze nalézt v pomocné třídě `helper.py`.

Zapojení modulu do procesu je záležitost, na kterou bylo při návrhu nástroje myšleno a je proto velmi jednoduchá a přímočará. V hlavním modulu aplikace `PipelineModule`, je nutné v metodě `__register_modules`

```
modules:
    -
        module: external
        input: file_path
```

Listing 3: Ukázka konfiguračního souboru

registrovat metody `execute` a `execute_module_check`.

Moduly, které vykonávají podobnou práci, mohou být sloučeny do jednoho modulu a vytvářet takzvané submoduly. Tyto submoduly nejsou registrovány samostatně, ale je zaregistrován jejich dispatcher. Dispatcher následně volá jednotlivé submoduly, tak jako v případě `PipelineModule`. Pro více detailů je vhodné nahlédnout do implementace.

4.1.2 Konfigurační soubor

Tento soubor slouží k upravování procesu dle požadavků uživatele. Konkrétně se v konfiguračním souboru definuje seznam modulů, jež budou spuštěny dle specifikovaného pořadí, a to se zadanými parametry. Ukázka jednoduchého konfiguračního modulu viz listing 3. Podrobnosti o možných parametrech každého modulu budou vždy uvedeny v příslušné sekci zabývající se daným modulem. Konfigurační soubor je ve formátu `yaml`, který byl zvolen na úkor typu `json`. Rozhodnutí bylo učiněno z důvodů lepší čitelnosti typu `yaml` pro běžné uživatele, kteří nemusí být vždy programátoři.

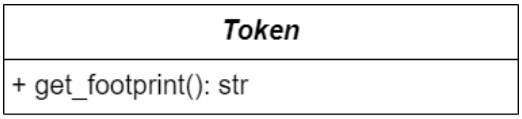
Modul může být i takzvaně externí, tzn. odkazovat na jiný konfigurační soubor (se stejnými požadavky na strukturu). Externí modul je označen klíčovým slovem `external` a je definován povinným parametrem `input`, který obsahuje cestu ke konfiguračnímu souboru, viz listing 3. Tento přístup umožňuje vytvářet adresářovou strukturu projektu a držet příslušné konfigurační soubory s odpovídajícím zdrojem/výstupem dat daného modulu.

4.1.3 Safe points

Safe points je jednoduchá implementace transakcí. Každý modul si může definovat svůj takzvaný token založený na abstrakci `Token`, viz diagram 4.2. Je zodpovědností programátora definujícího nový token, přetížit funkci `get_footprint`, aby dávala jedinečný hash pro různé tokeny. Každý token se nachází buď ve stavu `open`, nebo `close`. Tato informace je perzistentní, tzn. je zachována i po ukončení programu, a to i při konci neočekávaném.

Konkrétně jsou tyto informace uloženy v souboru `safe-points.tmp`.

Celá funkcionalita slouží k tomu, aby se hotová činnost zbytečně neprováděla znovu. V případě výpočetně náročných kroků, nebo ovlivnění modulu vnějšími závislostmi, které mohou vést k uživatelem neočekávaným chybám, je jistě vhodné mít takovou funkcionalitu a nebýt nucen dělat znovu již hotovou práci, nebo požadovat po uživateli přepisování konfiguračních souborů.



Obrázek 4.2: Diagram třídy Token

4.2 Získání video záznamů

Prvním logickým krokem procesu extrakce statického modelu lidského obličeje ze stereo/multiview videa musí být získání video záznamu obličeje osoby, která stojí modelem. Za tímto účelem bylo vytvořeno zařízení pro připevnění zvolených kamer, viz obrázek 4.3. Pro snímání videa byly využity kamery *GoPro Hero 7 Silver*. V projektu je zapojeno 10 těchto kamer. Další zařízení nebyla použita z důvodů omezeného rozpočtu, nikoliv z důvodů dostatečnosti tohoto pro fotogrammetrii poměrně malého počtu (například ukázkový model nástroje *Meshroom* využívá přes 100 snímků z různých úhlů).

Camera rig je vytvořené zařízení, které se sestává ze dvou kruhů připravených pro zavěšení kamer, viz obrázek 4.3. Tyto kamery jsou připevněny pomocí binderů a jsou rozmístěny tak, aby rovnoměrně pokrývaly úhel o velikosti 60°. Kruhy s kamerami jsou umístěny na závitových tyčích a lze je libovolně posouvat nahoru, resp. dolů, přemístěním matic, na kterých sedí. Poslední částí zařízení je základna, která umožňuje snadný vstup pro člověka, který stojí modelem. Zároveň je tato základna sestrojena tak, aby ji bylo možné podložit, tzn. vyvýšit celé zařízení. Také je možné na zadní část základny umístit závaží, aby se vyrovnala váha kamer umístěných pouze na jedné části kruhů. Tímto opatřením se zvýší stabilita celého zařízení. Jak je možné nahlédnout, je velmi snadné rozšířit zařízení o další snímací zařízení, popř. o další kruhy pro kamery.

Obrázek 4.3: Camera rig

4.2.1 Osvědčené postupy

Během získávání video záznamů a následné rekonstrukce za pomoci fotogrammetrických nástrojů se osvědčily následující přístupy a techniky.

- Kvalitní osvětlení scény pomocí studiových světel. Obecně všechny postupy vedoucí k co možná nejlepší kvalitě snímkům jsou žádoucí.
- Namalování různě barevných značek na obličej modelu. Za tímto účelem byly použity barvy přímo určené pro obličej a lehce omyvatelné vodou. Tento krok by měl přispět k lepší detekci význačných rysů a hledání korespondencí, viz sekce 2.2.2.
- Rozmístění speciálních značek, takzvaných CCTagů, do scény na pozadí. Tento krok by měl přispět k lepší detekci význačných rysů a hledání korespondencí, viz sekce 2.2.2. V současné verzi nástroje *Meshroom* je nutné dbát na velikost těchto značek, jelikož je známa chyba v nynějším oficiálním release buildu, která způsobuje, že pro příliš velké značky nalezení ve scéně vždy selže.
- Použití hlasového ovládání kamer pomocí příkazů *GoPro start recording* pro spuštění a *GoPro stop recording* pro ukončení zaznamenávání videa.

4.3 Zpracování videa

Dalším logickým krokem, jsou-li již získány video záznamy obličeje osoby, která stojí modelem, je zpracování těchto záznamů. Tento proces se skládá z následujících kroků: import videa ze snímačů (sekce 4.3.1), synchronizace video stop (sekce 4.3.2) a extrakce snímku z videa (sekce 4.3.3), popř. ještě oprava distorze snímků. Jednotlivé fáze jsou dále podrobně popsány v příslušných sekcích.

4.3.1 Import videa

Jak již bylo několikrát uvedeno, v tomto projektu jsou využívány kamery *GoPro Hero 7 Silver*, které si ukládají video záznamy na svoje lokální úložiště, konkrétně SD kartu. První možností, jak importovat videa do počítače by bylo ručně tyto záznamy stahovat pomocí USB kabelu připojenému k počítači, nebo s využitím čtečky SD karet. Takový postup by byl velmi pracný a zdlouhavý. Další možností je využití Wi-Fi sítě, kterou každá kamera poskytuje a následné stáhnutí videí do počítače. Pro druhou možnost, stahování přes Wi-Fi síť, byl naimplementován příslušný modul.

Průběh importu

Proces importu videa do počítače probíhá následovně. Ve specifikované složce pro import se vytvoří adresářová struktura obsahující složky očíslované dle pořadí stahovaného videa. Do každé takové složky se stáhne ze všech kamer příslušné video ve formátu `mp4`, které je pojmenováno dle identifikačního řetězce příslušné kamery. Například do složky s názvem `0` se stáhnou všechna první videa z SD karty každé kamery. Videa se po stažení automaticky nemažou, a proto je nutné ruční promazání před dalším snímáním modelů, tak aby bylo zaručeno, že související videa jsou na SD kartě kamery uložena ve stejném pořadí.

Tento modul využívá funkcionalitu *safe points*, jelikož se poměrně často stává, že se nelze k nějaké kameře připojit hned napoprvé, nebo vyvstane jiný problém.

Interní informace kamery

Na adrese `http://10.5.5.9/gp/gpMediaList` jsou po připojení k příslušnému zařízení dostupné informace o obsahu kamery. Tyto údaje jsou ve formátu `json`. Další adresa `http://10.5.5.9/videos/DCIM/{media_id}` obsahuje video, resp. jakékoliv médium, uložené na kameře, které je dostupné

```
module: import
output: folder_path
cameras:
    - camera_id_0
    - camera_id_1
wget-bin: file_path
```

Listing 4: Konfigurace import modulu

ke stažení. Jak lze snadno nahlédnout, jméno příslušného média se dosadí za `{media_id}`. Pro stahování videí se využívá utilita *wget*.

Příprava zařízení

Aby vše fungovalo, tak je nejprve nutné spárovat GoPro mobilní aplikaci s danou kamerou. Následně se musí uživatel připojit k Wi-Fi síti kamery s počítačem, kam chce videa stahovat, zadat specifické heslo pro každou kameru (toto heslo definuje sama kamera) a uložit daný profil. Tento postup je nutné provádět pouze při prvním použití kamery, nebo pokud nastanou s kamerou neočekávané problémy a je nutné ji resetovat. Je doporučeno nechávat výchozí název kamery, resp. Wi-Fi sítě kamery, pro snadnější fyzické dohledání (zařízení jsou označeny štítky).

Konfigurace modulu pro import videa, viz listing 4, závisí na parametru `output`, který udává složku pro import a seznamu identifikačních řetězců kamer uložených v poli `cameras`. Identifikátor kamery v konfiguračním souboru musí odpovídat názvu Wi-Fi sítě kamery. `Wget-bin` je cesta k programu *wget.exe*.

4.3.2 Synchronizace videa

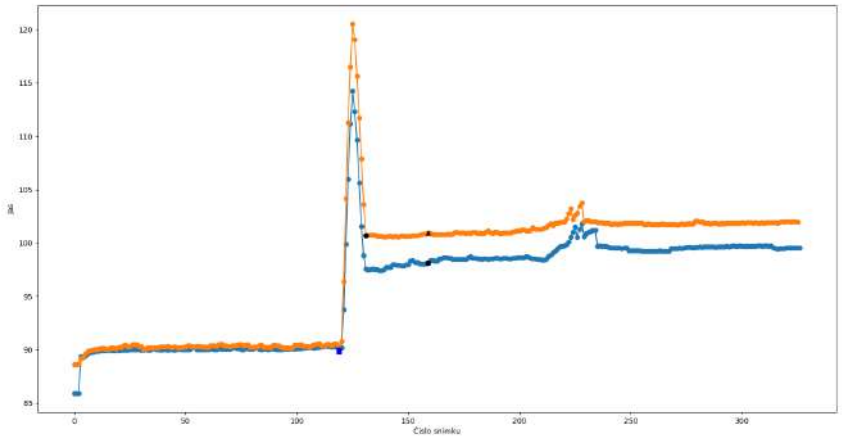
V procesu extrakce statického modelu lidského obličeje ze stereo/multiview videa bylo využito několika snímacích zařízení a to z důvodů možných pohybů osoby stojící modelem. Jinými slovy, pokud bychom využívali pouze jedno snímací zařízení, např. fotoaparát, tak bychom mohli při rekonstrukci modelu pomocí fotogrammetrie narazit na problém, že se model mezi jednotlivými snímky pohnul. Pokud by tomu tak opravdu bylo, tak by snímky nezachycovali stejnou pózu objektu. Tento fakt by mohl výsledky fotogrammetrie výrazně negativně ovlivnit. Z tohoto důvodu je využito více snímacích

zařízení. Bohužel však vyvstává jiný problém, kterým je synchronizace jednotlivých stop videa z rozlišných zdrojových kamer. Tato skutečnost vyplývá z nemožnosti zajistit spuštění kamer v identický moment. Rozumějte, že je dán, pomocí hlasového příkazu, kamerám příkaz ke spuštění ve stejný okamžik, ale reálně má každá kamera jinou odezvu, než opravdu spustí záznam videa. Jinými slovy, nelze se spolehnout na to, že kamery budou mít své video záznamy synchronizovány, tzn. první snímek z jedné kamery nutně neodpovídá prvnímu snímku z jiné kamery. Mohlo by se zdát, že 1/30 sekundy mezi dvěma snímky není mnoho (vycházíme ze snímkové frekvence 30 FPS), ale lidské mrknutí trvá cca 1/5 sekundy. Pokud by tedy byly kamery mimo synchronizaci o 6 snímků, tak by jedna kamera byla na začátku mrknutí a druhá již na konci. Je tedy vidět, že má smysl řešit synchronizaci video stop a čím lepší přesnosti je docíleno tím lépe.

První možnost, jak synchronizovat je na základě času. Tato synchronizace by však vyžadovala, aby měly kamery velmi přesně seřízený čas (ideálně s přesností na 1/30 sekundy). Bohužel vzhledem ke standardnímu využití kamer *GoPro*, by byla tato vlastnost ve většině případů zcela nepodstatná pro majoritní skupinu uživatelů. Z tohoto důvodu se na synchronizaci času mezi jednotlivými kamerami nelze spolehnout. Další možnost je provést synchronizaci externí. Jedním z možných přístupů je využití externího zdroje světla, který může být například speciálně zabarven. Následně musíme vnější zdroj ve videu rozpoznat. Prvním testovaným přístupem bylo použití externího blesku a následná detekce snímku s maximálním jasem (skokově rozlišným od ostatních snímků v sekvenci). Avšak experimentálně se projevilo, že snímací frekvence 30 FPS není dostatečná, aby každá kamera se 100% pravděpodobností zachytila tento záblesk světla. Externí blesky jsou však stavěny tak, aby byl záblesk světla intenzivní a krátký, proto bylo zapotřebí nalézt jiný zdroj světla pro synchronizaci.

Externí synchronizace pomocí světelného zdroje

Jelikož jsou při natáčení využívány dvě lampy studiových světel, tak byla jedna z lamp využita pro synchronizaci. Konkrétně je před spuštěním natáčení scéna osvětlena pouze jednou lampou. Poté co všechny kamery zahájí záznam videa je rozsvícena druhá lampa. Je-li prozkoumán průběh intenzity jednotlivých snímků v průběhu času, tak lze pozorovat náběh intenzity po rozsvícení druhé lampy. Následně na zvýšení jasu kamera reaguje a intenzita opět klesá, až se stabilizuje, viz obrázek 4.4.



Obrázek 4.4: Průběh jasu v po sobě jdoucích snímcích videa (osa x udává pořadí snímku, osa y jas)

Detekce synchronizačního bodu je tedy ekvivalentní detekci prvního snímku před náběhem. Takový bod lze jednoduše detekovat jako první bod úseku s nejdelším intervalem, kde dochází k nárůstu intenzity v každém po sobě jdoucím snímku. Aby však první snímky použité pro synchronizaci nebyly ovlivněny nárůstem intenzity světla, tak je pomocí klouzavého okénka volitelné velikosti detekován úsek, kde se signál stabilizuje. Jinými slovy se hledá první taková pozice, kde prvky náležící okénku mají rozptyl menší než definovaná hodnota δ. Takové body budeme nazývat body stabilní.

Implementace modulu je poměrně přímočará. Nejprve jsou pomocí *opencv* extrahovány snímky z videa, které jsou převedeny na odpovídající snímky ve stupních šedi. Poté je nad každým snímkem vypočtena průměrná hodnota jasu. Z těchto hodnot je následně sestaven průběh hodnot jasu v jednotlivých snímcích, nad nímž probíhá již popsaná detekce synchronizačních a stabilních bodů. Je nutné vybrat nejdelší interval mezi příslušnými synchronizačními a stabilními body. Přičtením délky, resp. počtu snímků, tohoto intervalu k synchronizačnímu bodu získáme startovací body pro extrakci. Jak je možné nahlédnout, kdyby byl stabilní bod považován za startovací, tak bychom mohli poškodit synchronizaci, jelikož nelze předpokládat stejný průběh jasu na každé kameře. Index pro jednotlivé kamery je uložen do souboru ve formátu `json`, vždy jako pár `{'camera_id': index}`.

```
module: synchronization
output: folder_path
input: folder_path
window: int
variance: float
visualize: bool
```

Listing 5: Konfigurace modulu pro synchronizaci

Konfigurace tohoto modulu, viz listing 5, obsahuje standardní položky jako je `input` a `output`, přičemž výchozí název souboru se synchronizací je *sync-params.json*. Další nepovinné parametry jsou délka klouzavého okénka `window`, která je standardně nastavena na 15, `variance` udávající maximální povolený rozptyl s výchozí hodnotou 0.1 a `visualize` umožňující zobrazení grafů s označenými významnými body (viz obrázek 4.4), který je ve výchozím stavu nastaven na `False`.

4.3.3 Extrakce snímků z videa

Dalším logickým krokem po synchronizaci jednotlivých stop video záznamů je extrakce snímků. Tyto snímky jsou dále použity při rekonstrukci za pomoci nástroje pro fotogrammetrii. Jak vyplývá ze zadání řešeného problému, tak by bylo vhodné umět detekovat snímky, na kterých se model bude pohybovat definovaným způsobem. Pokud jsou do sekvence zahrnuty i snímky, kde se model ještě nepohybuje, nebo už pohyb dokončil, tak je zbytečně plýtváno výpočetním výkonem na rekonstrukci nových modelů, které již nepřináší nové informace.

Detekce pohybu ve videu

Problematika detekce pohybu ve videu je dobře známa a hojně využívána, např. v odvětví bezpečnosti, kdy je snaha o detekci nepovoleného pohybu v oblasti zájmu, nebo při komprimaci videa. Existuje mnoho metod založených na rozdílu snímků, takzvané rozdílové metody, jež jsou základní a mezi které patří např. rozdíl aktuálního snímku a pozadí, rozdíl po sobě jdoucích snímků, dvojitý rozdíl atd. Dalším druhem metod jsou gradientní, např. Optical Flow. Pro detekci pohybu mezi dvěma po sobě jdoucími snímky byla vy-

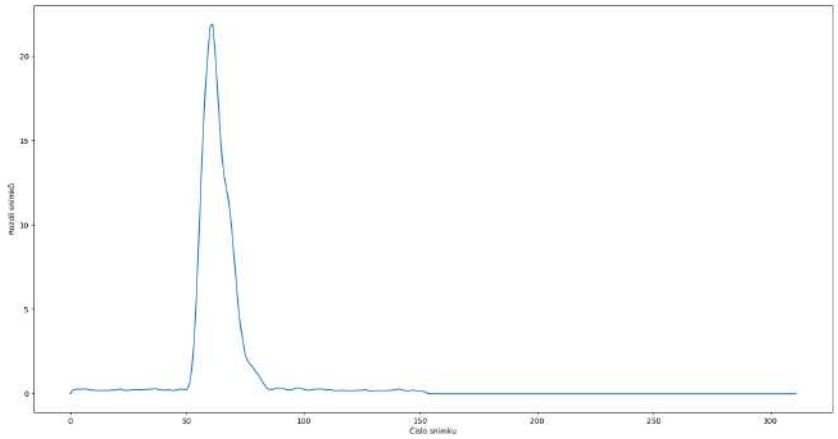
užita metoda rozdílová. Konkrétně se počítá rozdíl po sobě jdoucích snímků.

$$D = |I_{i-1} - I_i|, \tag{4.1}$$

kde I_i je jasová reprezentace i-tého snímku.

Tento přístup je velmi náchylný na změnu osvětlení snímané scény, proto je nutné dbát na konzistentní osvětlení při pořizování video záznamů. Tento požadavek, konzistentní osvětlení, však vyžaduje i synchronizace jednotlivých video stop, kterou by skoková změna (kromě té synchronizační) mohla poškodit.

Implementace vychází ze sestrojení průběhu rozdílů po sobě jdoucích snímků, resp. rozdílu jejich průměrné hodnoty jasu D, viz graf 4.5. Následně je v takové funkci nalezeno x největších zaznamenaných rozdílů, resp. pohybů. Jak je vidět, je zapotřebí převést snímky na jasovou reprezentaci, tzn. do odstínů šedi a vypočítat jejich rozdíl v absolutní hodnotě. Při výpočtu rozdílu je třeba dbát na datový typ, ve kterém jsou hodnoty jasu uchovány (běžně `uint8`), tak aby při výpočtech nepřetekl. Jak je možné nahlédnout, každá kamera může detekovat rozdílný průběh pohybu, jelikož se na model dívá z jiného úhlu. Z tohoto důvodu jsou průběhy rozdílů po sobě jdoucích snímků sloučeny do jednoho (pomocí jednoduchého součtu). Až nad tímto součtem průběhů je detekce prováděna. Dále z charakteru pohybu plyne, že je hledán souvislý interval po sobě jdoucích snímků. Proto je hledání x největších zaznamenaných rozdílů převedeno na nalezení maxima a následné rozšiřování intervalu do sousedních snímků, tak aby vždy byl vybrán adept s větším rozdílem. Opět je využito *opencv*.



Obrázek 4.5: Průběh rozdílů jasů (osa x udává pořadí snímku v sekvenci a osa y rozdíl jasů)

Konfigurace tohoto modulu, viz listing 6, obsahuje standardní položky jako je `input` a `output`. Dalším povinným parametrem je `sync-params`, který udává cestu k synchronizačnímu souboru a `count` obsahující počet snímků pro extrakci.

```
module: extraction
output: folder_path
input: folder_path
sync-file: file_path
count: int
```

Listing 6: Konfigurace modulu pro extrakci

4.4 Rekonstrukce

Jsou-li k dispozici snímky obličeje modelu, které jsou synchronizovány (viz sekce 4.3), tak je dalším logickým krokem v procesu extrakce statického modelu lidského obličeje ze stereo/multiview videa získání 3D modelu na základě těchto snímků. Tento proces je dále nazýván rekonstrukcí a je založen na fotogrammetrii, viz sekce 2. Za tímto účelem je využit fotogrammetrický software *Meshroom*. Aby však bylo možné provést kvalitní rekonstrukci, je zapotřebí kalibrovat vnitřní parametry kamer (sekce 4.4.1), resp. získat je-

jich hodnoty (ohnisková vzdálenost, optical center atd.). Po samotné rekonstrukci 3D modelu přichází na řadu další série úprav modelu, která je v práci uváděna pod názvem mesh-filtering (sekce 4.4.3) a zabývá se vylepšováním získaného modelu.

4.4.1 Kalibrace kamer

Jak již bylo zmíněno v teoretické části zabývající se fotogrammetrií, reálná snímací zařízení trpí distorzí snímků. Distorzi si lze představit jako systematickou chybu každého snímku, např. ve skutečnosti rovné čáry jsou na snímku více či méně zahnuté. Používané kamery *GoPro Hero Silver 7* trpí distorzí nazývanou rybí oko (fisheye). Taková distorze může být matematicky popsána. Na základě znalosti pozice bodů daného vzoru v souřadnicích snímku a v souřadnicích reálného světa lze tuto distorzi odstranit, resp. získat vnitřní parametry kamer. Standardně jsou pro kalibraci používané dva vzory, šachovnicový a kruhový. V této práci je použit vzor šachovnicový.

Princip kalibrace

Celá metoda stojí na znalosti pozic bodů šachovnicového vzoru. Souřadnice v snímku získáme jednoduše detekcí bodů šachovnice, kde se jednotlivá políčka (černá a bílá) setkávají. Získat reálné pozice těchto bodů pouze ze znalosti snímků však není možné. Z tohoto důvodu je zaveden předpoklad, že šachovnice je pevně umístěna v prostoru a kamera se pohybuje okolo. Jinými slovy, je stanoveno, že pro každý bod šachovnice je $Z = 0$ (jelikož je šachovnice plocha, tak tento předpoklad neodporuje fyzikální realitě). Následně je levý horní roh prohlášen za počátek souřadného systému, tzn. $(0, 0, 0)$. Poté lze dalším bodům přiřadit pozice $(0, 1, 0)$, $(0, 2, 0)$ atd. (postupně zleva doprava a shora dolů). Jak lze nahlédnout, byl definován souřadný systém v jednotkách velikosti jednoho políčka šachovnice, nikoliv v reálných jednotkách. Pokud je známa šířka, resp. výška, políčka šachovnice, tak je možné definovat souřadný systém v reálných jednotkách, např. v milimetrech. Na základě takto definovaných pozic bodů jak ve snímku, tak v reálném světě, lze vypočítat potřebné koeficienty popisující distorzi snímku a tuto chybu odstranit, nebo pouze vypočítat vnitřní parametry kamery. [12]

Implementace vychází z již zmíněných poznatků a je založena na využití knihovny *opencv*. Konkrétně jsou využity metody `findChessboardCorners` a `calibrateCamera`. Tento postup je poměrně přímočarý, proto je vhodné pro více detailů nahlédnout do implementace tohoto modulu, popř. na návody a dokumentaci *opencv*.

```
module: calibration
output: folder_path
input: folder_path
show-chessboars: bool
```

Listing 7: Konfigurace modulu pro kalibraci

Konfigurace , viz listing 7, obsahuje standardní vstup a výstup, přičemž vstupem je složka obsahující množinu fotek pro kalibraci (zachycujících šachovnicový vzor pod různými úhly). Výstupní soubor má název `calib-paramas.json`. Parametr `show-chessboars` umožňuje vykreslit detekci šachovnicového vzoru do složky `chessboars`, ve výchozím stavu je tento parametr nastaven na hodnotu `False`.

4.4.2 Meshroom

Jak již bylo uvedeno v teoretické části zabývající se fotogrammetrií, pro rekonstrukci modelu je použit nástroj s názvem *Meshroom*, viz sekce 2.2.2, který staví nad frameworkem *AliceVision*. *Meshroom* je de facto pouze grafickou nástavbou, jež umožňuje pohodlné používání pro široké spektrum uživatelů. Avšak na pozadí se spouští jednotlivé moduly právě frameworku *AliceVision* a proto je poměrně snadné tento proces napodobit pomocí vlastních skriptů. Aby byla tato činnost ještě jednodušší, tak na pozadí *Meshroom* běží konzole, pomocí níž lze snadno zjistit příkazy nutné pro spuštění jednotlivých fází.

Meshroom funguje na bázi modulárního programování, tzn. jsou nadefinované jednotlivé bloky, které lze vzájemně propojit do takzvaného grafu zpracování. Předdefinovaných modulů existuje celá řada, viz dokumentace [20]. Tato funkcionalita umožňuje poměrně snadné testování nastavení parametrů tak, aby byl získán co nejlepší výsledek. Konkrétně je možné vytvořit z jednoho modulu několik cest, jež se liší v času potřebném na výpočet, nebo kvalitě. Na základě této struktury lze včas detekovat, zda proces směřuje ke zdárnému výsledku a v případě, že ne, rychle zamítnout celé nastavení, resp. ukončit výpočet. Toto větvení také umožňuje využívat již spočtených výsledků a nepočítat je s každou změnou parametrů.

Používané sestavení komponent

V této sekci bude uvedeno sestavení komponent frameworku *AliceVision*, které je použito v modulu pro rekonstrukci.

1. `aliceVision_cameraInit.exe`
2. `aliceVision_featureExtraction.exe`
3. `aliceVision_imageMatching.exe`
4. `aliceVision_featureMatching.exe`
5. `aliceVision_incrementalSfM.exe`
6. `aliceVision_utils_sfmTransform.exe`
7. `aliceVision_prepareDenseScene.exe`
8. `aliceVision_depthMapEstimation.exe`
9. `aliceVision_depthMapFiltering.exe`
10. `aliceVision_meshing.exe`
11. `aliceVision_meshFiltering.exe`
12. `aliceVision_texturing.exe`

Většina těchto modulů frameworku byla zkoumána v teoretické části. Nyní budou popsány zejména změny vzhledem k výchozímu nastavení grafu zpracování v *Meshroom*.

V první fázi (camera init (1)) jsou využity vnitřní parametry kamery *GoPro Hero 7 Silver*, které byly získány z modulu pro kalibraci. Další změnou je využití pozice kamer první rekonstrukce z páté fáze (structure from motion (5)) v dalších rekonstrukcích náležících jedné sekvenci. Konkrétně je v páté fázi (structure from motion (5)) vytvořen soubor s názvem `viewsAndPoses.sfm`, který obsahuje umístění jednotlivých kamer. Tyto pozice jsou dále použity v následujících rekonstrukcích v dané sekvenci. Specificky se jedná o jejich zkopírování do souboru s názvem `cameraInit.sfm`, který vzniká v první fázi (camera init(1)). Oba tyto soubory jsou ve formátu `json`. Tímto přístupem je eliminován rozdíl v měřítku jednotlivých modelů ze stejné sekvence. Pokud by tento krok nebyl proveden, tak by registrace nutně selhala.

Ve druhé fázi (feature extraction (2)) je použita největší možná kvalita extrakce deskriptorů, tzn. `ultra`. Byť je tato možnost výpočetně náročná,

tak je to pro 10 snímků stále doba přijatelná. Naopak jelikož klasicky rekonstruujeme z řádově desetinásobně větších množin snímků, tak toto nastavení částečně dohání ztrátu kvality. Dále je v této fázi vypnuto vynucení výpočtu na CPU.

Pro všechny moduly využívající deskriptory význačných rysů jsou používány deskriptory typu `sift`, `akaze` a `cctag3`. V případě potřeby je možné v pozdějších modulech nepoužívat všechny deskriptory, které byly v krocích předchozích. Opačně to však neplatí, tzn. každý předchozí modul v grafu zpracování musí používat alespoň všechny deskriptory modulů následujících.

V šesté fázi (sfm transformation (6)) je ukotven celý model vzhledem ke specifikované kameře. V práci je použita prostřední (přední) výše umístěná kamera. Tímto krokem je docíleno blízkosti modelů následujících v sekvenci, a tudíž splněny podmínky pro lokální registraci ICP.

Sedmá fáze (depth map estimation (7)) je rozdělena do dalších čtyř fází, kde se postupně upravují parametry `rangeStart` a `rangeSize`. Stejně je tomu tak v defaultním nastavení Meshroom. Na tomto místě je tato informace poskytnuta pouze z důvodů srozumitelnosti.

V desáté fázi (mesh filtering (10)) je vypnuto vyhlazování výsledného trojúhelníkového modelu, jelikož není žádoucí ztráta informace s tím spojená.

Další možná přenastavení

Pro zlepšení výsledku je možné nastavit další parametry, viz [3]. Tyto parametry nejsou v současné verzi procesu používány, a to z důvodů nefunkčnosti, nebo minimálního vylepšení za cenu výrazně delšího výpočetního času.

- `GuidedMatching` umožňuje ve druhé fázi (feature extraction (2)) přidat druhý průchod hledáním význačných rysů. Tento druhý průchod používá první výsledek tohoto modulu jako omezení. Není využito, jelikož se v aktuální verzi softwaru jeví jako nefunkční.

- `minTrackLenght` je možné zvětšit v páté fázi (structure from motion (5)) a tím vynutit užití robustnějších shod. Není využito, jelikož nepřineslo znatelné zlepšení.

- `downscale` lze nastavit na hodnotu 1 a tím v sedmé fázi (depth map estimation (7)) získat lepší hloubkovou mapu. Není použito, jelikož se čas výpočtu prodlouží přibližně 4× a výsledné rekonstrukce v testovacím datasetu nebyly výrazně lepší.

```
module: reconstruction
output: folder_path
input: folder_path
transform-cam: string
quality: {low | normal | high | ultra}
alice-vision-bin: folder_path
```

Listing 8: Konfigurace modulu pro rekonstrukci

Implementace tohoto modulu je opět poměrně přímočará. Jedná se tedy o spouštění modulů frameworku *AliceVision* se zapojením funkcionality `safe-points`. Pro více detailů je vhodné nahlédnout do kódu příslušného modulu. Jak je vidět z možných parametrů konfigurace tohoto modulu, tak jsou vzhledem ke komplexnosti možností frameworku *AliceVision* jen velmi omezené. Pokud uživatel potřebuje změnit vnitřní parametry rekonstrukce, tak je nutné upravit metodu `__meshroom_reconstruction`, v které jsou jednotlivé parametry použité pro moduly frameworku *AliceVision* poměrně snadno přístupné a čitelné.

Konfigurace , viz listing 8, obsahuje standardní vstup a výstup, přičemž vstupem je složka obsahující množinu složek (standardně očíslovaných) s množinou souvisejících fotek (fotky standardně pojmenovány dle kamery původu). Do výstupní složky se vytvoří obdobná adresářová struktura, jako je ve složce výstupní, která obsahuje interní složky rekonstrukce *AliceVision*, výsledný model s texturou a bez textury. Dále je ve výstupní složce vytvořen adresář `Sequence`, který obsahuje sekvenci pojmenovanou `mesh-xxx.obj`, kde xxx udává pořadí rekonstrukce v rámci sekvence. Parametr `transform-cam` udává název fotky (standardně tedy identifikátor kamery původu) pro ukotvení rekonstrukce v prostoru. Volitelný parametr `quality` udává kvalitu při výpočtu deskriptorů. Vyšší kvalita není vždy žádoucí, nejen že výpočetní čas rychle narůstá, ale na určitých vstupních datech může při vynucování vysoké kvality selhat pátý krok (structure from motion (5)). Ve výchozím stavu je kvalita nastavena na hodnotu `ultra`. Posledním argumentem je `alice-vision-bin` obsahující cestu ke složce s binárními soubory frameworku *AliceVision*.

4.4.3 Mesh-filtering

V práci je naimplementováno několik modulů umožňujících práci s rekonstruovaným modelem. Konkrétně se jedná o vyřezávání modelu obličeje z celé scény, spuštění Meshlab skriptů umožňujících libovolnou práci s trojúhelníkovým modelem poskytovanou *Meshlabem* a jako poslední filtrace trojúhelníků modelu. Trojúhelníkový model je v práci reprezentován třídou `mesh`, která obsahuje datovou strukturu `Corner Table`.

Clipping

Mnoho fotogrammetrických nástrojů umožňuje rekonstrukci oblasti zájmu na základě masek vstupních snímků. Bohužel tato funkcionalita není v současnosti dostupná v *Meshroom*. Proto byl za účelem získání oblasti zájmu implementován jednoduchý modul, který provádí ořezání rekonstruovaného modelu, tak aby byl získán pouze model obličeje. Zbytek scény, který je mimo střed zájmu celého procesu, je odříznut.

Jak lze nahlédnout, zařízení pro zavěšení kamer definuje válcovou oblast zájmu. Vše mimo tento tvar (válec) je oříznuto. Je nutné si dát pozor na skutečnost, že při tomto ořezávání mohou vzniknout nesouvislé oblasti. Pro více detailů je vhodné nahlédnout do kódu příslušného modulu.

Konfigurace , viz listing 9, obsahuje standardní vstup a výstup, přičemž vstupem je složka obsahující množinu trojúhelníkových modelů pro oříznutí. Do výstupní složky se uloží oříznutý model pod stejným názvem. Parametry `center`, `height` a `radius` definují rozměry a umístění ořezávajícího válce. Ve výchozím stavu jsou tyto parametry nastaveny na $center = [0.0, 0.0, -1.5]$, $height = 2.0$ a $radius = 0.8$. `Filter-unnconected` umožňuje po ořezávání odfiltrovat všechny části, které nejsou spojeny s hlavním (největším) modelem. Ve výchozím stavu je filtrace zapnuta. Poslední parametr `format` udává formát ořezaných modelů a je ve výchozím stavu nastaven na formát `obj`.

Meshlab skript

Tento modul slouží jako prostředník pro jednoduché spuštění skriptů v *Meshlabu*, které umožňují velmi pokročilé možnosti filtrování trojúhelníkových modelů. Vychází z funkcionality *Meshlabu*, který umožňuje vytvoření a uložení skriptu ve formátu `mlx`. Takový skript lze vytvořit v grafickém rozhraní *Meshlab*u a následně spustit v *Meshlab serveru*. Ukázku takového skriptu lze vidět ve složce `Filters`. Je možné, že v nadcházejících verzích *Meshlabu* se některé filtry přejmenují a vyvstanou potíže. Tato skutečnost by měla být

```
module: mesh-filtering
submodule: clipping
output: folder_path
input: folder_path
center: [float, float, float]
height: float
radius: float
filter-unconnected: bool
format: {obj | ply}
```

Listing 9: Konfigurace modulu pro ořezávání

```
module: mesh-filtering
submodule: meshlab-script
output: file_path
input: file_path
script: file_path
meshlab-folder: folder_path
```

Listing 10: Konfigurace modulu pro meshlab skript

zřejmá z výpisu programu a snadno odstranitelná přegenerováním skriptu s novým filtrem, který odpovídá funkci původní. Pro více detailů je vhodné nahlédnout do kódu příslušného modulu, popř. dokumentace *Meshlabu*.

Konfigurace , viz listing 10, obsahuje standardní vstup a výstup. Parametr `script` uchovává cestu k uloženému *Meshlab* skriptu a `meshlab-folder` udává cestu k složce obsahující spustitelný soubor *Meshlab serveru*.

Filtrace trojúhelníků

Tento modul slouží k filtraci nekvalitních trojúhelníků, které vznikly při rekonstrukci trojúhelníkového modelu. Takové trojúhelníky vznikají v místech, kde bývá rekonstrukce špatná. Kvalita trojúhelníků je hodnocena na základě délky jeho hran. Přesněji řečeno je možné nastavit percentil délky hrany, při kterém bude trojúhelník považován za špatný, resp. dobrý. Zavedení percentilu, místo konkrétní délky, je z důvodů neznalosti měřítka rekonstruovaného modelu. Jinými slovy, je známo procento nejhorších trojúhelníků jež je žá-

```
module: mesh-filtering
submodule: triangles-filtering
output: folder_path
input: folder_path
percentile: float
format: {obj | ply}
```

Listing 11: Konfigurace modulu pro filtraci trojúhelníků

doucí odfiltrovat, např. 1%, ale konkrétní délka hrany špatného trojúhelníku záleží na měřítku modelu. Pro více detailů je vhodné nahlédnout do kódu příslušného modulu.

Konfigurace , viz listing 11, obsahuje standardní vstup a výstup, přičemž vstupem je složka obsahující množinu trojúhelníkových modelů pro odfiltrování špatných trojúhelníků. Do výstupní složky se uloží přefiltrovaný model pod stejným jménem. Parametr `percentile` udává percentil délek hran trojúhelníků, které jsou považovány za dobré, tzn. chceme je zachovat. `Format` udává formát přefiltrovaných modelů a ve výchozím stavu je nastaven na formát `obj`.

4.5 Registrace

Předposledním krokem procesu extrakce statického modelu lidského obličeje ze stereo/multiview videa je registrace modelů na sebe a to ať už se jedná o rigidní (sekce 4.5.1), nebo nerigidní (sekce 4.5.2) registraci. Tento krok si klade za cíl spojení jednotlivých částí modelu, které nemusí být nutně totožné svojí topologií. Metody vychází z teoretických poznatků uvedených v sekci 3. Následuje popis použitých nástrojů a technik.

4.5.1 Rigidní registrace

Pro rigidní (tuhou) registraci je využit algoritmus ICP. Důvodem pro použití ICP oproti zmíněnému RANSAC je blízkost zarovnávaných modelů, která vyplývá ze způsobu jejich získávání. Jak již bylo výše popsáno, ICP je algoritmus velmi dobře známý a často používaný. Z uvedených poznatků je zřejmé, že existuje mnoho dobře otestovaných implementací, a tudíž by bylo zbytečné se snažit o implementaci vlastní. Proto byla použita knihovna

```
module: alignment
submodule: rigid
output: folder_path
input: folder_path
target: int
```

Listing 12: Konfigurace modulu pro rigidní registraci

open3d, která mimo jiné obsahuje i algoritmus ICP.

Zarovnávání probíhá přes sousedy. Jinými slovy, model a je vždy zarovnán na svého souseda b, který je blíž cílovému modelu. Následně je výsledná transformace použita pro zarovnání všech modelů v sekvenci před a, které už byly v předchozích iteracích zarovnány na model a, za účelem registrace na model b. Tímto přístupem algoritmus zarovná všechny modely na cílový. Díky dobrým počátečním odhadům pro ICP je konvergence rychlejší, než kdyby nebyl odhad předem známý a každý model by musel projít přes všechny sousedy směrem k cílovému modelu bez tohoto odhadu.

Konfigurace , viz listing 12, obsahuje standardní vstup a výstup, přičemž vstupem je složka obsahující množinu trojúhelníkových modelů pro registraci. Do výstupní složky se uloží registrovaný vstupní (zdrojový) model pod stejným jménem. Parametr `target` udává kolikátý vstupní model bude použit jako cílový, tzn. model na který se ostatní zarovnávají.

4.5.2 Nerigidní registrace

Pro nerigidní (netuhou) registraci je využita vlastní implementace vytvořená na základě [4], která vznikla v rámci předmětu KIV/ZPOS na Západočeské univerzitě. Tato implementace byla vybrána z několika důvodů. Hlavním argumentem byla znalost implementace a možnost její úpravy. Dalším důvodem bylo otestování a případné vylepšení implementace, tak aby se dal tento program v budoucnosti bezproblémově používat. Tento nástroj byl vytvořen v programovacím jazyku *C#* s využitím komerční knihovny pro lineární algebru s názvem *BlueBit*. V následujícím textu budou popsány části, z kterých se celý program sestává.

Hledání geodetických vzdáleností

První modul má za úkol nalezení geodetické vzdálenosti z vrcholu A do všech vrcholů daného modelu do vzdálenosti d. Za tímto účelem byla použita metoda Fast Marching [15], jejíž implementace byla k dispozici. Jelikož implementace této metody není náplní této práce, tak zde dále nebude rozepisována. Pro teoretické základy metody viz [15], pro implementační detaily je možné nahlédnout do příslušného kódu.

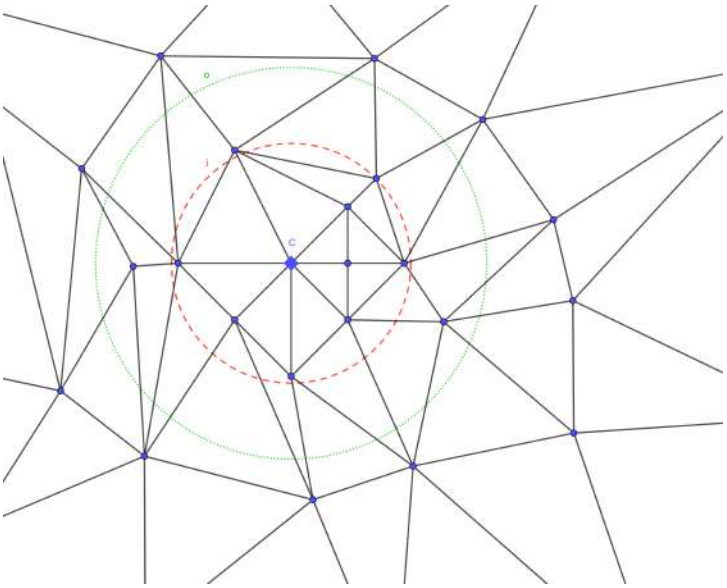
Deformační graf

Deformační graf je nejprve sestaven a následně jsou vyplněny všechny potřebné struktury, tak aby splňoval definici (sekce 4.5.2), tzn. každý vrchol trojúhelníkového modelu má informaci o čtyřech nejbližších vrcholech deformačního grafu a zároveň každý vrchol deformačního grafu zná všechny vrcholy, pro které je jedním ze čtyř nejbližších.

Původní implementace musela být přepsána, jelikož byla úzkým hrdlem celého procesu nerigidní registrace. Bylo však dodrženo původní rozhraní, tak aby byly co nejvíce omezeny problémy spojené s integrací do již stávajícího systému.

Sestavení deformačního grafu , viz obrázek 4.6, probíhá následovně. Nejprve je náhodně zvolen startovací vrchol C, z kterého jsou vyhledány všechny body do vzdálenosti r_o (v obrázku 4.6 tečkovaná kružnice o). Tyto body tvoří cluster s vrcholem v bodě C, který je zároveň prohlášen za vrchol deformačního grafu. Dále je sestrojeno jádro clusteru, které obsahuje všechny body se vzdáleností od bodu C, která je menší nebo rovna r_i (v obrázku 4.6 čárkovaná kružnice i). Jelikož poloměr r_i je roven ideální vzdálenosti vrcholů deformačního grafu, tak není možné, aby tyto body byly dalšími vrcholy deformačního grafu. Proto jsou do fronty adeptů na vrchol deformačního grafu umístěni sousedi těchto vrcholů, kteří nepatří do jádra clusteru. Pro každého adepta z fronty se postupuje stejně, pokud mezitím nebyl použit nebo zařazen do jádra některého z vrcholů deformačního grafu.

Nalezení nejbližších uzlů deformačního grafu staví na již nalezených clusterech bodů. Jednoduchým prozkoumáním clusterů zjistíme nejbližší vrcholy deformačního grafu pro každý vrchol zdrojového trojúhelníkového modelu. Jelikož je cluster větší než požadovaná vzdálenost mezi vrcholy deformačního grafu, tak je pravděpodobné (záleží na poměru poloměru clusteru r_o a jádra clusteru r_i), že každý bod bude patřit do více clusterů. Jak je



Obrázek 4.6: Konstrukce doformačního grafu

možné vidět, tak pravděpodobnost tohoto předpokladu roste se zvětšujícím se poloměrem clusteru (r_0).

Po této rychlé inicializaci přijde na řadu pomalejší, ale finální doplňování do požadovaného počtu nejbližších vrcholů deformačního grafu. Toto dohledávání je založeno na prohledávání sousedů. Konkrétně se vychází z myšlenky, že existuje v okolí takový bod b, který má požadovaný počet nejbližších vrcholů deformačního grafu. Nejprve je v sousedství bodu a nalezen právě takový bod b. Následně je z bodu a zahájeno prohledávání do vzdálenosti určené součtem vzdálenosti z bodu b k jeho nejvzdálenějšímu nejbližšímu vrcholu deformačního grafu a vzdálenosti z bodu b do bodu a.

Pro rychlý výpočet je nutné nalézt správný poměr mezi r_i a r_o. Používaný poměr v práci je 1 : 2. Urychlení staví na myšlence, že je lepší prohledávat do větší vzdálenosti u vrcholů deformačního grafu, kterých je řádově méně v porovnání s vrcholy trojúhelníkového modelu, než prohledávat, byť do menší vzdálenosti, u všech vrcholů trojúhelníkového modelu.

Trasování paprsku

Ray tracing je jedna z klasických úloh počítačové grafiky, proto je možné najít nějakou vysoce optimalizovanou knihovnu a přidat ji do stávajícího řešení. V této práci jsme se však vydali cestou vlastní implementace i za cenu ztráty rychlosti.

Jelikož je nutné najít mnoho průsečíků paprsků s daným trojúhelníkovým modelem, tak je potřeba využít nějaké urychlovací techniky. Konkrétně je

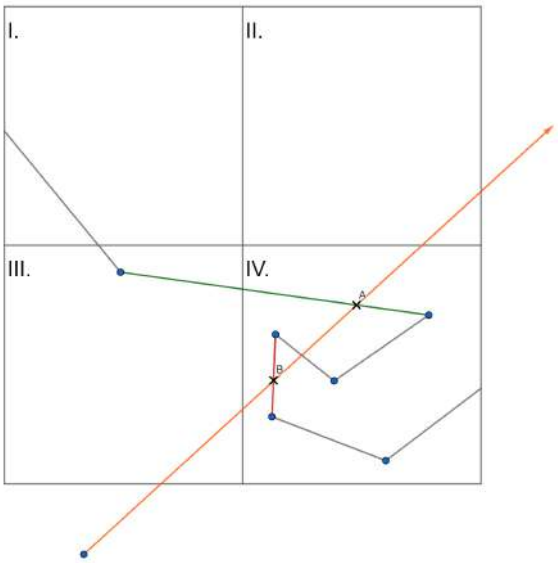
využita technika dělení prostoru, aby bylo možné testovat průsečík paprsku pouze s omezeným množstvím trojúhelníků.

Jako technika rozdělení prostoru byla zvolena prostorová „mřížka“ (volumetric grid). U této techniky je nutné vyřešit z kolika buněk se bude „mřížka“ sestávat. Pokud bude zvolen příliš velký počet buněk, tak metoda trasování paprsku stráví příliš mnoho času hledáním cesty v „mřížce“. Naopak bude-li počet buněk příliš malý, tak bude nutné testovat zbytečně mnoho trojúhelníků, a to vyústí v dlouhý výpočetní čas. Další problematiku, kterou je zapotřebí brát v úvahu, je nutnost dbát na numerickou přesnost výpočtů ve většině operací nad „mřížkou“, tzn. k porovnávání používat epsilon test.

Pohyb „mřížkou“ je založen na hledání průsečíků aktuální buňky s paprskem. Podle typu stěny, kterou paprsek protíná, je určen směr postupu (tzn. protíná-li pravou stěnu pokračuj doprava atd.). Je možné, že směrů pro další postup bude nalezeno více (rohy a spojnice hran). Opět je nutné vzít v úvahu numerické chyby při výpočtu průsečíků.

Postup přiřazení jednotlivých trojúhelníků příslušným buňkám „mřížky“ je dalším krokem, který musí být vyřešen před finálním hledáním korespondencí. Tento úkol sám o sobě není zcela triviální. V nástroji jsou připraveny dvě možnosti. První, lehčí, způsob je přidání trojúhelníku do všech buněk „mřížky“, které incidují s boundig boxem přidávaného trojúhelníku. Tato možnost je zcela dostačující je-li velikost trojúhelníků zanedbatelná vzhledem k velikosti buněk. Druhou možností je vložení hran trojúhelníku do buněk „mřížky“ následované záplavovým postupem, který nalezne zbývající buňky, s kterými trojúhelník inciduje.

Přidání hran trojúhelníku je velmi jednoduché, je to stejný systém jako pohyb paprsku „mřížkou“. Samotný záplavový postup se sestává z vybrání bodu uvnitř trojúhelníku a postupem do okolních buněk, do kterých se trojúhelník šíří, dokud nenarazíme na již přidanou hranici. Nalezení buněk, kam se šíří trojúhelník je poměrně snadné. Pomocí znaménkových testů vůči vrcholům aktuální buňky jsou zjištěny směry kam se šíří rovina určená přidávaným trojúhelníkem. Tento způsob by byl nutný, z hlediska výkonnosti, pokud by trojúhelníky nebyly přibližně stejně velké a jejich velikost by nebyla zanedbatelná vzhledem k velikosti buněk.



Obrázek 4.7: Speciální případ

Nalezení průsečíku s trojúhelníkem vychází z barycentrické rovnice 4.2, přičemž postup je následovný.

$$\alpha * \vec{A} + \beta * \vec{B} + \gamma * \vec{C} = \vec{X}, \tag{4.2}$$

kde $\alpha + \beta + \gamma = 1$ a $\vec{A}$, $\vec{B}$, $\vec{C}$ jsou vrcholy trojúhelníku.

Při vstupu do buňky jsou vyhledány všechny průsečíky s trojúhelníky incidujícími s touto buňkou, přičemž je vybrán nejbližší průsečík vzhledem k počátku paprsku. Pokud není žádný průsečík nalezen, tak se postupuje do další buňky. Naopak existuje-li průsečík, tak je nutné otestovat, zda nalezený průsečík leží v aktuální buňce. V případě že neleží, tak musí být otestovány všechny trojúhleníky v buňce, do které nalezený průsečík náleží, viz obrázek 4.7.

Coarse Alignment

Tato část nerigidní registrace slouží k zarovnání větších změn zdrojového modelu (S) na model cílový (T). Pro správnou funkci této fáze je nutné, aby na sebe byly modely alespoň nahrubo zarovnány, např. za pomoci ICP algoritmu. Teoretický popis metody viz sekce 3.1. Jak lze nahlédnout, v tomto kroku je využit již dříve popsaný ray traycing a deformační graf. Pro hledání nejbližších bodů byl využit KD strom.

Optimalizace energie je poměrně přímočará, tzn. nejprve je nutné provést výpočet derivací reziduí, viz příloha sekce 7.2. Výpočet těchto derivací byl

proveden za pomocí specializovaného softwaru, konkrétně se jednalo o *Matlab* a jeho funkcionality pro symbolické výpočty. Bohužel výsledek výpočtu byl velmi komplikovaný výraz a bylo nutné jej ručně zapsat do kompaktnější formy, pokud možno s použitím vektorové notace. Výsledné vztahy viz příloha sekce 7.2. Z důvodů důležitosti správného výsledku byly implementovány testy, opět v *Matlabu*, které ověřily správnost získaných vztahů. Tyto testy pomocí symbolické manipulace spočítaly pro výsledné vztahy, získané pomocí *Matlabu* (tudíž s největší pravděpodobností správné výsledky), hodnoty pro různě inicializované proměnné. Následně proběhlo porovnání výsledků s výpočtem založeným na ručně získaných vztazích (ruční výpočet je velmi citlivý na chyby). Dalším faktorem hovořícím o velké pravděpodobnosti správnosti výpočtů je fakt, že ke stejnému výsledku došli dva nezávislí pracovníci.

Minimalizace energie probíhá pomocí Gauss-Newtonova algoritmu. Tento postup vyžaduje znalost matice $\boldsymbol{J}_r^T \boldsymbol{J}_r$, jejíž výpočet probíhá následovně. Nejprve je inicializována na nulovou matici odpovídajícího rozměru. Dále je vypočten outer produkt vektoru parciálních derivací (řádka $\boldsymbol{J}_r$) a tento výsledek přičten k hledané matici. Jak lze nahlédnout, tak není zapotřebí ukládat jacobiho matici, a navíc je takový výpočet mnohem rychlejší než násobení matic.

Dalším problémem, který bylo nutné vyřešit, bylo selhávání výpočtu pro špatné modely, resp. pro modely s ořezanými korespondencemi. Tento problém byl způsoben tím, že pokud byly oříznuty korespondence tak, že derivace daného rezidua byla vždy nulová, pak výsledná matice $\boldsymbol{J}_r^T \boldsymbol{J}_r$ byla singulární. Proto je výpočet omezen pouze na dobré korespondence a po každém cyklu Gauss-Newtonova algoritmu je pomocí lineárního blendingu vypočtena nová pozice pro korespondence, jež jsou považovány za špatné. Dále byla odstraněna podmínka značící špatnou korespondenci na základě podobnosti normál, jelikož jsou rekonstruované modely často zatíženy šumem a bylo by příliš mnoho špatných korespondencí.

Jak je možné nahlédnout, tak matice $\boldsymbol{J}_r^T \boldsymbol{J}_r$ je řídká, a proto je vhodné tohoto poznatku využít při řešení soustavy rovnic v Gauss-Newtonově algoritmu. Bylo testováno několik knihoven pro výpočty a to: *Meta Numerics*, *Math.NET* a *BlueBit*. Poslední zmíněná knihovna, *BlueBit*, dosahuje nejlepších výsledků vzhledem k rychlosti výpočtů. Z tohoto důvodu byla do výsledného nástroje zařazena. Avšak i tato knihovna trpí jistým množstvím nedostatků, a proto bylo by vhodné najít stejně výkonnou knihovnu, která je lépe odladěná.

Fine-Scale Linear Alignment

Tato část nerigidního zarovnání slouží k zarovnaní jemných detailů zdrojového modelu (S) na model cílový (T). Pro správnou funkci tohoto typu registrace je nezbytné, aby modely, které mají být detailně zarovnány, byly na sebe již co nejlépe registrovány. Teoretický popis metody viz sekce 3.2.

Optimalizace energie

Jak je možné nahlédnout, pro vyřešení této úlohy je zapotřebí implementace ray traycingu a optimalizace zapsané energie. Minimalizace energie je řešena pomocí metody nejmenších čtverců.

$$\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{y} = \boldsymbol{A}^T\boldsymbol{b}$$

Je nutné vzít v úvahu, že matice $\boldsymbol{A}^T$ (resp. $\boldsymbol{A}$) je řídká. Tato vlastnost je velmi důležitá pro urychlení nalezení řešení metodou nejmenších čtverců. Z těchto důvodů bylo implementováno speciální násobeni matic $\boldsymbol{A}^T\boldsymbol{A}$ (pouze ze znalosti hodnot matice $\boldsymbol{A}$), tak aby výsledná matice byla řídká a samotné násobení rychlé (je dostačující jeden průchod maticí $\boldsymbol{A}$). Pro výpočet nejmenších čtverců byly uvažovány tři knihovny: *Meta.Numerics*, *Math.Net* a *BlueBit*. První zmíněná knihovna umožňuje snadné řešení nad řídkou maticí, kdežto druhá knihovna nabízí iterativní solvery soustavy a lepší metody pro husté matice. Obě tyto knihovny jsou volně dostupné. Nakonec byla použita komerční knihovna *BlueBit*, jelikož dosahovala lepších výsledků.

Omezení nerigidní registrace

Poslední poznámkou k implementaci nerigidní registrace jsou požadavky, které musí splňovat vstupní trojúhelníkové modely. Tyto modely musí být manifoldy, nesmějí mít trojúhelníky s nulovým povrchem (zero-area faces) a modely by se měly sestávat z jedné souvislé trojúhelníkové sítě. Pokud bude model obsahovat více komponent, které budou dostatečně malé, tak selže tvorba deformačního grafu, jež nebude schopna v souvislé okolní oblasti nalézt dostatečný počet uzlů deformačního grafu se stanovenou vzdáleností.

Konfigurace , viz listing 13, obsahuje standardní vstup a výstup, přičemž vstupem je složka obsahující množinu trojúhelníkových modelů pro registraci. Do výstupní složky se uloží registrovaný vstupní (zdrojový) model pod stejným jménem. `Target` je cesta k cílovému modelu, který je ve výchozím stavu nastaven na prostřední model celé sekvence, tzn. prostřední model vstupní složky. Parametr `aligner-folder` udává složku s nerigidním

```
module: alignment
submodule: non-rigid
output: folder_path
input: folder_path
target: file_path
aligner-folder: folder_path
sequence: bool
```

Listing 13: Konfigurace modulu pro nerigidní registraci

registračním nástrojem (spustitelný soubor). Poslední parametr `sequence` je ve výchozím stavu nastaven na `False`. Pokud je tento parametr nastaven na hodnotu `True`, tak registrace probíhá přes sousedy až na cílový model.

4.6 Remeshing

Poslední část procesu extrakce statického modelu lidského obličeje ze stereo/multiview videa se zabývá spojením na sebe registravaných modelů a následným remeshingem. Tyto části jsou opět implementovány jako moduly. Jelikož jejich implementace je buď velmi přímočará, nebo využívá již výše popsaných modulů, tak budou popsány pouze stručně.

4.6.1 Spojení modelů

V tomto modulu se spojí jednotlivé, samostatné a již registrované trojúhelníkové modely do jednoho. Jak lze nahlédnout, implementace je poměrně přímočará, tzn. je zapotřebí pouze přeindexovat položky jednotlivých souborů. Pro více detailů je nutné nahlédnout do implementace příslušného modulu.

Tento modul je možné nahradit skriptem pro *Meshlab*, v kterém se nahraje více modelů a následně se spojí (volba Flatten Visible Layers).
Případně je možné přepočítat normály nad nově vzniklým modelem.

Konfigurace , viz listing 14, obsahuje pouze standardní vstup a výstup, přičemž vstupem je složka, která obsahuje množinu trojúhelníkových modelů pro spojení.

```
module: mesh-merge
output: file_path
input: folder_path
```

Listing 14: Konfigurace modulu pro spojení modelů

4.6.2 Poissonovská rekonstrukce

Tento modul staví na již popsaném modulu, konkrétně na modulu pro spouštění meshlab skriptů, viz sekce Meshlab skript. Jak již název napovídá, tak je použit skript pro Poissonovskou rekonstrukci, viz filtr `meshlab_poisson.mlx` ve složce `Filters`. Tento skript využívá standardního nastavení parametrů pro poissonovskou rekonstrukci v *Matlabu*, tzn. hloubka rekonstrukce 8, hloubka adaptivního octree 5, hloubka konjugovaných gradientů 0, škálovací faktor 1.1, minimální počet vzorků 1.5, interpolační váha 4 a Gauss-Seidel relaxace 8. Jedinou výjimkou je parametr `pre-celan`, který je přenastaven na hodnotu `True`.

Konfigurace je stejná jako v případě Meshlab skriptů, viz sekce Meshlab skript. Samozřejmě je nutné zvolit filtr `meshlab_poisson.mlx` ze složky `Filters`, aby byla získána příslušná funkcionalita.

5 Dosažené výsledky

V této sekci budou uvedeny provedené experimenty s vytvořeným nástrojem pro extrakci statického modelu lidského obličeje ze stereo/multiview videa a jejich výsledky. Dále jsou zmíněny použité postupy, které přímo nevyplývají z funkčnosti nástroje, který byl popsán v předchozích částech této práce. Nakonec jsou zdůrazněny jak špatné, tak dobré vlastnosti jednotlivých nástrojů, přístupů a i samotných výsledků.

Experimenty jsou zaměřeny na dvě hlavní části celého procesu, na fotogrammetrii a následné spojování modelů, resp. registraci. U jednotlivých testů je nutné vzít v potaz, že není k dispozici žádný ideální model vzhledem ke kterému by bylo možné provádět měření a srovnání. Z tohoto důvodu jsou modely porovnávány mezi sebou, popř. je zhodnocena vizuální kvalita výsledku.

Jelikož celý proces generuje velké množství dat (v řádech stovek gigabajtů), tak není možné v této sekci zahrnout všechny nástrojem vygenerované výsledky. Všechny výsledné modely jsou k této práci k dispozici a je možné si je prohlédnout, popř. vyžádat od autorů práce. Přehled všech výsledných modelů je v příloze, viz sekce 7.3.

5.1 Fotogrammetrie

Pro fotogrammetrii je využit již dříve popisovaný nástroj *Meshroom*, resp. framework *AliceVision*. Zvolený nástroj velmi ovlivňuje výslednou kvalitu rekonstrukce, a proto je možné jej vyměnit za jiný (lepší) a tím výrazně vylepšit celý proces. Kroky, které také ovlivňují výsledky fotogrammetrie, ale nebylo s nimi dále experimentováno, jsou kalibrace kamer a synchronizace video stop. Experimenty, které byly připraveny pro část zabývající se fotogrammetrií, jsou zaměřeny na vliv rychlosti otáčení osoby stojící modelem na výslednou kvalitu rekonstrukce modelu. Celkem byly nasnímány čtyři osoby, přičemž na testování se využily tři nejlépe rekonstruované. Tyto osoby souhlasí se zveřejněním materiálu, na kterém jsou zaznamenány v rámci této práce. Na další zpracování nebo zveřejňování těchto materiálů jinou osobou se tento souhlas nevztahuje.

Pro snímání modelu byly vytvořeny co nejideálnější podmínky, tzn. osvětlení pomocí dvou studiových světel, využití camera rig a otočné židle, namalování různobarevných značek na tvář osoby stojící modelem a v neposlední

řadě rozmístění CCtag značek na pozadí snímané scény. Bylo využito deseti kamer *GoPro Hero 7 Silver*.

Rychlost otáčení byla stanovena na 1 sekundu (rychlé) a na 2 sekundy (střední). Za tuto dobu musel model otočit hlavu o 60° (konkrétně z $-30°$ do 30°, chápeme-li prostřední kameru jako počátek). Je nutné si uvědomit, že časy jsou přibližné, jelikož se pracuje s lidmi a je velmi těžké, ba neproveditelné, zajistit přesnou rychlost a rovnoměrnost otáčení modelu. Velmi rychle se ukázalo, že kvalita rychlého otáčení je horší než středního, a to z důvodů viditelného rozmazání ve snímcích. Pomalejší otáčení již nebylo testováno, jelikož nárůst výpočetního času potřebného pro celý proces by rostl úměrně (v některých případech dokonce kvadraticky). Dále tedy byla většina experimentů prováděna na šedesáti snímcích, resp. středně rychlém otáčení modelu.

Kvalita rekonstrukce záleží také na pozici modelu v sekvenci. První model kalibruje pozice kamer a může i upravit vnitřní parametry kamer. Z tohoto důvodu bývá rekonstrukcí nejlepší. Tzn. pokud je první model špatný, nemá cenu v rekonstrukci dalších modelů pokračovat. Obvykle bývají i modely ke konci sekvence velmi kvalitní, a to z důvodů ukončení pohybu hlavy modelu. Naopak prostřední rekonstrukce bývají zatíženy největší chybou, viz obrázek 5.1.

Obrázek 5.1: První, prostřední a poslední rekonstrukce

Problémem celé rekonstrukce je obtížné vynucení stejného měřítka a pozice (ukotvení) jednotlivých modelů v prostoru. Tento problém způsobuje fakt, že pozice kamer získané při kalibraci nelze sdílet (technicky je to možné, ale rekonstrukce zpravidla selže) napříč rekonstrukcemi různých sekvencí modelů, tzn. rozdílné sekvence modelů obecně nemají stejné měřítko (jednou sekvencí rozumíme rekonstrukci na základě jednoho videa, tzn. sekvence

snímků). Je možné, že v dalších verzích *Meshroomu* bude měřítko modelu odpovídat reálným rozměrům snímaného objektu, popř. bude lepší dokumentace pro práci s pevně ukotvenými kamerami v prostoru (camera rig).

Také je nutné zmínit, že při nastavení vysoké kvality (parametr `quality`) může v některých případech selhat pátý krok rekonstrukce (structure from motion(5)).

Další slabinou současného postupu je počet kamer. Většina ukázkových modelů *Meshroom* využívá více jak sto snímků, zatímco v současné verzi popisovaného procesu je využito pouze deseti kamer. Bylo by tedy jistě prospěšné počet kamer zvýšit. Na závěr je také nutné zmínit, že zlepšení kvality snímků by se jistě významně projevilo na kvalitě rekonstrukcí.

Výpočetní čas fotogrammetrie je řádově několik hodin i na výkonném počítači. Konkrétně jedna rekonstrukce trvá 5 až 10 minut, tzn. pro střední rychlost, resp. 60 modelů, rekonstrukce trvala přibližně 8 hodin. Tato část je výpočetně velmi náročná. Ostatní akce jako extrakce a synchronizace trvají řádově minuty.

5.2 Sjednocení modelů

Hlavním zaměřením experimentů je oblast sjednocení modelů, resp. jejich registrace. Pro pokusy byly použity tři registrované modely, dva muži a jedna žena. Osoby s modelem na sobě neměly žádné doplňky, jako jsou např. brýle a čepice. Žena byla požádána, aby si svázala vlasy, jelikož splývající prameny vlasů podél tváře by byly mnohem těžší pro rekonstrukci pomocí fotogrammetrie a jistě by do modelů zanášely další šum. Před samotnou registrací byly odfiltrovány nekvalitní trojúhelníky, které obecně značí nejistotu rekonstrukce v dané oblasti. Před spuštěním nerigidní registrace je nutné upravit vstupní modely, aby neobsahovaly trojúhelníky s nulovým povrchem (zero-area faces), nemanifoldní prvky a malé nesouvislé oblasti.

První zkoumanou oblastí byl vliv rychlosti otáčení na celou registraci. Jak již naznačují pokusy zaměřené na fotogrammetrii, tak se lze domnívat, že výsledky rychlého otáčení nebudou tak dobré. Dále bylo vyzkoušeno, jak moc pohyb modelu ovlivňuje kvalitu výsledného modelu. Poslední oblastí pro experimenty bylo zkoumání různých přístupů registrace a jejich vlivu na výsledky. Tyto rozlišné přístupy jsou následující:

1. Rigidní registrace na prostřední zrekonstruovaný model v sekvenci
2. Nerigidní registrace na prostřední zrekonstruovaný model v sekvenci, za použití již rigidně registrovaných modelů z přístupu 1
3. Nerigidní registrace přes sousední modely v sekvenci až na cílový (prostřední) model. Opět za použití již rigidně registrovaných modelů z přístupu 1
4. Nerigidní registrace modelů z metody 3 na výsledek vytvořený na základě modelů z metody 3 (myšleno po poissonovském remeshingu)

Pro experimenty nebylo použito fine scale alignment, jelikož cílový model, na který se registruje, nemusí být kompletní a často nebývá nejkvalitnější rekonstrukcí snímaného modelu. Právě kvůli těmto skutečnostem se při použití fine scale alignment stávalo, že velmi dobře rekonstruované detaily, např. ucho, z krajních rekonstrukcí sekvence se během sjednocování modelů deformovaly.

Rychlost otáčení modelu je poměrně důležitým faktorem, který měl své negativní dopady, v případě rychlého otáčení, již ve výsledcích rekonstrukce za pomoci fotogrammetrie. Špatná kvalita rekonstrukcí se výrazně podepíše na výsledném modelu již v metodách 1 a 2, a to i pro sekvence s žádným úmyslným pohybem. Na modelu se objevují rozmanité artefakty jako např. lokální propadliny a celkové deformace tvarů. Naopak střední rychlost otáčení se projevila jako dostačující. Výsledky pro střední rychlost otáčení, žádný úmyslný pohyb a metodu registrace 4 viz obrázek 5.2

Jak lze vidět z obrázků 5.2, tak jsou zachycené tvary velmi podobné osobám, které stály modelem. V případě modelů mužů se jako nejproblematičtější oblast ukázal nos, který trpí jistým množstvím nerovností. Naopak uši dopadly velmi dobře u modelů mužů, nikoliv však u modelu ženy. Tento fakt je způsobem množstvím a umístěním vlasů.

5.2.1 Porovnání jednotlivých metod

V případě modelů bez úmyslného pohybu se metody projevily na používaném datasetu jako rovnocenné. Jinými slovy, nevznikaly významné artefakty a modely se mezi sebou lišily jen velmi mírně. Tento poznatek je důležitý z hlediska výpočetního času, který je potřeba pro jednotlivé metody. Rigidní registrace je rychlejší než nerigidní. Dále nerigidní zarovnání přes sousedy má časovou složitost $O(n^2)$ oproti zbývajícím metodám se složitostí $O(n)$.

Obrázek 5.2: Nejlepší výsledky pro jednotlivé modely

Naopak pro sekvence s úmyslným pohybem, který byl u každého modelu rozlišný (např. pohyb obočím, očima, nebo ústy), zaznamenaly odlišné metody rozdíly v kvalitě výsledku (viz obrázek 5.3 a 5.4). Z testů vyplynulo, že jsou si metody 1 a 2 velmi podobné, vzhledem ke kvalitě výsledku. Stejně tak jsou si podobné metody 3 a 4

Na obrázcích 5.3 a 5.4 můžeme vidět artefakty, které vznikly špatnou registrací. Tyto artefakty jsou důsledkem vzdálenosti modelů vstupujících do poissonovské rekonstrukce, při které vznikly nežádoucí vrstvy. Zmíněné vzdálenosti jsou způsobeny špatným zarovnáním pomocí nejjednoduššího postupu založeném na nerigidní registraci. Jak je vidět na obrázcích 5.3 a 5.4, tak metody 3 a 4 tyto artefakty potlačují, byť ne zcela odstraňují. Také je možné, že by se v dalších iteracích metody 4 zahladily zbývající artefakty, byť se tato možnost nejeví příliš pravděpodobná na základě faktu, že se tak nestalo v prvních iteracích.

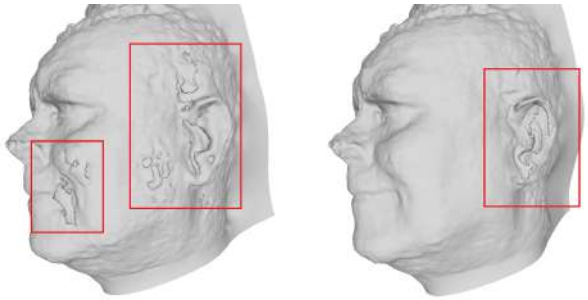
Výpočetní čas registrace se liší dle použité metody. Rigidní registrace je poměrně rychlá a trvá v řádech desítek minut pro střední rychlost, tzn. 60 snímků. Nerigidní metoda číslo 2, resp. jedna iterace metody číslo 4, trvá přibližně hodinu, ale velmi závisí na vstupních modelech. Nejdéle trvá nerigidní registrace za pomoci metody 3, která pro střední rychlost trvala až 10 hodin.

5.3 Možná vylepšení

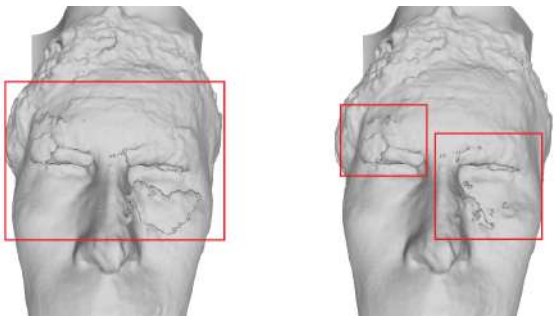
V oblasti fotogrammetrie by bylo vhodné zvýšit její kvalitu. Kromě zřejmých možností jako je větší počet kvalitnějších snímacích zařízení, se nabízí možnost lepší práce se soustavou kamer (camera rig), které jsou po celou dobu na stejném místě. Dále by bylo vhodné zajistit jednotné měřítko rekonstrukcí, tak aby odpovídalo realitě. Pokud se měřítko jednotlivých rekonstrukcí mezi různými snímanými objekty změní, tak je nutné měnit parametry ořezávacího válce, tzn. tento krok není zcela automatický. Další možností je neořezávat na základě pevně definovaného válce, ale nalézt ve scéně oblast zájmu. Poslední možností, jelikož se projekt *AliceVision* stále vyvíjí, bude v budoucnosti jistě přidání masek pro označení oblasti zájmu, které by tento krok velmi usnadnily. Kvalita fotogrammetrie by možná také šla zvýšit promítáním specifického vzoru na obličej osoby stojící modelem.

V oblasti registrace zůstává jedinou oblastí, která by si zasloužila další práci nástroj pro nerigidní registraci. Byť tento software prošel značnými

úpravami v rámci této práce, tak zůstává pár oblastí, které by si zaslouží vylepšení. Nejprve by bylo vhodné nemít žádné omezení na vstupní modely (manifold bez trojúhelníků s nulovým obsahem, který nemá malé komponenty, ale je pokud možno souvislý). Nástroj by mohl úpravy vstupních modelů provádět automaticky, nebo upravit algoritmy, aby jim tyto modely nevadily. Další možnost pro vylepšení nabízí metoda nerigidní registrace 4, při které by bylo vhodné umožnit počáteční odhad pro nerigidní transformaci. Taková funkcionalita by s sebou přinášela mnoho výzev, které by bylo nutné řešit, ale jistě by přinesla vylepšení jak výsledků, tak zkrácení výpočetního času.



Obrázek 5.3: Artefakty modelu muže s pohybem (vpravo metoda 2, vlevo metoda 4 potlačující některé artefakty)



Obrázek 5.4: Artefakty modelu ženy s pohybem (vpravo metoda 2, vlevo metoda 4 potlačující některé artefakty)

6 Závěr

V této práci byl navržen a představen nástroj pro automatizaci extrakce statického modelu lidského obličeje ze stereo/multiview videa.

Nástroj pracuje nad adresářovou strukturou obsahující konfigurační soubory, kterými je řízen celý proces. Takový návrh aplikace spolu s rozčleněním funkcionality do modulů, jež dokonce dovolují velmi snadnou definici příslušných konfiguračních sekcí, umožňuje velmi snadné sestavení celého procesu z jednotlivých modulů. Je tedy možné do jisté míry přizpůsobit celý průběh extrakce specifickým potřebám. Také výměna libovolného modulu za jiný by měla být velmi snadným úkonem.

V části zabývající se fotogrammetrií byl zvolen nástroj *Meshroom*, který je volně dostupný a velmi vhodný pro automatizaci. Výsledky rekonstrukce jsou poměrně kvalitní vzhledem k počtu a kvalitě použitých snímacích zařízení. Jednou ze špatných vlastností tohoto softwaru je nemožnost vynucení reálného měřítka modelu a s tím spojené problémy, jako je např. nutnost změny parametrů válce použitého pro ořezávání rekonstruované scény, za účelem zachování pouze modelu lidského obličeje, pro každou novou sekvenci snímků.

Další oblast, registrace modelů lidského obličeje, podává dobré výsledky. Nástroj pro nerigidní registraci byl značně urychlen výměnou modulu zodpovědného za tvorbu deformačního grafu. I samotný proces hrubého zarovnání byl zbaven závažných chyb vedoucích k selhání registrace. Jediné omezení tohoto nástroje je nutnost kvalitního vstupního modelu, tzn. manifold bez trojúhelníků s nulovou plochou a bez malých nesouvislých částí.

Výsledné modely lidského obličeje závisí svojí kvalitou na množství pohybu osoby stojící modelem. Jejich kvalita by se pravděpodobně dala vylepšit experimentováním s parametry nerigidní registrace a zapojením jemného zarovnání. Dále by bylo jistě vhodné zapojit do procesu více kamer, popř. otestovat další fotogrammetrické nástroje, nebo rozmístění kamer, jelikož z výsledků vyplývá, že kvalita rekonstrukce má zásadní vliv na konečný výsledek.

7 Příloha

7.1 Residua

Residua pro i-tý vrchol:

$$E^1_{fit} = \alpha_{point}||\tilde{\mathbf{x}}_i - \mathbf{c}_i||$$
$$E^2_{fit} = \alpha_{plane}(\mathbf{n}_i^T \cdot (\tilde{\mathbf{x}}_i - \mathbf{c}_i))$$
$$E^1_{rigid} = \mathbf{a}_{i,1}^T \cdot \mathbf{a}_{i,2}$$
$$E^2_{rigid} = \mathbf{a}_{i,1}^T \cdot \mathbf{a}_{i,3}$$
$$E^3_{rigid} = \mathbf{a}_{i,2}^T \cdot \mathbf{a}_{i,3}$$
$$E^4_{rigid} = 1 - ||\mathbf{a}_{i,1}||$$
$$E^5_{rigid} = 1 - ||\mathbf{a}_{i,2}||$$
$$E^6_{rigid} = 1 - ||\mathbf{a}_{i,3}||$$
$$E^1_{smooth} = ||\mathbf{A}_i(\mathbf{x}_1 - \mathbf{x}_i) + \mathbf{x}_i + \mathbf{b}_i - (\mathbf{x}_1 + \mathbf{b}_1)||$$
$$E^2_{smooth} = ||\mathbf{A}_i(\mathbf{x}_2 - \mathbf{x}_i) + \mathbf{x}_i + \mathbf{b}_i - (\mathbf{x}_2 + \mathbf{b}_2)||$$
$$E^3_{smooth} = ||\mathbf{A}_i(\mathbf{x}_3 - \mathbf{x}_i) + \mathbf{x}_i + \mathbf{b}_i - (\mathbf{x}_3 + \mathbf{b}_3)||$$
$$E^4_{smooth} = ||\mathbf{A}_i(\mathbf{x}_4 - \mathbf{x}_i) + \mathbf{x}_i + \mathbf{b}_i - (\mathbf{x}_4 + \mathbf{b}_4)||$$

7.2 Derivace

Parciální derivace, které nejsou pro dané residuum uvedeny jsou vždy nulové. $\frac{\partial E^1_{fit}}{\partial A^k_{11}}$ ... parciální derivace E^1_{fit} pro i-tý vrchol podle prvku [1,1] transformační matice k-tého souseda

$$\begin{bmatrix} \frac{\partial E^1_{fit}}{\partial A^k_{11}} & \frac{\partial E^1_{fit}}{\partial A^k_{12}} & \frac{\partial E^1_{fit}}{\partial A^k_{13}} \\ \frac{\partial E^1_{fit}}{\partial A^k_{21}} & \frac{\partial E^1_{fit}}{\partial A^k_{22}} & \frac{\partial E^1_{fit}}{\partial A^k_{23}} \\ \frac{\partial E^1_{fit}}{\partial A^k_{31}} & \frac{\partial E^1_{fit}}{\partial A^k_{32}} & \frac{\partial E^1_{fit}}{\partial A^k_{33}} \end{bmatrix} = \alpha_{point}\alpha_{fit}\omega_k \frac{(\vec{c_i} - \frac{\sum_{j\in N(i)}\omega_j(\vec{b_j}+\vec{x_j}+A_j(\vec{x_i}-\vec{x_j}))}{\sum_{j\in N(i)}\omega_j})(\vec{x_k}-\vec{x_i})^T}{||\vec{c_i} - \frac{\sum_{j\in N(i)}\omega_j(\vec{b_j}+\vec{x_j}+A_j(\vec{x_i}-\vec{x_j})}{\sum_{j\in N(i)}\omega_j})|| \sum_{j\in N(i)}\omega_j}$$

$$\begin{bmatrix} \frac{\partial E^1_{fit}}{\partial b^1_1} & \frac{\partial E^1_{fit}}{\partial b^1_2} & \frac{\partial E^1_{fit}}{\partial b^1_3} \\ \frac{\partial E^1_{fit}}{\partial b^2_1} & \frac{\partial E^1_{fit}}{\partial b^2_2} & \frac{\partial E^1_{fit}}{\partial b^2_3} \\ \frac{\partial E^1_{fit}}{\partial b^3_1} & \frac{\partial E^1_{fit}}{\partial b^3_2} & \frac{\partial E^1_{fit}}{\partial b^3_3} \\ \frac{\partial E^1_{fit}}{\partial b^4_1} & \frac{\partial E^1_{fit}}{\partial b^4_2} & \frac{\partial E^1_{fit}}{\partial b^4_3} \end{bmatrix} = -\alpha_{point}\alpha_{fit}\vec{\omega} \left[\frac{\vec{c_i} - \frac{\sum_{j\in N(i)}\omega_j(\vec{b_j}+\vec{x_j}+A_j(\vec{x_i}-\vec{x_j}))}{\sum_{j\in N(i)}\omega_j}}{||\vec{c_i} - \frac{\sum_{j\in N(i)}\omega_j(\vec{b_j}+\vec{x_j}+A_j(\vec{x_i}-\vec{x_j}))}{\sum_{j\in N(i)}\omega_j}|| \sum_{j\in N(i)}\omega_j} \right]^T$$

$$\begin{bmatrix} \frac{\partial E^2_{fit}}{\partial A^k_{11}} & \frac{\partial E^2_{fit}}{\partial A^k_{12}} & \frac{\partial E^2_{fit}}{\partial A^k_{13}} \\ \frac{\partial E^2_{fit}}{\partial A^k_{21}} & \frac{\partial E^2_{fit}}{\partial A^k_{22}} & \frac{\partial E^2_{fit}}{\partial A^k_{23}} \\ \frac{\partial E^2_{fit}}{\partial A^k_{31}} & \frac{\partial E^2_{fit}}{\partial A^k_{32}} & \frac{\partial E^2_{fit}}{\partial A^k_{33}} \end{bmatrix} = \frac{\alpha_{plane}\alpha_{fit}\vec{n}\omega_k(\vec{x_i}-\vec{x_k})^T}{\sum_{j\in N(i)}\omega_j}$$

$$\begin{bmatrix} \frac{\partial E^2_{fit}}{\partial b^1_1} & \frac{\partial E^2_{fit}}{\partial b^2_1} & \frac{\partial E^2_{fit}}{\partial b^3_1} & \frac{\partial E^2_{fit}}{\partial b^4_1} \\ \frac{\partial E^2_{fit}}{\partial b^1_2} & \frac{\partial E^2_{fit}}{\partial b^2_2} & \frac{\partial E^2_{fit}}{\partial b^3_2} & \frac{\partial E^2_{fit}}{\partial b^4_2} \\ \frac{\partial E^2_{fit}}{\partial b^1_3} & \frac{\partial E^2_{fit}}{\partial b^2_3} & \frac{\partial E^2_{fit}}{\partial b^3_3} & \frac{\partial E^2_{fit}}{\partial b^4_3} \\ \frac{\partial E^2_{fit}}{\partial b^1_4} & \frac{\partial E^2_{fit}}{\partial b^2_4} & \frac{\partial E^2_{fit}}{\partial b^3_4} & \frac{\partial E^2_{fit}}{\partial b^4_4} \end{bmatrix} = \frac{\alpha_{plane}\alpha_{fit}\vec{n}\vec{\omega}^T}{\sum_{j\in N(i)}\omega_j}$$

$$\begin{bmatrix} \frac{\partial E^1_{rigid}}{\partial A^i_{11}} & \frac{\partial E^1_{rigid}}{\partial A^i_{12}} & \frac{\partial E^1_{rigid}}{\partial A^i_{13}} \\ \frac{\partial E^1_{rigid}}{\partial A^i_{21}} & \frac{\partial E^1_{rigid}}{\partial A^i_{22}} & \frac{\partial E^1_{rigid}}{\partial A^i_{23}} \\ \frac{\partial E^1_{rigid}}{\partial A^i_{31}} & \frac{\partial E^1_{rigid}}{\partial A^i_{32}} & \frac{\partial E^1_{rigid}}{\partial A^i_{33}} \end{bmatrix} = \begin{bmatrix} A_{12}*\alpha_{reg} & A_{11}*\alpha_{reg} & 0 \\ A_{22}*\alpha_{reg} & A_{21}*\alpha_{reg} & 0 \\ A_{32}*\alpha_{reg} & A_{31}*\alpha_{reg} & 0 \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial E^2_{rigid}}{\partial A^i_{11}} & \frac{\partial E^2_{rigid}}{\partial A^i_{12}} & \frac{\partial E^2_{rigid}}{\partial A^i_{13}} \\ \frac{\partial E^2_{rigid}}{\partial A^i_{21}} & \frac{\partial E^2_{rigid}}{\partial A^i_{22}} & \frac{\partial E^2_{rigid}}{\partial A^i_{23}} \\ \frac{\partial E^2_{rigid}}{\partial A^i_{31}} & \frac{\partial E^2_{rigid}}{\partial A^i_{32}} & \frac{\partial E^2_{rigid}}{\partial A^i_{33}} \end{bmatrix} = \begin{bmatrix} A_{13}*\alpha_{reg} & 0 & A_{11}*\alpha_{reg} \\ A_{23}*\alpha_{reg} & 0 & A_{21}*\alpha_{reg} \\ A_{33}*\alpha_{reg} & 0 & A_{31}*\alpha_{reg} \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial E^3_{rigid}}{\partial A^i_{11}} & \frac{\partial E^3_{rigid}}{\partial A^i_{12}} & \frac{\partial E^3_{rigid}}{\partial A^i_{13}} \\ \frac{\partial E^3_{rigid}}{\partial A^i_{21}} & \frac{\partial E^3_{rigid}}{\partial A^i_{22}} & \frac{\partial E^2_{rigid}}{\partial A^i_{23}} \\ \frac{\partial E^3_{rigid}}{\partial A^i_{31}} & \frac{\partial E^3_{rigid}}{\partial A^i_{32}} & \frac{\partial E^3_{rigid}}{\partial A^i_{33}} \end{bmatrix} = \begin{bmatrix} 0 & A_{13}*\alpha_{reg} & A_{12}*\alpha_{reg} \\ 0 & A_{23}*\alpha_{reg} & A_{22}*\alpha_{reg} \\ 0 & A_{33}*\alpha_{reg} & A_{31}*\alpha_{reg} \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial E^4_{rigid}}{\partial A^i_{11}} & \frac{\partial E^4_{rigid}}{\partial A^i_{12}} & \frac{\partial E^4_{rigid}}{\partial A^i_{13}} \\ \frac{\partial E^4_{rigid}}{\partial A^i_{21}} & \frac{\partial E^4_{rigid}}{\partial A^i_{22}} & \frac{\partial E^4_{rigid}}{\partial A^i_{23}} \\ \frac{\partial E^4_{rigid}}{\partial A^i_{31}} & \frac{\partial E^4_{rigid}}{\partial A^i_{32}} & \frac{\partial E^4_{rigid}}{\partial A^i_{33}} \end{bmatrix} = \begin{bmatrix} -\frac{A_{11}*\alpha_{reg}}{\sqrt{A^2_{11}+A^2_{21}+A^2_{31}}} & 0 & 0 \\ -\frac{A_{21}*\alpha_{reg}}{\sqrt{A^2_{11}+A^2_{21}+A^2_{31}}} & 0 & 0 \\ -\frac{A_{31}*\alpha_{reg}}{\sqrt{A^2_{11}+A^2_{21}+A^2_{31}}} & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial E^5_{rigid}}{\partial A^i_{11}} & \frac{\partial E^5_{rigid}}{\partial A^i_{12}} & \frac{\partial E^5_{rigid}}{\partial A^i_{13}} \\ \frac{\partial E^5_{rigid}}{\partial A^i_{21}} & \frac{\partial E^5_{rigid}}{\partial A^i_{22}} & \frac{\partial E^5_{rigid}}{\partial A^i_{23}} \\ \frac{\partial E^5_{rigid}}{\partial A^i_{31}} & \frac{\partial E^5_{rigid}}{\partial A^i_{32}} & \frac{\partial E^5_{rigid}}{\partial A^i_{33}} \end{bmatrix} = \begin{bmatrix} 0 & 0 & -\frac{A_{12}*\alpha_{reg}}{\sqrt{A^2_{12}+A^2_{22}+A^2_{32}}} \\ 0 & 0 & -\frac{A_{22}*\alpha_{reg}}{\sqrt{A^2_{12}+A^2_{22}+A^2_{32}}} \\ 0 & 0 & -\frac{A_{32}*\alpha_{reg}}{\sqrt{A^2_{12}+A^2_{22}+A^2_{32}}} \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial E^6_{rigid}}{\partial A^i_{11}} & \frac{\partial E^6_{rigid}}{\partial A^i_{12}} & \frac{\partial E^6_{rigid}}{\partial A^i_{13}} \\ \frac{\partial E^6_{rigid}}{\partial A^i_{21}} & \frac{\partial E^6_{rigid}}{\partial A^i_{22}} & \frac{\partial E^6_{rigid}}{\partial A^i_{23}} \\ \frac{\partial E^6_{rigid}}{\partial A^i_{31}} & \frac{\partial E^6_{rigid}}{\partial A^i_{32}} & \frac{\partial E^6_{rigid}}{\partial A^i_{33}} \end{bmatrix} = \begin{bmatrix} 0 & -\frac{A_{13}*\alpha_{reg}}{\sqrt{A^2_{13}+A^2_{23}+A^2_{33}}} & 0 \\ 0 & -\frac{A_{23}*\alpha_{reg}}{\sqrt{A^2_{13}+A^2_{23}+A^2_{33}}} & 0 \\ 0 & -\frac{A_{33}*\alpha_{reg}}{\sqrt{A^2_{13}+A^2_{23}+A^2_{33}}} & 0 \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial E^k_{smooth}}{\partial A^i_{11}} & \frac{\partial E^k_{smooth}}{\partial A^i_{12}} & \frac{\partial E^k_{smooth}}{\partial A^i_{13}} \\ \frac{\partial E^k_{smooth}}{\partial A^i_{21}} & \frac{\partial E^k_{smooth}}{\partial A^i_{22}} & \frac{\partial E^k_{smooth}}{\partial A^i_{23}} \\ \frac{\partial E^k_{smooth}}{\partial A^i_{31}} & \frac{\partial E^k_{smooth}}{\partial A^i_{32}} & \frac{\partial E^k_{smooth}}{\partial A^i_{33}} \end{bmatrix} = \frac{\alpha_{reg}(\vec{b_i} - \vec{b_k} - \vec{x_k} + \vec{x_i} + A_i(\vec{x_k} - \vec{x_i})(\vec{x_k} - \vec{x_i})^T)}{10 * ||\vec{b_i} - \vec{b_k} - \vec{x_k} + \vec{x_i} + A_i(\vec{x_k} - \vec{x_i})||}$$

$$\begin{bmatrix} \frac{\partial E^k_{smooth}}{\partial b^i_1} \\ \frac{\partial E^k_{smooth}}{\partial b^i_2} \\ \frac{\partial E^k_{smooth}}{\partial b^i_3} \end{bmatrix} = \frac{\alpha_{reg}(\vec{b_i} - \vec{b_k} - \vec{x_k} + \vec{x_i} + A_i(\vec{x_k} - \vec{x_i}))}{10 * ||\vec{b_i} - \vec{b_k} - \vec{x_k} + \vec{x_i} + A_i(\vec{x_k} - \vec{x_i})||}$$

$$\begin{bmatrix} \frac{\partial E^k_{smooth}}{\partial b^k_1} \\ \frac{\partial E^k_{smooth}}{\partial b^k_2} \\ \frac{\partial E^k_{smooth}}{\partial b^k_3} \end{bmatrix} = -\frac{\alpha_{reg}(\vec{b_i} - \vec{b_k} - \vec{x_k} + \vec{x_i} + A_i(\vec{x_k} - \vec{x_i}))}{10 * ||\vec{b_i} - \vec{b_k} - \vec{x_k} + \vec{x_i} + A_i(\vec{x_k} - \vec{x_i})||}$$

7.3 Výsledné modely

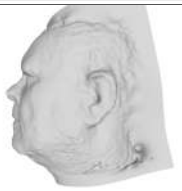
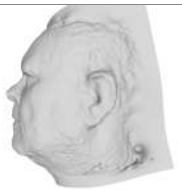
Následuje přehled kompletních výsledků. Jednotlivé metody odpovídají číslování ze sekce 5 a jsou vždy uvedeny v prvním sloupci tabulky. Vzhledem k omezenému prostoru pro jednotlivé modely není vždy možné rozlišit jemné rozdíly a je proto vhodné prozkoumat modely dodané s touto prací.

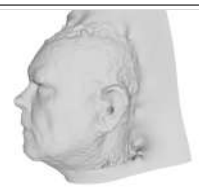
Muž č. 1, střední rychlost, bez úmyslného pohybu			
1.			
2.			
3.			
4.			

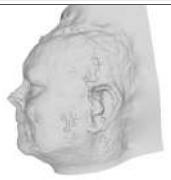
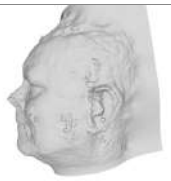
Muž č. 1, rychle, bez úmyslného pohybu			
1.			
2.			

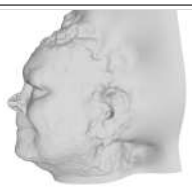
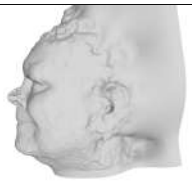
Muž č. 1, střední rychlost, s úmyslným pohybem			
1.			
2.			
3.			
4.			

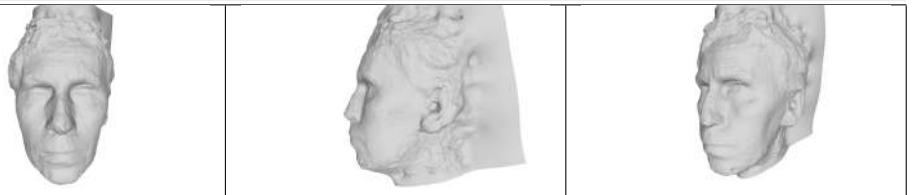
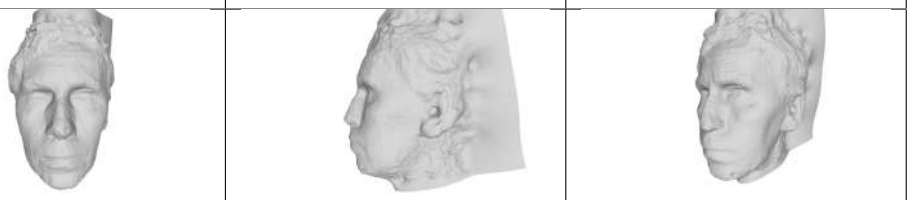
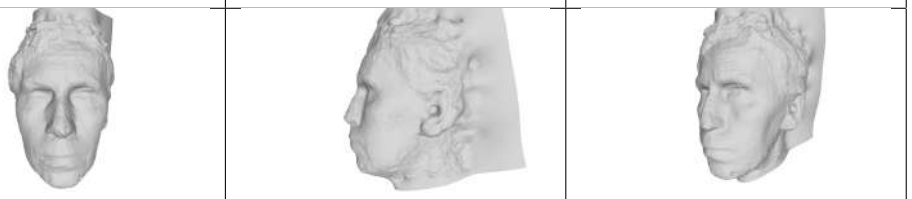
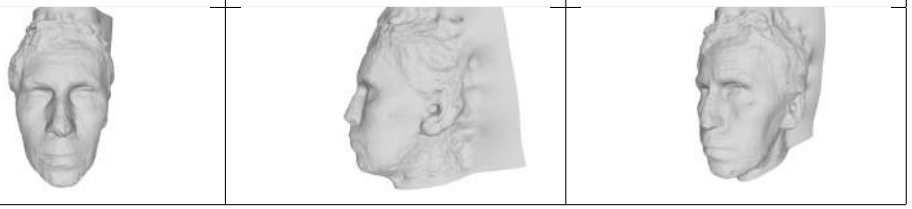
Muž č. 1, rychle, s úmyslným pohybem			
1.			
2.			

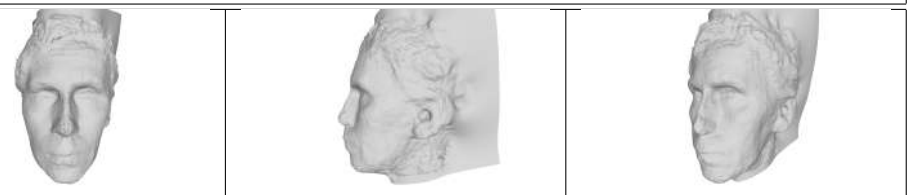
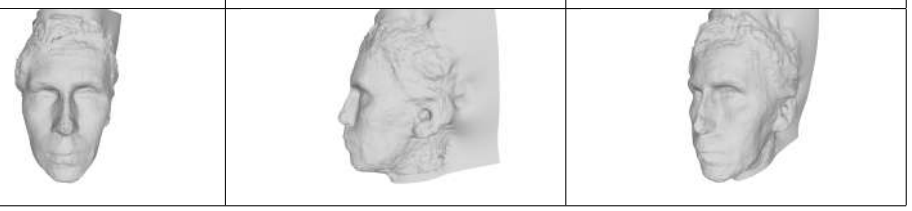
Muž č. 2, střední rychlost, bez úmyslného pohybu			
1.			
2.			
3.			
4.			

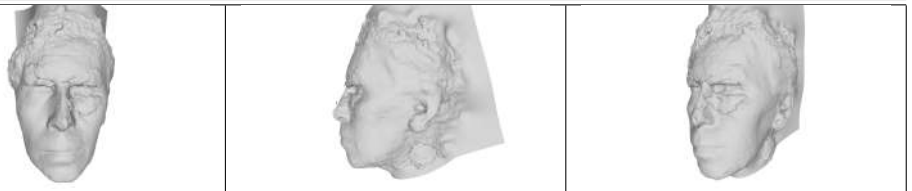
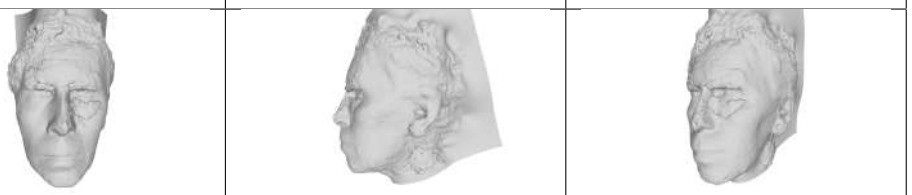
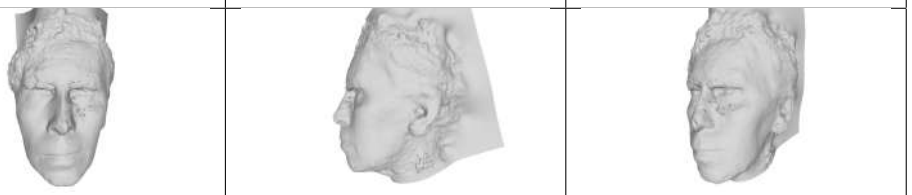
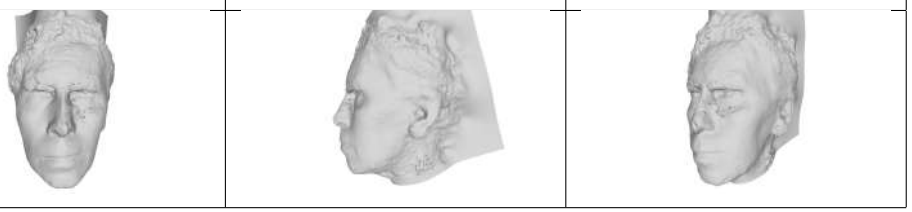
Muž č. 2, rychle, bez úmyslného pohybu			
1.			
2.			

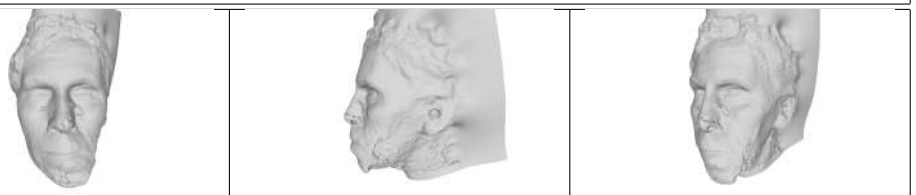
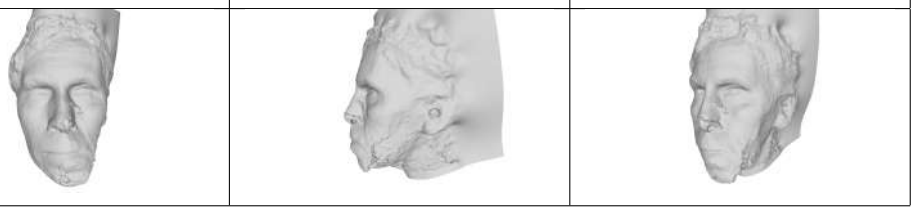
Muž č. 2, střední rychlost, s úmyslným pohybem			
1.			
2.			
3.			
4.			

Muž č. 2, rychle, s úmyslným pohybem			
1.			
2.			

Žena č. 1, střední rychlost, bez úmyslného pohybu			
1.			
2.			
3.			
4.			

Žena č. 1, rychle, bez úmyslného pohybu			
1.			
2.			

Žena č. 1, střední rychlost, s úmyslným pohybem			
1.			
2.			
3.			
4.			

Žena č. 1, rychle, s úmyslným pohybem			
1.			
2.			

Literatura

[1] Aiger, D. – Mitra, N. J. – Cohen-Or, D. 4-Points Congruent Sets for Robust Pairwise Surface Registration. *ACM Trans. Graph.* August 2008, 27, 3, s. 1–10. ISSN 0730-0301. doi: 10.1145/1360612.1360684. Dostupné z: https://doi.org/10.1145/1360612.1360684.

[2] AliceVision. *PHOTOGRAMMETRY PIPELINE* [online]. [cit. 17.7.2020]. Dostupné z: https://alicevision.org/#photogrammetry/.

[3] AliceVision/Meshroom. *Reconstruction Parameters* [online]. [cit. 17.7.2020]. Dostupné z: https://github.com/alicevision/meshroom/wiki/Reconstruction-parameters.

[4] Bojsen-Hansen, M. – Li, H. – Wojtan, C. Tracking Surfaces with Evolving Topology. *ACM Trans. Graph.* July 2012, 31, 4. ISSN 0730-0301. doi: 10.1145/2185520.2185549. Dostupné z: https://doi.org/10.1145/2185520.2185549.

[5] Brunnström, K. – Stoddart, A. Genetic Algorithms for Free-Form Surface Matching. 08 1996.

[6] Chow, C. K. – Tsui, H. T. – Lee, T. Surface registration using a dynamic genetic algorithm. *Pattern Recognition*. 2004, 37, 1, s. 105 – 117. ISSN 0031-3203. doi: https://doi.org/10.1016/S0031-3203(03)00222-X. Dostupné z: http://www.sciencedirect.com/science/article/pii/S003132030300222X.

[7] Gratton, S. – Lawless, A. – Nichols, N. Approximate Gauss–Newton Methods for Nonlinear Least Squares Problems. *SIAM Journal on Optimization*. 01 2007, 18, s. 106–132. doi: 10.1137/050624935.

[8] Hanzl, V. *Teoretické základy fotogrammetrie* [online]. Vysoké učení technické v Brně, Fakulta stavební, 2006. [cit. 17.7.2020]. Dostupné z: http://fast.darmy.net/opory%20-%20III%20Bc/GE15-Fotogrammetrie_I--M01-Teoreticke_zaklady_fotogrammetrie.pdf.

[9] Hartley, R. I. – Zisserman, A. *Multiple View Geometry in Computer Vision*, s. 239–261. Cambridge University Press, ISBN: 0521540518, second edition, 2004. Dostupné z: https://www.robots.ox.ac.uk/~vgg/hzbook/hzbook2/HZepipolar.pdf.

[10] Hata, K. – Savarese, S. *Epipolar geometry* [online]. Standford University. [cit. 17.7.2020]. Dostupné z: https://web.stanford.edu/class/cs231a/course_notes/03-epipolar-geometry.pdf.

[11] Hruda, L. – Dvořák, J. – Váša, L. On evaluating consensus in RANSAC surface registration. *Computer Graphics Forum*. 08 2019, 38, s. 175–186. doi: 10.1111/cgf.13798.

[12] K, A. M. . A. *Camera Calibration* [online]. 2013. [cit. 17.7.2020]. Dostupné z: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_calib3d/py_calibration/py_calibration.html.

[13] Kazhdan, M. – Hoppe, H. Screened Poisson Surface Reconstruction. *ACM Trans. Graph.* July 2013, 32, 3. ISSN 0730-0301. doi: 10.1145/2487228.2487237. Dostupné z: https://doi.org/10.1145/2487228.2487237.

[14] Kazhdan, M. – Bolitho, M. – Hoppe, H. Poisson Surface Reconstruction. In Sheffer, A. – Polthier, K. (Ed.) *Symposium on Geometry Processing*. The Eurographics Association, 2006. doi: 10.2312/SGP/SGP06/061-070. ISBN 3-905673-24-X.

[15] Kimmel, R. – Sethian, J. A. Computing geodesic paths on manifolds. *Proceedings of the National Academy of Sciences*. 1998, 95, 15, s. 8431–8435. ISSN 0027-8424. doi: 10.1073/pnas.95.15.8431. Dostupné z: https://www.pnas.org/content/95/15/8431.

[16] Li, H. – Sumner, R. – Pauly, M. Global Correspondence Optimization for Non-Rigid Registration of Depth Scans. *Computer Graphics Forum*. 07 2008, 27. doi: 10.1111/j.1467-8659.2008.01282.x.

[17] Lowe, D. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*. 11 2004, 60, s. 91–. doi: 10.1023/B:VISI.0000029664.99615.94.

[18] MathWorks. *Fisheye camera calibration* [online]. . [cit. 17.7.2020]. Dostupné z: https://www.mathworks.com/help/vision/ug/fisheye-calibration-basics.html.

[19] MathWorks. *What Is Camera Calibration?* [online]. . [cit. 17.7.2020]. Dostupné z: https://www.mathworks.com/help/vision/ug/camera-calibration.html.

[20] Meshroom. *Meshroom Manual* [online]. [cit. 17.7.2020]. Dostupné z: https://meshroom-manual.readthedocs.io/en/latest/.

[21] Mittelhammer, R. – Judge, G. – Miller, D. *Econometric Foundations Pack with CD-ROM*, s. 197–198. Cambridge: Cambridge University Press, 01 2000. ISBN 0-521-62394-4.

[22] POCH, D. Skenování animací lidských obličejů zařízením MS Kinect, Plzeň, 2018. Bakalářská práce (Bc.). Západočeská univerzita v Plzni, Fakulta aplikovaných věd. Vedoucí práce Libor Váša.

[23] Procházková, J. – Martišek, D. Notes on Iterative Closest Point Algorithm. 04 2018.

[24] Zakharov, A. – Tuzhilkin, A. – Zhiznyakov, A. Finding Correspondences Between Images using Descriptors and Graphs. *Procedia Engineering*. 12 2015, 129, s. 391–396. doi: 10.1016/j.proeng.2015.12.131.

[25] Zhou, Q.-Y. – Park, J. – Koltun, V. Fast Global Registration. 9906, 10 2016. doi: 10.1007/978-3-319-46475-6_47.