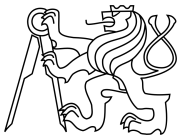
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Rekonstrukce 3D modelu pomocí hloubkových kamer
Student:	Bc. Jakub Průša
Vedoucí:	Ing. Luboš Helcl
Studijní program:	Informatika
Studijní obor:	Znalostní inženýrství
Katedra:	Katedra teoretické informatiky
Platnost zadání:	Do konce letního semestru 2016/17

Pokyny pro vypracování

1. Proveďte rešerši dostupných technologií a metod.
a) Porovnejte metody více pevně pozicovaných kamer oproti jedné pohyblivé kameře.
b) Na základě této rešerše zvolte vhodnou technologii pro rekonstrukci 3D modelu.
2. Pro zvolenou technologii navrhněte robustní řešení tak, aby byl výsledek použitelný v obuvnickém průmyslu.
3. Navrhnuté řešení implementujte.
4. Vyzkoušejte použití Vašeho řešení v obuvnickém průmyslu - pro sestavení obuvnického kopyta.
a) Skenování nohy pomocí hloubkových kamer.
b) Rekonstrukce 3D modelu.
c) Převod 3D modelu pro potřeby obuvnického kopyta.
5. Zhodnoťte použitelnost Vašeho řešení v praxi.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

doc. Ing. Jan Janoušek, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 2. února 2016

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra teoretické informatiky

Diplomová práce

Rekonstrukce 3D modelu pomocí hloubkových kamer

Bc. Jakub Průša

Vedoucí práce: Ing. Luboš Helcl

10. ledna 2017

Poděkování

Děkuji svému vedoucímu práce **Ing. Lubošovi Helclovi** za časté konzultace a pomoc při řešení problémů, které během práce vyvstaly.

Velké poděkování patří mé přítelkyni **Ing. Magdě Friedjungové**, která mi byla po celou dobu oporou.

Velký dík patří mým **rodičům**, kteří mě po celou dobu studia podporovali.

V neposlední řadě děkuji **Ing. Miroslavu Hrončokovi** za typografické a formální připomínky.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 10. ledna 2017 .....

Odkaz na tuto práci

Průša, Jakub. *Rekonstrukce 3D modelu pomocí hloubkových kamer.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Tato práce se zabývá skenováním lidské nohy pomocí hloubkových kamer za účelem získání skenování je získání 3D skenu, který je následně převeden na ševcovské kopyto. Součástí práce je i rešerše hloubkových kamer.

Klíčová slova hloubkové kamery, 3D skenování, 3D model, obuvnické kopyto, ševcovské kopyto, mračno bodů, Intel® RealSense™, rekonstrukce 3D modelu, transformace

Abstract

The thesis is focuses on scanning human feet using depth cameras. Such process is done in order to capture 3D scans which are later transformed into appropriate mechanical forms, so-called lasts. The thesis also includes study and comparison of available depth cameras.

Keywords depths cameras, 3D scanning, 3D model, shoe last, point cloud, Intel® RealSense™, reconstruction of 3D model, transformation

Obsah

Seznam obrázků

Seznam ukázek kódu

Seznam tabulek

Úvod

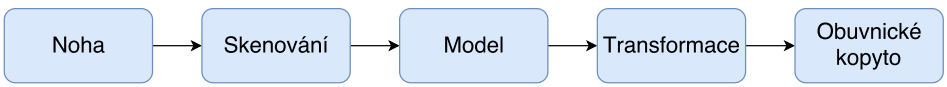
V současné době velmi roste poptávka po věcech, které jsou jedinečné, nápadité a přesně padnoucí. Zároveň vzniká mnoho nových a zajímavých technologií v oblasti hloubkových kamer. Tyto technologie postupně přestávají být záležitostí špičkových laboratoří, ale stávají se součástí elektronických zařízení, která jsou běžně dostupná široké laické veřejnosti.

Motivace a cíl práce

V této diplomové práci jsem se rozhodl dva výše popsané trendy propojit tak, aby byly viditelné praktické využití moderních technologií a dostupnost řešení veřejnosti. Zvolil jsem proto úlohu tvorby zakázkové obuvi, kdy dochází k syntéze technologie hloubkových kamer a vytvoření ševcovského kopyta pro obuv na míru.

Ve své práci používám mj. kamery Intel® RealSense™, které se začínají objevovat v běžně dostupných počítačích. Za velmi krátkou dobu bude tedy možné, aby si každý zájemce o obuv na míru pořídil snímky svých nohou pomocí svého počítače a v teple domova[1].

Cílem mé práce je vypracovat rešerši skenovacího procesu pomocí hloubkových kamer. Dále zhodnotit jednotlivé přístupy a zvolit nejvhodnější, naskenovat nohu a její počítačový model převést na obuvnické kopyto (anglicky last form).



Obrázek 0.1: Zjednodnušený proces mé práce

Struktura práce

Práce je členěna na teoretickou část, která zahrnuje první a druhou kapitolu. První kapitola poskytuje úvod do problematiky rekonstrukce 3D modelu pomocí hloubkových kamer a skenerů. Druhá kapitola popisuje nezbytnou terminologii a přípravu pomocného modelu. Implementační část práce začíná třetí kapitolou, ve které je popsán samotný skenovací proces. Zde jsou popsány i jednotlivé kamery, které podrobím srovnání, a následně i všechny možnosti vyčítání dat z kamer. Jednotlivé kroky rekonstrukce pro vytvoření 3D modelu z mračna bodů jsou popsané ve čtvrté kapitole. V páté, kapitole jsem diskutoval možnosti převodu modelu na ševcovské kopyto. Nachází se zde popisy mnou navržených algoritmů společně s jejich srovnáním. Práce obsahuje mnohá tvrzení, která jsem ověřil testováním v poslední, šesté kapitole. Tato testování jsem doložil četnými grafy a obrázky.

Část I

Teoretická část

Kapitola 1

Analýza

Tato kapitola se zabývá především problematikou konstrukce ševcovského kopyta a analýzou skenerů a kamer vhodných k jeho sestavení. Analyzované skenery a kamery byly voleny tak, aby reflektovaly poslední trendy v oblasti vývoje.

1.1 Ševcovské kopyto

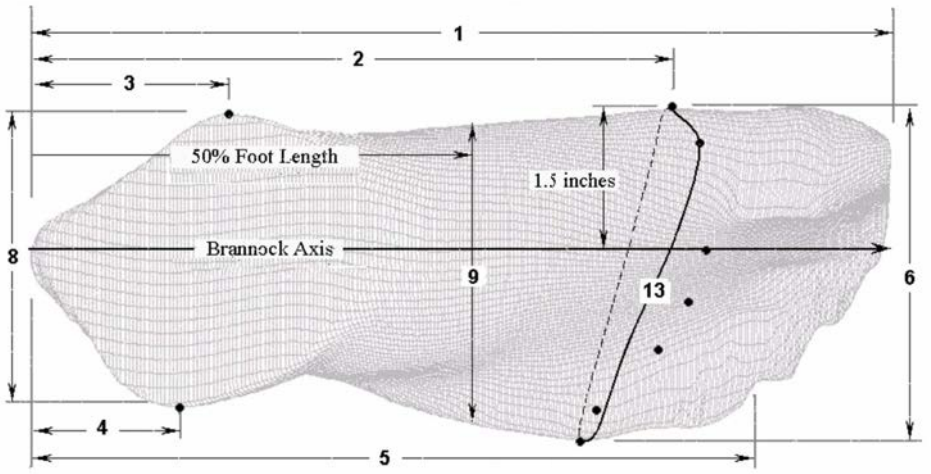
Ševcovské neboli obuvnické kopyto je forma, na které se při ruční výrobě tvarují svršky bot. Jejím původním materiálem bylo dřevo, ale v současnosti se vyrábí z mnoha různorodých materiálů. Kopyto může být výraběno ručně nebo pomocí strojů, a to buď pomocí CNC nebo opačným postupem na 3D tiskárnách [2].

(a) Dřevěná ševcovská kopyta [3]

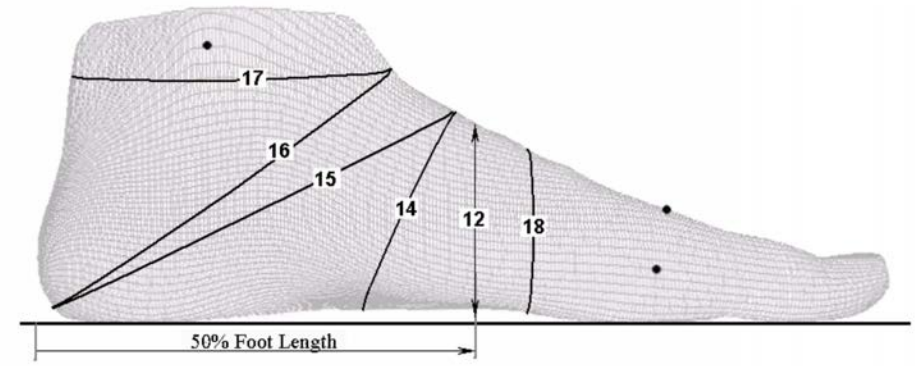
(b) Kopyta vytištěná na 3D tiskárně

Obrázek 1.1: Dřevěná kopyta a kopyta, které jsem vytiskl na 3D tiskárně

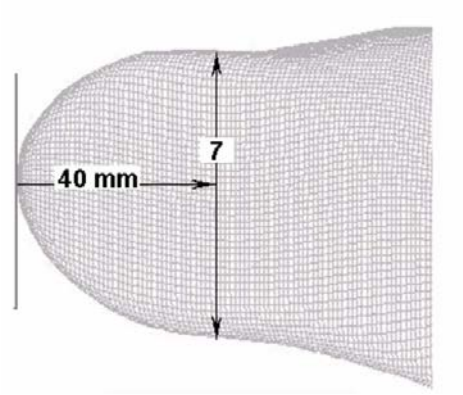
Pro tuto práci bylo nutné se seznámit s celým procesem výroby ševcovského kopyta. V případě, že se jedná o kopyto na míru určitému zákazníkovi, provádí se běžně celkem 18 měření [4]. Ukázka těchto měření je vidět na obrázku 1.2.



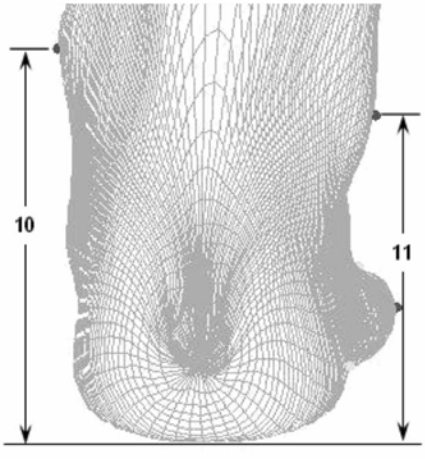
(a) Pohled shora



(b) Pohled ze strany



(c) Pohled zespodu



(d) Pohled zezadu

Obrázek 1.2: Ukázka měření pro ševcovské kopyto [4]

1.2 Skenery a kamery

V současné době existuje mnoho různých typů skenerů, které se dají použít pro skenování 3D objektů, v našem případě nohou. V této práci se budu zabývat dvěma typy skenerů:

- skenery promítající vodící čáru,

  například YETI 3D Foot Scanner [5];

- hloubkové kamery a skenery,

  například Xbox Kinect [6][7].

Další typy skenerů, které se běžně používají, ale v této práci se jim věnovat nebudu, jsou:

- kontaktní skenery,
- laserové skenery.

1.2.1 Skenery promítající vodící čáru

Jedná se o skenery, které se skládají ze dvou hlavních částí – laserového projektoru a kamery. Laserový projektor promítá na objekt rovnou čáru, která se na objektu vlivem jeho tvaru jeví jako křivka. Následně dojde k vyfocení této křivky kamerou, a díky triangulaci[1] dojde ke zpracování a převedení křivky do 3D modelu. Model se o část pootočí a tento postup se opakuje, dokud model neopíše celý kruh.

Existují různé modifkace, kdy se může pohybovat i laserový projektor, nebo se můžeme setkat s více laserovými projektory najednou či dokonce s více kamerami, které fotografují promítanou čáru.

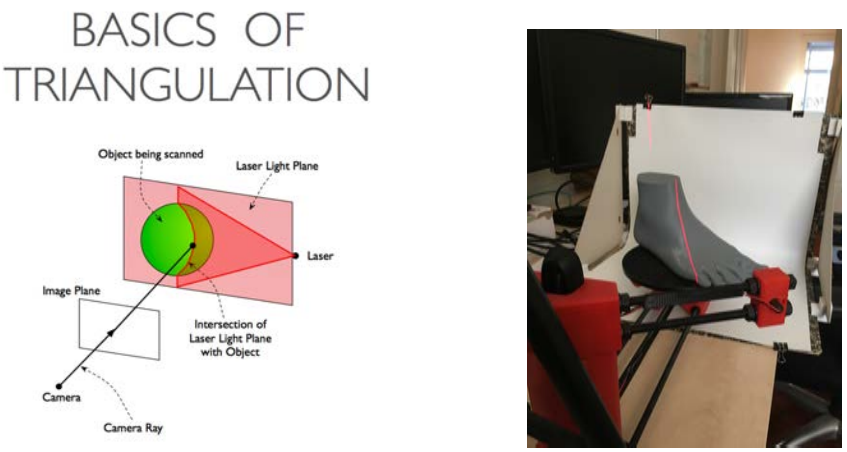
Tato technologie bývá často zaměňována s laserovým skenerem, kde se laserový paprsek odráží od objektu a putuje zpátky do skeneru. Výhoda této technologie je především v její jednoduchosti, díky níž je možné si postavit na podobném principu fungující vlastní 3D skener.

1.2.2 Hloubkové kamery a skenery

Hloubková kamera se skláda ze tří základních částí [9]:

1. RGB kamery,
2. infračervené kamery,
3. infračerveného projektoru.

[1] Triangulace je způsob zjišťování souřadnic a vzdáleností. Provádí se trigonometrickým výpočtem.



(a) Princip fungování [8]

(b) Fotografie skeneru

Obrázek 1.3: Princip fungování a fotografie skeneru promítajícího vodící čáru

Infračervený projektor promítá nepravidelnou síť bodů v infračerveném světle o vlnové délce od 700 nm do 1 mm [10]. Následně jsou tyto body sejmuty pomocí infračervené kamery a předány do procesoru pro zpracování hloubkového obrazu přímo v kameře. Z tohoto procesoru lze poté získat hloubkovou mapu. Procesor pozná podle hustoty bodů výslednou hloubku. Síť bodů je nepravidelná a procesor rozeznává jednotlivé vzory v této síti a mezi nimi trianguluje vzdálenost.

Následně je možné obraz z hloubkové mapy spojovat s obrazem z RGB kamery.

1.2.3 Konkrétní modely 3D skenerů

V této sekci jsou popsané konkrétní skenery, které buďto v práci využívám nebo jsou běžně využívané v praxi [12][13][4].

1.2.3.1 YETI 3D Foot Scanner

Na základě rešerše jsem zjistil, že se v praxi nejvíce používá skener od firmy Vorum s modelovým názvem YETI 3D Foot Scanner [5] (dále jen YETI). Jedná se o laserový skener, který pomocí 4 laserů promítá na daný objekt čáru, kterou následně 8 kamerami snímá. Z tohoto snímání je zrekonstruován 3D model. Tento skener je přímo zaměřen na skenování nohou do úrovně nad kotník. Skener umí skenovat pouze jednu nohu najednou. Bohužel u tohoto skeneru výrobce neudává přesné parametry, nelze ho tak lépe porovnávat s ostatími.

Většina vědeckých článků (například [13][4][2][14]) ohledně obuvnických kopyt, skenování nohou a další práce v tomto odvětví neřeší možnost jiného

(a) Hloubková kamera

(b) Síť bodů

(c) Hloubková mapa

Obrázek 1.4: Ukázka fungování hloubkové kamery [11]

skenování, než pomocí tohoto skeneru. Ve své práci chci proto naopak prozkoumat jiné možnosti.

1.2.3.2 Ciclop 3D Scanner

Jedná se o 3D skener od španělské firmy bq[15] s podobnou technologií jako má YETI skener. Skener má 2 laserové projektory, které promítají červenou čáru na objekt, kterou následně zpracuje kamera umístěná uprostřed skeneru. Jedné se o open-source a open-hardware projekt [16]. Kamera, která se používá pro snímání červených čar, je běžně dostupná webkamera Logitech C270 [17]. Tuto kameru je možné nahradit jakoukoliv jinou kamerou, z tohoto důvodu nedává smysl zabývat se u kamery konkrétními parametry.

S tímto skenerem mám dlouhodobé zkušenosti, především díky mému působení v laboratoři 3D tisku na FIT ČVUT.[2] Ke skeneru je dodáván kompletní

[2]Skener jsem si pro potřeby diplomové práce vypůjčil z laboratoře 3D tisku FIT ČVUT.

(a) Logo výrobce (b) Fotografie skeneru

Obrázek 1.5: YETI 3D Foot Scanner a logo výrobce[5]

nástroj pro ovládání a skenování. Pomocí něj jsem skenoval modely nohou. Více o výsledcích skenování najdete v kapitole 3 na straně 33 .

Obrázek 1.6: Skener Ciclop [15]

1.2.3.3 Microsoft Xbox Kinect 1

Jedná se o nejstaršího zástupce mezi kamerami a skenery, které jsem v této práci použil. Tato kamera má velkou zásluhu v oblasti hloubkových kamer, kdy jako první přinesla relativně levnou možnost používat hloubkové kamery

pro širokou veřejnost. Byla představena v roce 2010 a od roku 2011 vyšlo oficiální SDK[3] pro vývoj aplikací, které podporují Kinect[4]. Kamera, jako jediná ze zmíněných, má ve svém podstavci zabudovaný motor pro automatizovanou změnu úhlu snímání. Z tohoto důvodu má kamera přídavný kabel pro napájení.[6][7]

Název kamery	Xbox Kinect 1
Rozlišení RGB kamery	VGA (640×480) : 15 fps
Rozlišení hloubkové kamery	VGA (640×480) : 15 fps
Operační vzdálenost	0,7 m až 6 m
Výrobce	Microsoft
Rok vydání	2010

Tabulka 1.1: Parametry Xbox Kinect 1

1.2.3.4 ASUS Xtion PRO LIVE

Skener je velmi podobný Xbox Kinectu. Hlavním rozdílem oproti Kinectu je vyšší rozlišení RGB kamery a absence motoru. Ukázka kamery se nachází na obrázku 1.4. Nevýhodou tohoto skeneru je, že projekt není dále vyvíjen – poslední aktualizace na stránkách výrobce pochází z roku 2013. Od té doby nebyl vydán ani nový firmware či aktualizace dokumentace [18]. Nicméně jsem se rozhodl tento skener do práce zahrnout a srovnat jeho kvality s ostatními.[5]

Název kamery	ASUS Xtion PRO LIVE
Rozlišení RGB kamery	SXGA (1280×1024)
Rozlišení hloubkové kamery	VGA (640×480) : 30 fps
Operační vzdálenost	0,8 m až 3,5 m
Výrobce	ASUS
Rok vydání	2013

Tabulka 1.2: Parametry ASUS Xtion PRO LIVE

1.2.3.5 Intel® RealSense™ F200

Obchodní označení je Short-Range Intel® RealSense™ Camera [19]. Intel® RealSense™ je technologie, která v současné době není zatím příliš populární (například oproti Xbox Kinectu), protože je ve stádiu takzvaných dev-kitů. To znamená, že kamery nejsou veřejně vydané mezi širokou veřejnost, ale je

[3]SDK neboli Software dev-kit je sada vývojových nástrojů umožňující vyvíjení aplikací pro určité softwarové balíčky, framework, platformy, systémy, konzole apod.

[4]Tento skener jsem si vypůjčil od vedoucího této práce.

[5]Tento skener jsem si vypůjčil od vedoucího této práce.

možné si zakoupit vývojový kit neboli balíček se zmíněnou kamerou. Největším limitem je momentální nemožnost použití této kamery či technologie pro komerční účely, protože je nutné počkat na její oficiální vydání.

Tato technologie by časem měla být dostupná ve všech laptopech a tabletech, kde nahradí standardní webovou kameru [20]. Je to technologie, která má šanci změnit to, jak používáme počítače, protože obsahuje ,mimo jiné, odladěné rozpoznávání gest, které se dá použít nejen pro jejich ovládání. Důkazem toho, že Intel® myslí tuto technologii vážně, je i to, že v současné době existuje již 6 notebooků, které RealSense™ technologii mají [21]. Pouze jeden z nich má technologii na bázi stereo kamer (obchodní označení Long-Range Intel® RealSense™ Camera [22]), ostatních pět používá technologii hloubkové kamery:

- Long-Range Intel® RealSense™ Camera:

  HP Spectre x2 12-a001dx.

- Short-Range Intel® RealSense™ Camera:

  HP ENVY 34-a010,

  Lenovo IdeaPad Y700 15,

  Dell Inspiron 17 5000 Laptop Touch,

  Dell Inspiron 24 7000 Desktop,

  HP Pavilion Gaming 15-ak010nr.

Za povšimnutí stojí fakt, že integrátory kamery jsou různéí výrobci počítačů, tudíž se nejedná pouze o jednu značku, která by se společností Intel® exkluzivně spolupracovala. V budoucnu se tak nejspíš dočkáme dalších počítačů různých značek s touto technologií.[6]

Již nyní existuje několik aplikací, které kameru používají. Namátkou CatchEye [23], ArcSoft™ DepthCam [24].

Název kamery	Intel® RealSense™ F200
Rozlišení RGB kamery	1080p (1920×1080) : 30 fps
Rozlišení hloubkové kamery	VGA (640×480) : 30 fps
Operační vzdálenost	0,2 m až 1,2 m (5 m)
Výrobce	Intel®
Rok vydání	2015[a]

Tabulka 1.3: Parametry Intel® RealSense™ F200

[a]Kamera ještě nebyla oficiálně vydána, k dispozici je pouze balíček pro vývojáře.

[6]Tuto kameru jsem si zakoupil jako dev-kit z vlastní zvědavosti již dříve.

(a) Ukázka kamery

(b) Skenování nohy kamerou v notebooku [1]

Obrázek 1.7: Intel® RealSense™ F200

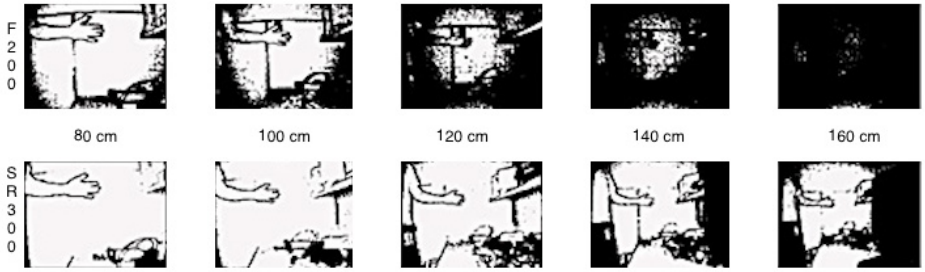
1.2.3.6 Intel® RealSense™ SR300

Tato kamera se podle oficiálních stránek nedá koupit ani jako dev-kit. Je možný pouze její předprodej[7]. Důkazem, že se jedná o novou technologii, může být i první zmínka o kameře SR300, která pochází z února roku 2016[19]. Tato kamera má mít lepší vlastnosti než předchozí kamera F200 [19].

Velkou výhodou této kamery je odnímatelný kabel, který je tak možné vyměnit za jiný, například delší. Kamera F200 má zabudovaný kabel USB 3.0, který nebylo možné prodloužit pomocí USB prodlužovacího kabelu. Kamera se s prodlouženým kabelem nechovala korektně kvůli vysokému datovému toku. Obě kamery (F200 a SR300) vyžadují USB 3.0 port, který je přímo napojený na základní desku počítače, a tak není možné používat USB rozbočovače.

Obrázek 1.8: Kamera Intel® RealSense™ SR300 [19]

[7] Od společnosti Intel® jsem obdržel jeden testovací vzorek, a tak jsem mohl i tuto kameru podrobit srovnání.



Obrázek 1.9: Porovnání hloubkové mapy z modelu F200 a SR300 [19]

Název kamery	Intel® RealSense™ SR300
Rozlišení RGB kamery	1080p (1920×1080) : 30 fps
Rozlišení hloubkové kamery	VGA (640×480) : 30 fps
Operační vzdálenost	0,2 m až 1,2 m (5 m)
Výrobce	Intel®
Rok vydání	2016[a]

Tabulka 1.4: Parametry Intel® RealSense™ SR300

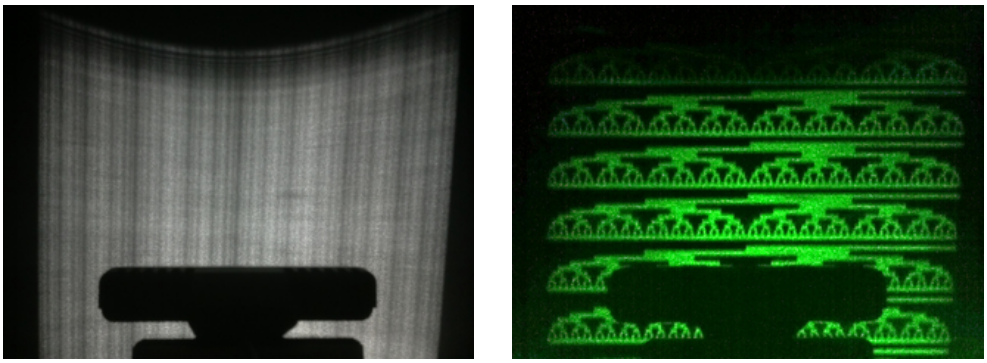
[a]Kamera ještě nebyla oficiálně vydána, k dispozici je pouze balíček pro vývojáře.

1.2.4 Intel® RealSense™ technologie

Na obrázku 1.4 je vidět, jak funguje Xbox Kinect – promítá nepravidelnou síť bodů, kterou následně vyhodnocuje. Kamery RealSense™ fungují na podobném principu, avšak nepromítají síť bodů, ale jiné obrazce. Tyto obrazce nejsou statické, ale pohybují se po vertikální ose.

Výhodou tohoto přístupu je možnost snímat hloubku z většího prostoru. Dochází tak k většímu osvícení, než v případě statických obrazců. Nevýhodou je snížená schopnost rozpoznávat objekty, které se rychle pohybují z pozadí do popředí a naopak. Anglicky se tento problém nazývá *Motion Range Trade Off* [25].

Na obrázku 1.10b je ukázka vzoru, který je rovný, ačkoliv není promítán na rovnou zeď.



(a) Vzor s vysokým časem expozice (b) Vzor s nízkým časem expozice

Obrázek 1.10: Promítané vzory u technologie Intel® RealSense™[26]

Kapitola 2

Příprava dat a modelu

V této kapitole jsou popsány veškéré pojmy, které se v práci opakovaně vyskytují. Zároveň je zde vysvětlen celý proces přípravy a tisku 3D modelu lidské nohy.

2.1 Mračno bodů

Mračno bodů [27][28] definuje 3D objekt pomocí nasnímaných bodů s polohou x, y, z. Přídavnou hodnotou pro tyto body může být směr normály a informace o barvě bodu. Body nejsou nijak propojené. Vizuálně se s mračnem bodů hůře pracuje, jelikož povrch není vyjádřen plošně a viditelnost všech bodů uživatele může mást. Při provádění řezu mračnem bodů se zobrazují v rovině řezu pouze protnuté body. Tento výstupní formát používají hlavně skenery, které snímají velkou oblast kolem sebe.

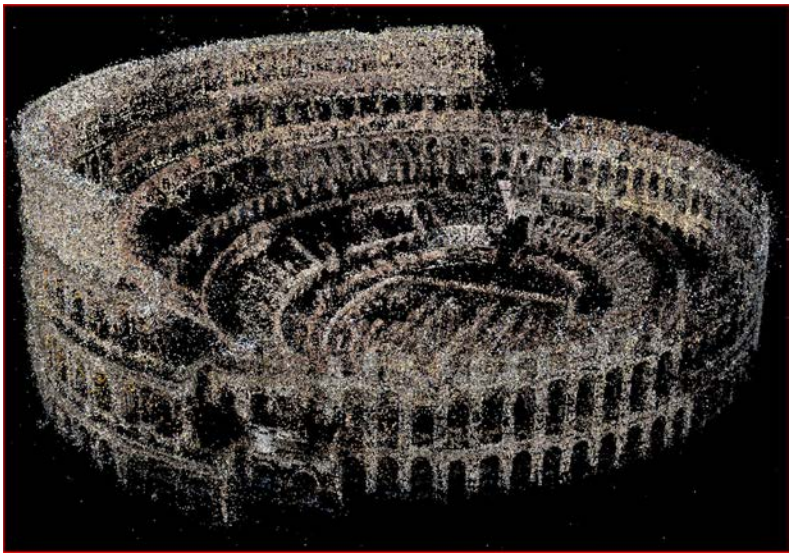
Mračna bodů se běžně nepoužívají v počítačových programech pro práci s 3D grafikou, ale jsou primárně určena pro zpracování a převedení na 3D model, který je pak nadále používán.

2.2 Model

Ve své práci často pracuji s pojmem model. Ale co to vlastně je model? Modelem je myšlena reprezentace určitého objektu nebo systému z určitého úhlu pohledu. V našem případě se jedná o reprezentaci reality v co nejpřesnějším 3D počítačovém modelu.[30] Následující sekce podrobněji vysvětluje, co je to 3D model.

2.2.1 3D model

3D model reprezentuje 3D objekt pomocí kolekce bodů v 3D prostoru. Jednotlivé body v prostoru jsou spojené pomocí geometrických entit jako například



Obrázek 2.1: Mračno bodů [29]

trojúhelníky, čáry, zakřivené plochy. 3D modely mohou být vytvořeny několika způsoby:

1. modely vytvořené ručním modelováním,
   prakticky jakýkoliv model;
2. modely vytvořené algoritmicky,
   modely vzniklé například generováním fraktálovitých útvárů;
3. modely vytvořené naskenováním,
   jakýkoliv objekt, který byl naskenován pomocí 3D skeneru.

2.2.2 Reprezentace 3D modelů

Existují celkem tři nejběžnější způsoby reprezentace počítačového 3D modelu:

1. NURBS,
2. polygonální reprezentace,
3. Splines & Patches.

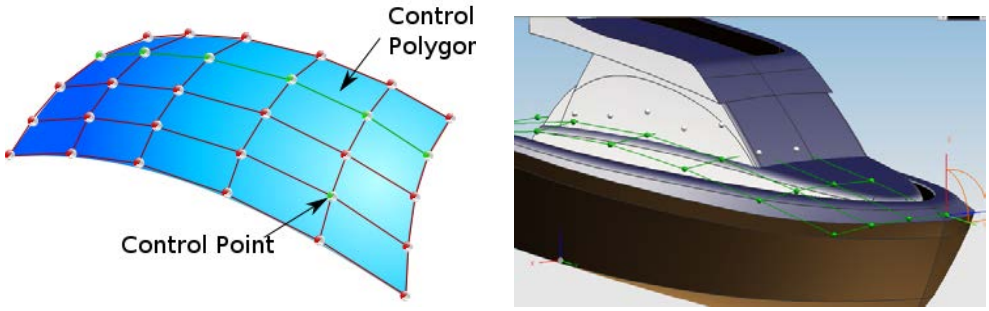
V této práci jsou blíže popsány pouze dva způsoby – NURBS a polygonální reprezentace, které jsou považovány za ty nejpodstatnější. Třetí způsob v této práci popsán není, jelikož se jedná o kombinaci dvou předchozích způsobů. V případě 3D skenování objektů se používá pouze polygonální reprezentace, proto se jí budu věnovat více v následujících sekcích.

2.2.2.1 NURBS

Non-Uniform Rational Basis Spline (NURBS) představuje matematický model v počítačové grafice pro generování a reprezentování křivek a ploch.

Vývoj NURBS začal v padesátých letech 20. století kvůli potřebě matematicky přesně reprezentovat volné tvary jako trupy lodí, vnější povrchy letadel, aeroplánů a karoserie automobilů, aby by mohly být přesně reprodukovány.

NURBS umožňuje reprezentaci geometrických tvarů v kompaktní formě. Může být efektivně ovládán počítačovými programy a přitom umožňuje snadnou lidskou interakci. NURBS povrchy jsou funkce dvou parametrů mapovaných na povrch v třírozměrném prostoru. Tvar povrchu je dán kontrolními body. [31]



(a) Prvky pro ovládání NURBS povrchů (b) NURBS model

Obrázek 2.2: NURBS ukázka[32]

NURBS lze v analogii vektorového a bitmapového obrázku přirovnat k vektorovému obrázku.

2.2.2.2 Polygonální reprezentace

Jedná se o reprezentaci modelu pomocí polygonové sítě [33]. Polygonová síť se skládá z několika částí:

- body (anglicky *vertices*),

  v našem případě body v 3D prostoru;

- hrany (anglicky *edges*),

  spojující jednotlivé body;

- Stěny (anglicky *faces*),

  nejmenší trojúhelníkovitá primitiva mající povrch;

- Polygon (anglicky *polygons*),

  sestavené z jednotlivých stěn;

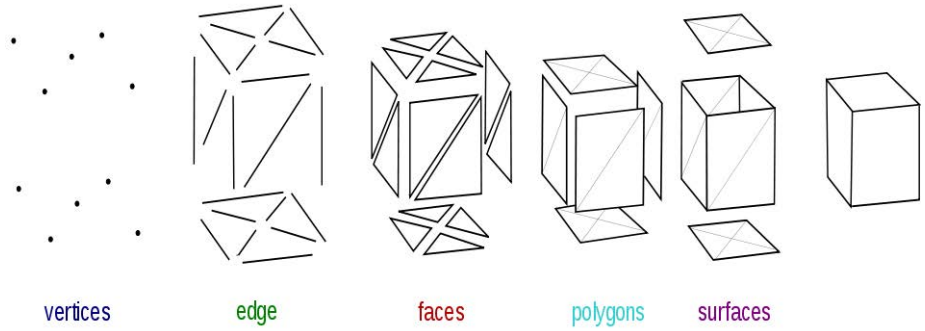
- Plochy (anglicky *surfaces*),

  skládající se z polygonů;

- objekty (anglicky *objects*),

  nejmenší primitiva mající objem.

Stěny jsou většinou trojúhelníkovité, protože trojúhelník je nejmenším 2D objektem, který má povrch a všechny body v jedné rovině. Například u čtverce lze body rozmístit tak, že je není možné protnout jednou rovinou. Polygonová reprezentace vzniká například při CSG modelování. [34]



Obrázek 2.3: Skladba polygonové sítě [35]

Nevýhodou této reprezentace může být nemožnost přesného vyjádření zakřivené plochy. Může se mu pouze přiblížit pomocí vetšího množství polygonů.

Naopak její výhodou je rychlost zpracování modelů pomocí této reprezentace. Oproti jiným reprezentacím (například NURBS 2.2.2.1)dosahuje stejně detailní scéna v pologynové reprezentaci 60 snímků za vteřinu oproti jiným modelům, kde je rychlost pouze okolo 10 snímků za vteřinu.

V analogii k vektorovému a bitmapovému obrázku lze pologynovou síť přirovnat k bitmapovému obrázku.

Existuje několik možností, jak polygonovou síť uložit do souboru. V této práci popíši pouze 3 nejběžnější možnosti ukládaní – STL, OBJ a PLY. Každá má různé odlišnosti a jsou na sebe částečně převoditelné.

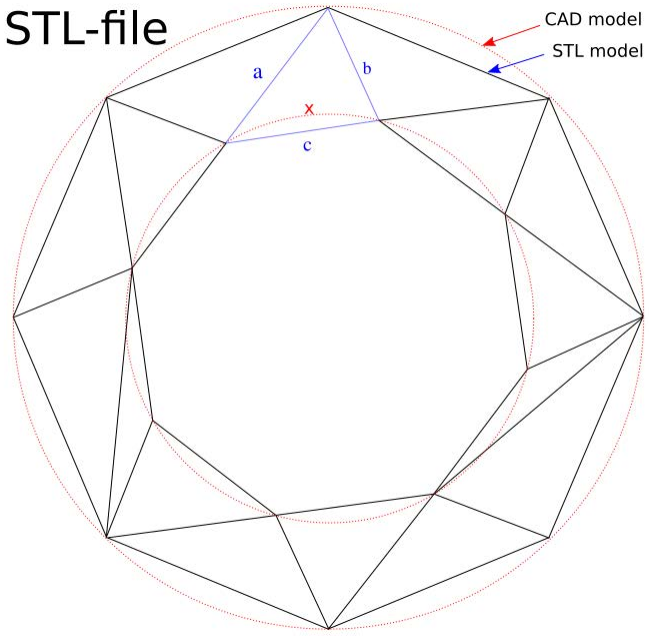
STL

STL je zkratkou pro STereoLitography nebo také Standard Tessellation Language. Tento formát vyvinutý firmou 3D Systems [36] představuje nejjednodušší reprezentaci polygonové sítě. Veškeré stěny, které jsou reprezentovány trojúhelníky, jsou vypsány popořadě za sebou, respektive body, které dané straně náleží, a normálový vektor, který udává rub a líc. Veškerá čísla v souboru jsou bez jednotek, proto je třeba dbát na iterpretaci modelu. Například 1 jednotka může znázorňovat 1 centimetr nebo 1 palec. [35]

Tato reprezentace může být buď ASCII[8] nebo binární. Běžně se používá binární kvůli úspoře místa. Ve standardním STL formátu není možné uchovávat informace o barevnosti modelu.

```
facet normal ni nj nk
    outer loop
        vertex v1x v1y v1z
        vertex v2x v2y v2z
        vertex v3x v3y v3z
    endloop
endfacet
```

Ukázka kódu 2.1: Struktura ASCII STL souboru



Obrázek 2.4: Nedostatky STL formátu [37]

OBJ

Původně vyvinut firmou Wavefront Technologies. Formát je otevřený, takže se velmi rychle ujal i u jiných firem, v programech či v různých odvětvích. Hlavním rozdílem oproti STL souborovému formátu je možnost ukládat i barevnou informaci k modelu. [38] Oproti STL formátu má navíc tyto informace:

[8] ASCII je zkratka pro pro American Standard Code for Information Interchange a jde o kódovou tabulku, která definuje znaky anglické abecedy, a jiné znaky používané v informatice.

- UV pozici pro každý koordinační bod textury,
- normálové vektory jednotlivých bodů.

Formát OBJ neukládá normálový vektor pro jednotlivé stěny. Spoléhá tak na pořadí bodů, ze kterého je možné odvodit normálový vektor podle pravidla pravé ruky[39]. OBJ je možné převést na STL formát, ale bohužel dojde ke ztrátě informace o barevnosti modelu. OBJ reprezentace je pouze ASCII.

PLY
Formát PLY, plným názvem Polygon File Format, vznikl primárně pro potřeby ukládat třídimenzionální data ze 3D skenerů. Reprezentace je velmi podobná STL reprezentaci, liší se pouze drobnými rozdíly – PLY reprezentace podporuje například komentáře.

Nejpodstatnějším rozdílem je ovšem možnost definovat libovolné množství vlastností pro daný soubor. Tímto mechanizmem může soubor PLY obsahovat i barevnou informaci o modelu narozdíl od formátu STL.

PLY reprezentace může být ASCII nebo binární.

2.3 Předloha pro skenování

Pro potřeby skenování jsem ze začátku práce používal svoji nohu jako referenční model. Toto řešení se zdálo jako výhodné, především z důvodu testování přímo na reálné noze a tak maximálnímu přiblížení se reálnému použití. Ukázalo se, že existuje několik důvodů, proč tento postup není nejvhodnější. Všechny tyto body jsem shrnul v následujícím seznamu 3.

1. složité vyhodnocení přesnosti naskenovaných modelů,
2. velké časové prodlevy při experimentech, kdy jsem sám byl předlohou,
3. opakovatelnost.

2.3.1 Složité vyhodnocení přesnosti naskenovaných modelů

Po naskenování své nohy jsem viděl v počítači výsledný sken. Od pohledu se noha podobala mé noze, ale bohužel jsem neměl jak vyhodnotit přesnost. Nabízela se možnost porovnávat skeny stejné nohy z různých skenerů a následně tyto skeny mezi sebou validovat. Nicméně pokud by všechny skenery měly stejnou systematickou chybu, tak by při tomto postupu nebyla odhalena. Další možností bylo svoji nohu naměřit pomocí různých obuvnických nastrojů a výsledky porovnávat se skenem. Při tomto postupu bych mohl narazit na

```
ply
format ascii 1.0           { ascii/binary, format version number }
comment this file is a cube  { comments keyword specified, like all lines }
element vertex 8           { define "vertex" element, 8 of them in file }
property float x           { vertex contains float "x" coordinate }
property float y           { y coordinate is also a vertex property }
property float z~{ z~coordinate, too }
element face 6             { there are 6 "face" elements in the file }
property list uchar int vertex_index { "vertex_indices" is a list of ints }
end_header                 { delimits the end of the header }
0 0 0                      { start of vertex list }
0 0 1
0 1 1
0 1 0
1 0 0
1 0 1
1 1 1
1 1 0
4 0 1 2 3                  { start of face list }
4 7 6 5 4
4 0 4 5 1
4 1 5 6 2
4 2 6 7 3
4 3 7 4 0
```

Obrázek 2.5: Ukázka PLY souboru

problém, že jsem schopen porovnat jenom absolutní rozměry modelu. V případě, že bych měl přesný, ale například částečně zmenšený sken, musel bych všechny naměřené údaje přepočítat.

Mimo to by bylo nutné z modelu v příslušných místech extrahovat rozměry, a to manuálně nebo automaticky. Manuální možnost by byla velmi časově náročná vzhledem k celkovému množství skenů. Automatická možnost by razantně zvětšila rozsah práce, jelikož by bylo nutné navrhnout a implementovat algoritmus, který by byl otestován a vyhodnocen. Až v případě úspěchu by bylo možné tento postup použít.

2.3.2 Velké časové prodlevy při experimentech, kdy jsem sám byl předlohou

Při skenování bylo nezbytné přecházet mezi počítačem a kamerou. Zároveň jsem nemohl pozorně sledovat průběh skenování, protože jsem se soustředil na

samotné skenování. Řešením by byl lidský model, který by mi byl předlohou pro skenování, ale to by bylo časově neefektivní, především v době vývoje.

2.3.3 Opakovatelnost

V případě, že by na práci v budoucnu chtěl někdo navázat a chtěl by srovnávat výsledky ze svých skenů s mými skeny, narazil by na problém, že mám jinak velké a tvarované nohy. Tudíž by nebylo co porovnávat. Zároveň kdybych si během vývoje poranil nohu nebo by mi například pouze otekla, tak by výsledky před a po úrazu byly jiné.

2.3.4 Tvorba předlohy

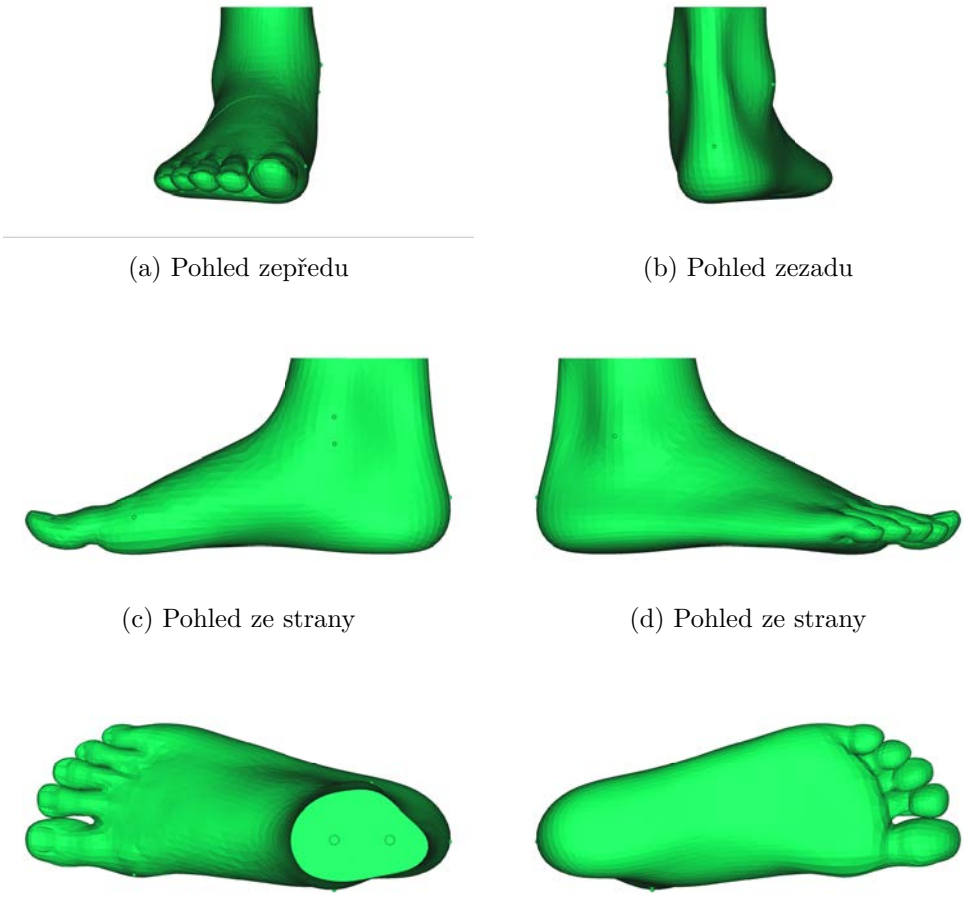
Na základě výše uvedených důvodů jsem se rozhodl pro vytvoření předlohy, kterou podrobím skenovat. Abych vyřešil všechny předchozí problémy, navrhl jsem následující postup:

1. Vznik počítačového modelu nohy.
2. Příprava pro 3D tisk.
3. Tisk na 3D tiskárně.
4. Skenování modelu.
5. Porovnání naskenovaného modelu s prvnotním počítačovým modelem.

Jelikož model lidské nohy je volně dostupný, bylo by zbytečné v rámci této práce modelovat lidskou nohu. Použil jsem model z webové databáze 3D modelů Thingiverse.[40] Jedná se o model celého člověka v životní velikosti, který je předpřipravený pro 3D tisk – je rozdělený na více částí[41]. Z tohoto modelu jsem zvolil pravou nohu. Rozhodl jsem se vytisknout model nohy nad kotník a k tomu část mezi kotníkem a kolenem - tato část je volitelná a sloužila při testování, abych mohl ověřit, že nedochází ke zkreslení postupu kvůli zakončení nohy nad kotníkem.

Celkem jsem vytiskl 4 části z celkového modelu.

- *R_Leg_1.stl*
- *R_Leg_2.stl*
- *R_Leg_3.stl*
- *R_Leg_4.stl*



(a) Pohled zepředu

(b) Pohled zezadu

(c) Pohled ze strany

(d) Pohled ze strany

(e) Pohled shora

(f) Pohled zespodu

Obrázek 2.6: Ukázka 3D modelu R_Leg_1.stl a R_Leg_2.stl spojeného dohromady

2.3.5 Tisk a sestavení předlohy

Pro výrobu nohy jsem se rozhodl použít 3D tisk. Konkrétně technologii tisknutí termoplastem (FDM). Dále bylo potřeba zvolit vhodný materiál. K dipozici bylo několik běžných materiálů používaných ve 3D tisku pro FDM technologii[42].

ABS Výsledný model bude pevný, ale materiál se nehodí pro větší objekty kvůli velkému smršťování.

PLA Výsledný model bude křehčí, nicméně se plast téměř nesmršťuje, proto není problém vytisknout velký model.

PET-G Model bude pevný jako z materiálu ABS, ale tisknout se bude jako PLA, tudíž se nebude tolik smršťovat. Nevýhodou je vyšší cena.

HIPS Vhodný převážně pro podpůrný materiál díky tomu, že je rozpustný v lemonesolu [9].

Po pečlivém úvažení jsem vybral materiál PLA, především kvůli možnosti tisknout velké objekty a kvůli nižší ceně v případě neúspěšného tisku. Pro přípravu modelu pro tisk jsem zvolil nástoj Cura for Lulzbot [44]. Jedná se o nástroj vyvinutý pro 3D tiskárny Lulzbot. Pro 3D tisk jsem použil tiskárnu Lulzbot Taz 6 [45].

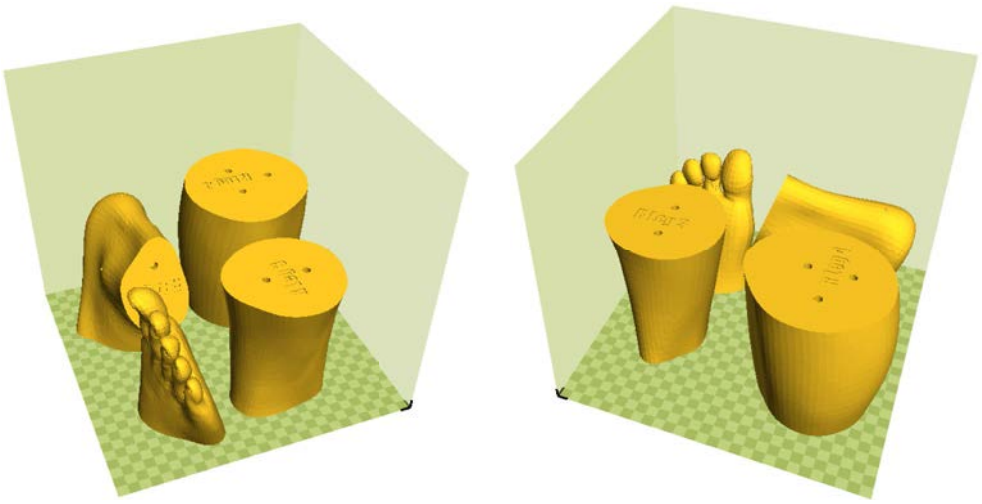
Parametry tisku jsem zvolil následující:

- Výška vrstvy: 0,2 mm
- Počet obvodových stěn: 3
- Procentuální výplň: 15%
- Rychlost tisku: 30mm/s

Celková doba tisku byla 108 hodin. Bohužel po 80 hodinách došlo k odlepení z tiskové podložky a musel jsem začít tisknout model od začátku. Při druhém pokusu o tisk jsem tiskl jednotlivé díly postupně – kdyby došlo k chybě během tisku, aby se nepokazily opět všechny čtyři díly najednou. Celkově jsem se v druhém pokusu dostal na 96 hodin čistého tiskového času.

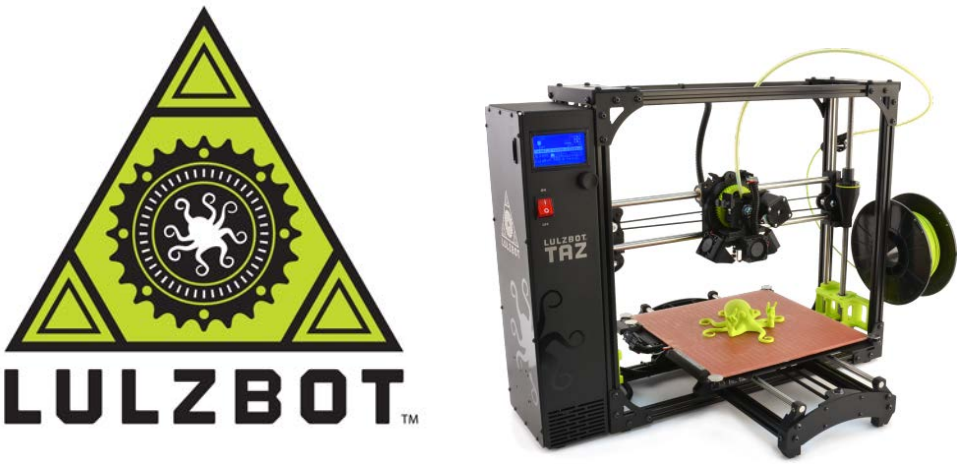
Vytištěné modely jsem následně opatřil upevňovacími kolíky a slepil poloviny spodní a poloviny horní části k sobě. Samotné části jsem již k sobě nelepil, abych mohl skenovat jak s horní částí, tak bez ní.

[9] Lemonesol je typ rozpouštědla, které je vhodné právě pro plasty typu HIPS, např. [43]



(a) Tiskový plát – pohled zepředu (b) Tiskový plát – pohled zezadu

Obrázek 2.7: Přípravy pro 3D tisk v aplikaci Cura for Lulzbot



(a) Logo produktu (b) Fotografie tiskárny

Obrázek 2.8: Lulzbot logo a tiskárna TAZ 6 [45]

2.3.6 Ukázka sestavené nohy

V této sekci je představena vytištěná noha – jak část končící u kotníku, tak i část mezi kotníkem a kolenem. Horní(žlutá) a spodní(šedá) část je záměrně slepena pouze dočasně, aby bylo možné skenovat pouze spodní část nebo obě části.

(a) Pohled zepředu

(b) Pohled zezadu

(c) Pohled ze strany

(d) Pohled ze strany

(e) Pohled shora

(f) Pohled zespodu

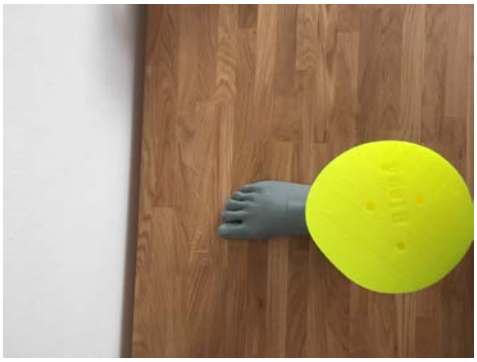
Obrázek 2.9: Ukázka vytištěného modelu nad kotník

(a) Pohled zepředu

(b) Pohled zezadu

(c) Pohled ze strany

(d) Pohled ze strany



(e) Pohled shora

(f) Pohled zespodu

Obrázek 2.10: Ukázka vytištěného modelu pod koleno

Část II

Implementační část

Kapitola 3

Skenování nohy

Před samotným skenováním bylo nutné zvolit způsob, jakým budou z jednotlivých kamer čtena data a o jaké data se bude vlastně jednat – zda půjde pouze o hloubkové mapy, mračna bodů nebo hotové 3D modely.

Bohužel v současné chvíli neexistuje zcela unifikované řešení pro vyčítání dat z kamer a následné zpracování. Bylo by zajímavé se pokusit něco takového vyvinout, ale bohužel vzhledem k množství kamer, k rychlosti, s jakou vznikají nové kamery, a k existenci proprietárních vývojových balíčku (anglicky SDK) je takový úkol takřka nemožný.

3.1 Vyčítání dat z kamer

Tato sekce se věnuje různým možnostem, jak vyčítat data z kamer, které jsem zvolil pro srovnání.

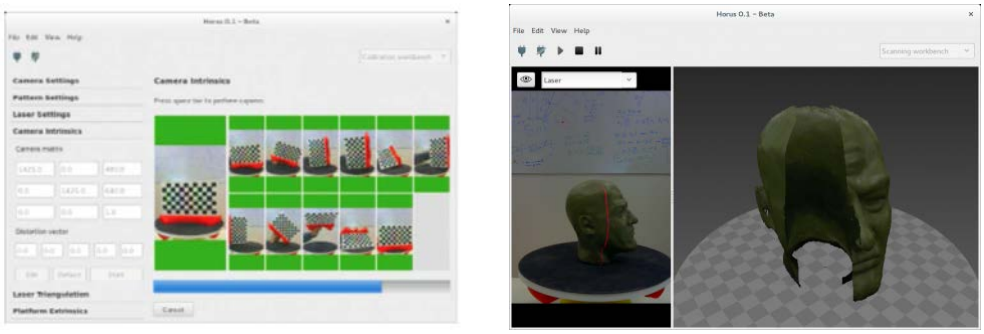
3.1.1 Horus

Jedná se o open-source program přímo pro komunikaci se 3D skenerem Ciclop[16]. Ve skeneru může být jakákoliv webkamera, která musí být viditelná operačním systémem.

Také lze vyvinout kompletně vlastní řešení na rozpoznávání čáry a ovládání laserových projektorů. Vzhledem k tomu, že se nejedná o hloubkovou kameru, tak jsem od této cesty upustil. Proto jsem veškeré skeny tímto skenerem realizoval pomocí programu Horus. Pokud by se ukázalo, že se jedná o nejvhodnější skener, lze z open-source projektu vyjít pro reálný program. Výstupem z této aplikace je mračno bodů.

3.1.2 Xbox SDK

Jedná se o vývojové prostředí pro Xbox přímo od společnosti Microsoft, konkrétně jde o plugin do Microsoft Visual Studia. Je potřeba mít nainstalované



(a) Kalibrace (b) Proces skenování

Obrázek 3.1: Ukázka programu Horus[16]

i Visual Studio, aby bylo možné vyvíjet aplikace pro Xbox. Ze všech možných přístupů se jedná o nejnižší možnou vrstvu. Výstupem z této aplikace může být jak mračno bodů, tak 3D model, a to z důvodu chybějící vrstvy, která by realizovala skenování. (Výstup je potřeba tudíž naprogramovat)

3.1.3 OpenNI 2.0

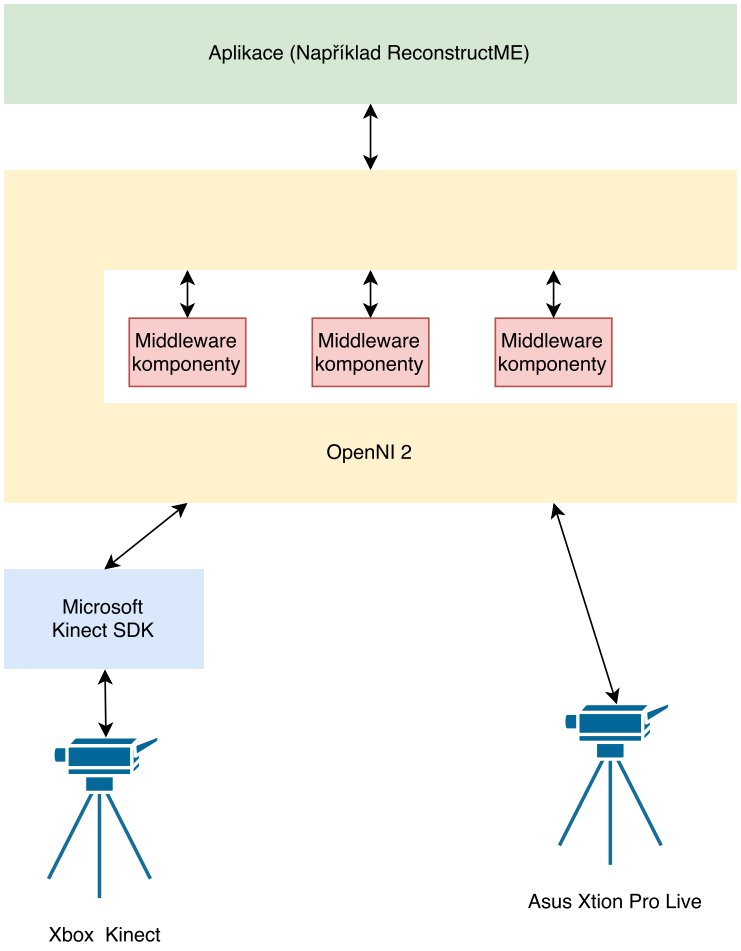
OpenNI 2.0[46] je projekt s univerzálním přístupem ke kamerám a následnému zpracování dat. OpenNI 2.0 obsahuje rozhraní, které lze jednoduše rozšířit i o další middleware komponenty, například o komponentu pro rozpoznávání gest ruky.

Nevýhodou je, že tento projekt přestává být vyvíjen[47]. Firma, která projekt spravuje, momentálně investuje svůj čas do vývoje vlastní hloubkove kamery (Stucture Sensor [48]) a souběžně připravuje i vývojářský balíček[49], který je velmi podobný OpenNI 2.0 s tím rozdílem, že místo podpory různých kamer podporuje pouze jejich vlastní kameru.

Výstupem z této aplikace může být jak mračno bodů, tak 3D model, a to z důvodu chybějící vrstvy, která by realizovala skenování. (Výstup je potřeba tudíž naprogramovat)

3.1.4 ReconstructMe

Jedná se o skenovací software postavený na OpenNI 2.0 – všechny kamery, které jsou projektem OpenNI 2.0 podporovány, budou fungovat i zde [50]. Program ReconstructMe má i SDK, díky kterému je možné vyvíjet aplikace na vyšší vrstvě než v případě OpenNI 2.0, viz. obrázek 3.2. Výstupem z této aplikace je 3D model.



Obrázek 3.2: Diagram skenovacích možností pro kamery Xbox Kinect a Asus Xtion Pro Live

3.1.5 RealSense™ SDK

Jedná se o vývojový balíček přímo od společnosti Intel® pro programování pro jejich hloubkové kamery. Když vyjde nová kamera, je toto SDK vždy aktuální a zároveň existuje internetové fórum [51], kde Intel® odpovídá na problémy vývojářů s tímto SDK.

Nevýhodou je podpora pouze operačního systému Windows® 10 64-bit, tudíž vývojáři na jiných platformách nemají možnost toto SDK použít. Zatím to nevypadá, že by se tato situace měla v blízké budoucnosti změnit.

Další nevýhovou, se kterou jsem se setkal během vývoje, je časté vydávání nových verzí. Několikrát jsem řešil zpětnou nekompatibilitu a úpravu kódu, abych práci udržel aktuální.

Výstupem z této aplikace může být mračno bodů nebo 3D model.

3.1.6 Intel® RealSense™ Cross Platform API

Jedná se o oficiální produkt společnosti Intel®. Hlavní rozdíl oproti RealSense™ SDK je v tom, že se nejedná o vývojářský balíček, ale pouze o drivery pro vyčítání dat z kamery. Jak již název napovídá, má tento produkt podporu i na jiných operačních systémech než pouze pro Windows® 10 64-bit. Funguje tedy i na macOS a Linuxu[52].

Ukažme si jako modelový příklad aplikaci, která se rozhodne pro své ovládání používat gesta rukou. V případě, že stačí, aby cílová platforma byla pouze Windows, aplikacé může klidně využít RealSense™ SDK 3.1.5, kde již je částečně předpřipravené rozpoznávání gest. V případě, že by aplikace, měla fungovat i na operačních systémech macOS a Linux, bylo by nutné použít Cross Platform API, kde ovšem není žádná příprava pro rozpoznávání rukou a programátor tak musí začít vyvíjet toto řešení od začátku. To znamená rozpoznávání rukou, kloubů, prstů a následně i gest.

Výstupem z této aplikace může být jak mračno bodů, tak 3D model, a to z důvodu chybějící vrstvy, která by realizovala skenování. (Výstup je potřeba tudíž naprogramovat)

3.1.7 Výsledné řešení vyčítání dat

Pro vyčítání dat z Ciclop 3D Scanneru jsem zvolil program Horus, protože je to open source program s možností využití jeho vývojového balíčku v případě vývoje vlastní aplikace postavené nad tímto programem.

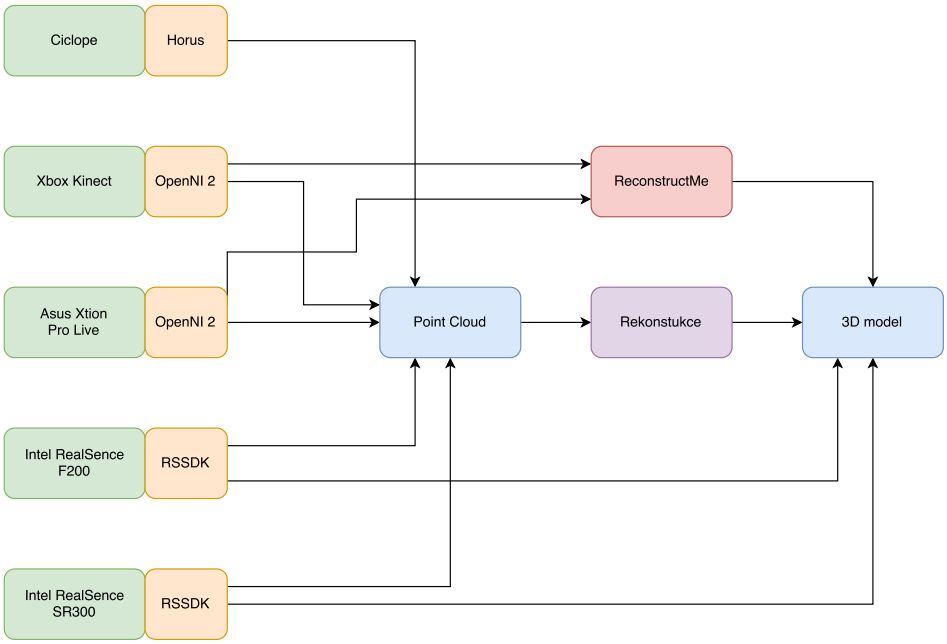
V případě kamer Kinect a ASUS Xtion PRO LIVE jsem chtěl zvolit něco unifikovaného pro obě kamery, proto jsem se rozhodl pro řešení postavené nad OpenNI 2.0. Nicméně jsem nezvolil přímo program ReconstructMe, který má také vlastní vývojový balíček.

Pro kamery Intel® RealSense™ jsem zvolil RealSense™ SDK i za cenu toho, že je pak vývoj možný pouze na platformě Windows. Naštěstí se jedná o velmi populární platformu[53], takže to není tolik limitující. Hlavním důvodem, proč jsem takto volil, je možnost vyzkoušení veškerých funkcí, kterých jsou kamery schopny, narozdíl od Intel® RealSense™ Cross Platform API.

Na diagramu 3.3 je vidět, jak jsem se rozhodl data z kamer vyčítat a zda je výstupem z kamery pouze mračno bodů nebo hotový 3D model.

3.2 Srovnání skenování s více kamerami oproti skenování s jednou kamerou

Původně jsem chtěl v této práci také srovnat skenování s jednou hloubkovou kamerou oproti více kamerám, které snímají stejný objekt z více úhlů. Narazil jsem však na problém, že se více kamer navzájem ruší. To je způsobené tím, že každá kamera promítá obrazce, které pak snímá. Když dvě kamery promítají obrazce na stejný objekt, dochází pak k rušení [54].



Obrázek 3.3: Diagram skenovacích možností

Jistě existují techniky, jak snímat objekt pomocí více kamer. Tento úkol není tedy nereálný, svojí realizací však přesahuje rozsah této práce. Jednalo by se například o multiplexování promítaných bodů či o nějakou modulaci v čase, kdy by se kamery z různých úhlů pravidelně střídaly v promítání. Následně by vyvstaly problémy ohledně synchronizace času mračen bodů apod. Nicméně se jedná o zajímavé téma vhodné k dalšímu bádání.

3.3 Srovnání kamer mezi sebou

Tato sekce je věnována srovnání jednotlivých kamer mezi sebou a problémům, které při srovnávání vyvstaly.

3.3.1 Problémy s Intel® RealSense™

Problémy s kamerami Intel® RealSense™ začaly již hned při pokusu o instalaci vývojového prostředí (dále jen SDK). SDK funguje pouze na platformě Windows, takže není možné vyvíjet na operačním systému macOS, který běžně používám. Proto bylo nezbytné zajistit počítač s Windows, kdy se ukázalo, že minimální požadavky na hardware jsou relativné přísné[19]:

- procesor alespoň 6th Generation Intel® Core™ Processor,

  ekvivalentem je Intel® Core i7-6700T – SkyLake;

- USB konektivita alespoň USB 3.0 s přímým připojením na základní desku počítače;
- operační systém Windows® 10 64-bit;
- operační paměť alespoň 4 GB.

Po zajištění odpovídajícího hardware nastaly komplikace s intalací samotného SDK. SDK vyžaduje, aby byl na počítači nainstalovaný tzv. Media Feature Pack, jinak nedojde k započetí instalace SDK. Relativně rychle jsem nalezl Media Feature Pack na strankách a nainstaloval [55]. Bohužel systém Windows hlásil stále absenci tohoto balíčku. Poté jsem zjistil, že existuje i druhý balíček, který má stejný název i číslo, ale je trochu jiný. Bohužel s tím to také nešlo. Nakonec jsem objevil i třetí balíček, u kterého se ukázalo, že je to ten správný a SDK se podařilo nainstalovat.

Horší zjištění však následovalo – s kamerou od společnosti Intel® se nedařilo navázat spojení. Později se ukázalo, s pomocí Intel® online fóra[51], že některé počítače nejsou kompatibilní s kamerami. Takže jsem vyzkoušel jiný počítač, se kterým se již ke kameře dalo připojit. Všechna tato zjištění byla velmi časově náročná, protože komunita ohledně této technologie je zatím relativně malá.

3.3.2 Srovnávací kritéria

Bylo nutné vymyslet, jak jednotlivé kamery mezi sebou porovnávat, aby bylo možné rozhodnout, která je pro potřeby této práce nejlepší. Stanovil jsem tedy několik bodů, podle kterých jsem kamery srovnával. Každý bod je hodnocen od 0 do 10, kde 0 znamená nejhorší a 10 nejlepší výsledek. V tomto srovnání jsou jednotlivá kritéria stejně důležitá a mají tedy stejnou váhu.

Jednotlivá srovnávací kritéria:

- porovnání naskenované nohy s předlohou,

  např. zda naskenovaná noha není 1,5× větší než originál apod.;
- práce s kamerou,

  zda např. není problém, že se kamera nesmí pohnout ani o milimetr během skenování apod.;
- kvalita skenu,

  např. v oblasti prstů – zda se podařilo zaznamenat různé detaily;
- použitelnost pro tuto práci,

  např. jestli není nutná speciální podložka, která by zmenožnila, aby skenování prováděl běžný uživatel doma.

Níže jsou popsány postřehy z každého skenování, následované tabulkou s výsledky, a závěrem.

3.3.3 Ciclop + Horus

Skenování probíhalo podle plánu. Software se podařilo nainstalovat bez problémů. Podle návodu ke skeneru byla provedena jeho kalibrace[16]. Následně jsem začal se skenováním. Skenovaný objekt musí být na točící se podložce. Skener není určený k zatížení o váze lidského těla, proto by v dodávaném stavu nemohl být použitelný v průmyslu. Nicméně pokud by se ukázal jako nejlepší, není problém tuto technologi přenést na skener, který lidské tělo unese.

Problém nastal s aktuálním nastavením skeneru na velikost nohy. Model nohy odpovídá velikosti 44 (EU)[56], což činilo problém při otáčení objektu - pokud byl model přesně na středu, tak chyběl kousek prstů vpředu a kousek paty vzdadu. V případě posunutí byla kompletní jedna část na úkor druhé.

Obrázek 3.4: Ukázka skenu z kamery Ciclope

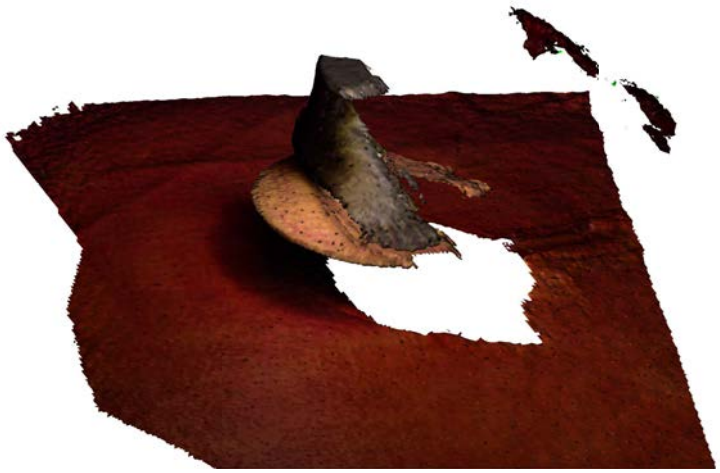
3.3.4 Windows Xbox Kinect + ReconstructMe

Skenování zde začalo podle plánu. Nainstaloval jsem vše potřebné, zdržela mě pouze instalace Xbox SDK for Windows, která existuje v několika verzích a kromě SDK je potřeba i samostatný driver.

Následně jsem začal provádět 3D skenování pomocí programu Reconstruct Me. Bohužel se mi nedařilo udělat žádný souvislý sken tak, abych měl použitelný 3D model nebo alespoň souvislou síť bodů. Možností by bylo udělat více statických pohledů, z každého by bylo výstupem částečné mračno bodů a ta poté ručně složit dohromady. Tomuto jsem se chtěl vyhnout, aby byl proces co nejjednodušší a obsahoval co nejméně manuálních kroků.

Zkoušel jsem točení objektem okolo své osy se zafixovanou kamerou na stativu a točení kamerou okolo zafixováneho objektu. Bohužel výsledek byl v obou případech ne moc dobrý.

Z tohoto důvodu jsem prozatím od Xbox Kinectu upustil a věnoval jsem se dalším skenerům. V případě, že by se ukázalo, že je to tak se všemi ostatními, bylo by nutné znovu zopakovat skenování za pomoci statických pohledů a skládání mračen bodů.



Obrázek 3.5: Ukázka skenu z kamery Xbox kinect

3.3.5 ASUS Xtion PRO LIVE + ReconstructMe

Zde proces skenování začal podobně jako u Windows Xbox Kinectu, pouze nebyl takový problém s drivery a nebylo potřeba žádné další SDK. S objektem jsem při skenování točil okolo své osy před kamerou, která ho snímala. Kvalita skenu byla od počátku větší, než z Windows Xbox Kinectu. Podařilo se mi udělat několik skenů, které byly použitelné po závěrečné srovnání.

3.3.6 Intel® RealSense™ F200 + RealSense™ SDK

Zde byl problém s chybějícím programem pro skenování. Pomocí dokumentace k RealSense™ SDK a výchozích programů jsem naprogramoval jednoduchý program pro skenování pomocí této kamery, který se nachází na přiloženém CD. Zkoušel jsem skenovat opět s fixní kamerou a točící se předlohou a naopak. Lepších výsledků jsem dosahoval, když jsem kamerou točil okolo fixního předmětu. Z kamery jsem měl díky vlastnímu programu možnost dostat buď

Obrázek 3.6: Ukázka skenu z kamery ASUS Xtion Pro Live

mračno bodů nebo hotový uzavřený 3D model, který má objem. Skeny se jevily kvalitněji než u předchozích kamer.

Obrázek 3.7: Ukázka skenu z kamery Intel® RealSense™ F200

3.3.7 Intel® RealSense™ SR300 + RealSense™ SDK

Zde jsem použil program určený pro kameru F200 s drobnými modifikacemi. Kamera má mít lepší vlastnosti na snímání hloubky, viz 1.9. Kamera má velmi podobné výsledky jako kamera F200.

Obrázek 3.8: Ukázka skenu z kamery Intel® RealSense™ SR300

3.3.8 Závěrečné srovnání kamer

Kamery a skenování byly vyzkoušeny v různých podmínkách tak, abych nalezl ideální parametry skenování. Například u světelných podmínek jsem zjistil, že nemají na výsledek skenování vliv (kromě naskenované textury, která není pro ševcovské kopyto potřebná). Problémem je pouze denní světlo, které ruší infračervené kamery.

Některá hodnocení v tabulce jsou velmi nízká z následujících důvodů. Kamera Ciclop dostala málo bodů za použitelnost pro práci především kvůli tomu, že je nutné, aby objekt byl na točící se podstavě, která ovšem neunese lidskou nohu. Zároveň ohnisko skeneru je menší, než by bylo potřeba, a proto se stává, že u větších nohou mohou chybět prsty nebo pata.

Kamera ASUS Xtion PRO LIVE dostala v použitelnosti pro práci dokonce 0 bodů a to z toho důvodu, že se projekt už dále nevyvíjí. Takže přestože má dobré výsledky, je zcela nevhodná pro použití v reálném provozu.

Kamery Intel® RealSense™ mají také nízký počet bodů v použitelnosti pro práci. To je však z důvodu zohlednění aktuálního stavu projektu, který je ve stádiu náhledu pro vývojáře, takže se kamery nedají využít v komerečním

	Porovnání naskenované nohy s předlohou	Práce s kamerou	Kvalita skenu	Použitelnost pro práci	Celkový počet bodů	Pořadí
Ciclop 3D Scanner	4	7	5	3	19	4.
Windows Xbox Kinect	5	4	3	2	14	5.
ASUS Xtion PRO LIVE	7	7	7	0	21	3.
Intel® RealSense™ F200	9	7	9	4	29	2.
Intel® RealSense™ SR300	9	8	9	5	31	1.

Tabulka 3.1: Výsledek srovnání kamer

projektu. Je pouze otázkou času, kdy se toto změní. Kamera SR300 dostala o jeden bod více oproti kameře F200, kvůli možnosti výměně kabelu.

Z tohoto srovnání vzešlo, že se pro práci nejlépe hodí hloubkové kamery Intel® RealSense, konkrétně kamera SR300, která vykazuje nepatrně lepší výsledky.

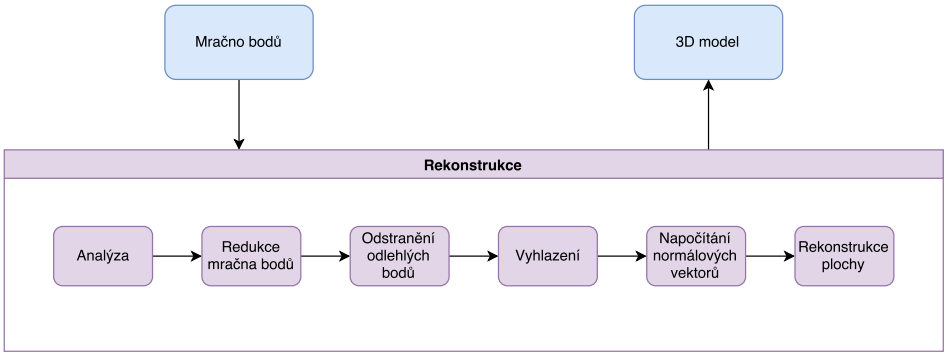
Kapitola 4

Rekonstrukce

V této kapitole je popsán převod mračna bodů na 3D model, tzv. rekonstrukce. Tento proces je důležitý především u skenerů, respektive programů, skrze které se ke skenerům přistupuje. Některé programy nemají možnost ze skenu vytvořit přímo hotový model. Pokud je možné z programu získat přímo hotový 3D model, má smysl zabývat se rekonstrukcí.

Pro implementaci níže zmíněných postupů jsem se rozhodl využít vývojový balíček programu MeshLab [57]. Jedná se o open-source program, který umožňuje zpracovávat a upravovat mračna bodů.

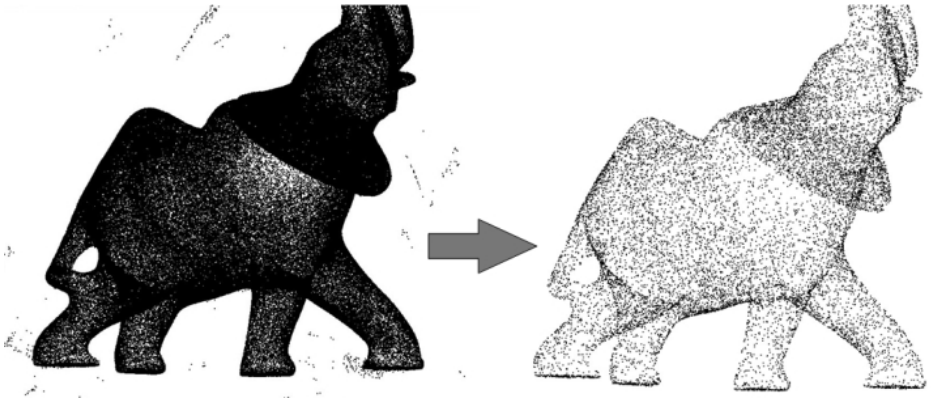
Proces rekonstrukce jsem shrnul v diagramu 4.1. Následující sekce se podrobně zabývají jednotlivými kroky, společně s motivací.



Obrázek 4.1: Diagram procesu rekonstrukce[58]

4.1 Analýza

Nejdříve je nutné provést analýzu, co bude nezbytné s mračnem bodů udělat. Například pokud budeme mít velmi řídké mračno bodů, není potřeba body redukovat. Stejně tak pokud máme normálové vektory, není třeba je zahazovat a znovu počítat.



Obrázek 4.2: Ukázka před započetím procesu a těsně před rekonstrukcí plochy

Při využívání 3D skenerů se ukázalo, že obsahujé velké množství bodů. Kamery Intel® RealSense™ mají řádově statisíce bodů – sken lidské nohy má okolo 300 000 bodů. V takových případech dává smysl snížení počtu bodů.

Pro komunikaci s vývojovým balíčkem programu MeshLab se používají speciální datové struktury, viz 4.1. Jedná se o iterátory přes množinu bodů nebo přes množinu bodů s normálovým vektorem.

```
std::pair<Point_3<K>
Vector_3<K>>
boost::tuple<..,Point_3<K>, ..., Vector_3<K> >.
```

Ukázka kódu 4.1: Ukázka speciálních datových struktur

V rámci analýzy můžeme zjistit průměrnou vzdálenost přes včechny body pomocí metody k-nejbližších sousedů[10] (dále jen k-NN), k je potřeba specifikovat explicitně. Tato informace je užitečná během celého procesu, například pro hustotu výstupní sítě 3D modelu.

4.2 Redukce mračna bodů

Vzhledem k velkému počtu bodů bude velký i výpočetní čas. Z tohoto důvodu je nutná redukce (neboli zjednodušení) počtu bodů tak, aby i při menším počtu byla zachována stejná kvalita a optimální výpočetní čas. Existuje několik možností redukce mračna bodů.

[10] Jedná se o algoritmus strojového učení, konkrétně učení s učitelem pro klasifikační úlohy. Více o algoritmu v [59][60].

```
int main(int argc, char*argv[])
{
    const char* fname = argv[1];
    // Reads a .xyz point set file in points.
    // As the point is the second element of the tuple (that is with index 1)
    // we use a property map that accesses the 1st element of the tuple.

    std::vector<IndexedPointWithColorTuple> points;
    std::ifstream stream(fname);
    if (!stream ||
        !CGAL::read_xyz_points(
            stream, std::back_inserter(points),
            CGAL::Nth_of_tuple_property_map<1,IndexedPointWithColorTuple>()))
    {
      std::cerr << "Error: cannot read file " << fname << std::endl;
      return EXIT_FAILURE;
    }

    for(unsigned int i = 0; i < points.size(); i++)
    {
      points[i].get<0>() = i;   // set index value of tuple to i
    }

    // Computes average spacing.
    const unsigned int nb_neighbors = 6;
    FT average_spacing = CGAL::compute_average_spacing<Concurrency_tag>(
        points.begin(), points.end(),
        CGAL::Nth_of_tuple_property_map<1,IndexedPointWithColorTuple>(),
        nb_neighbors);
    std::cout << "Average spacing: " << average_spacing << std::endl;
    return EXIT_SUCCESS;
}
```

Ukázka kódu 4.2: Program pro načtení souboru a výpočet průměrné hustoty bodů

4.2.1 Náhodná redukce

Jednou z možností, jak řešit zjednodušení sítě bodů, je náhodné odebrání předem daného procenta bodů. Tento algoritmus však nereflektuje, zda body odebírá z části mračna, kde je hustá nebo řídká koncentrace. Algoritmus je díky své jednoduchosti velmi rychlý.

4.2.2 Mřížkové odstranění

Mračno bodů je rozděleno do stejně velkých sekcí (kvádrů). Následně je z každé sekce odebrán libovolný bod. Tento postup je časově náročnější než náhodná redukce.

```
// simplification by clustering using erase-remove idiom
double cell_size = 0.001;
points.erase(CGAL::grid_simplify_point_set(points.begin(),
        points.end(),
        cell_size),
        points.end());
```

Ukázka kódu 4.3: Ukázka programu pro mřížkovou redukci

4.2.3 Weighted Locally Optimal Projection

Zkráceně pouze WLOP. Jedná se o implementaci algoritmu [61], který se kromě redukce zároveň snaží o rovnoměrné rozdělní zbývajících bodů. Algoritmus má dva pro nás nejzajímavější parametry – rádius sousedů a informaci, zda je potřeba uniformní vzorkování.

4.2.3.1 Rádius sousedů

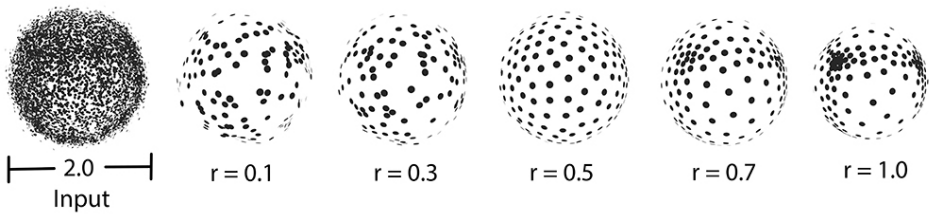
Dle doporučeného nastavení by měl každý bod obsahovat alespoň dva rádiusy ostatních bodů. V případě malého rádiusu souseda hrozí, že vygenerovaný výsledek nebude rovnoměrný. Při velkém rádiusu hrozí stejný problém, ale z jiného důvodu – body se smrsknou do středu lokální plochy (anglicky underfitting). Pokud se tato hodnota nastaví jako záporná, algoritmus se pokusí odhadnout ideální hodnotu. Ukázka vlivu tohoto parametru je vidět na obrázku 4.3.

4.2.3.2 Uniformní vzorkování

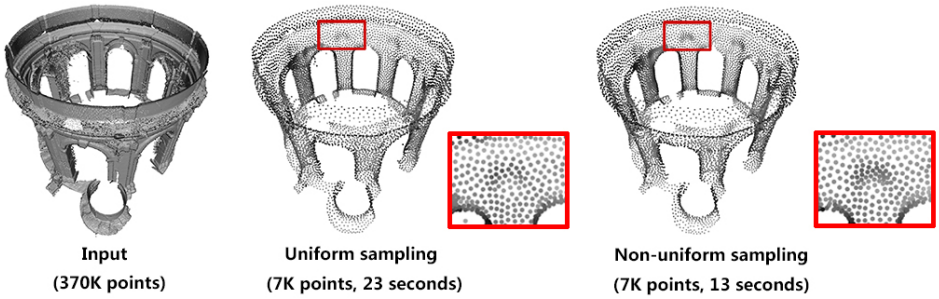
Pokud je proměnná require_uniform_sampling nastavena na hodnotu true, WLOP je odolný proti nerovnoměrnému vzorkování bodů a generuje vzorkované body s rovnoměrným rozdělením na úkor výpočetního času. Totéž funguje i naopak. [62][61] Ukázka vlivu tohoto parametru je vidět na obrázku 4.4.

```
// percentage of points to retain.
const double retain_percentage = 2;
// neighbors size.
const double neighbor_radius = 0.5;
  CGAL::wlop_simplify_and_regularize_point_set
                          <Concurrency_tag>
                          (points.begin(),
                           points.end(),
                           std::back_inserter(output),
                           retain_percentage,
                           neighbor_radius,
                        false
                           );
```

Ukázka kódu 4.4: Ukázka WLOP algoritmu



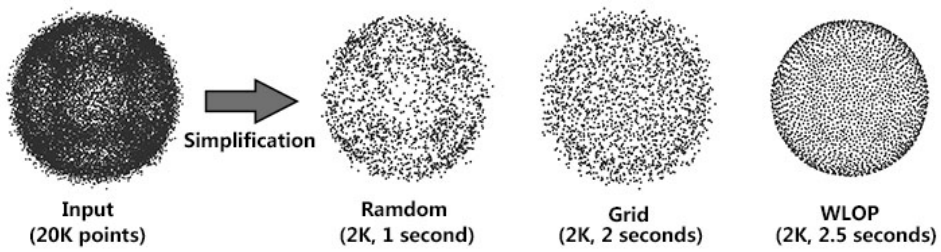
Obrázek 4.3: Různé nastavení pro rádius sousedů[58]



Obrázek 4.4: Ukázka algoritmu s uniformním vzorkováním a bez něj[58]

4.2.4 Srovnání redukcí

Vyzkoušel jsem tyto algoritmy a jako nejlepší se jeví WLOP. Různé vlastnosti algoritmů jsou nejlépe vidět na obrázku 4.5.



Obrázek 4.5: Srovnání náhodného zjednodušení, mřížkového a WLOP[58]

4.3 Odstranění odlehlých bodů

Odstranění odlehlých bodů je vhodné, pokud sken obsahuje body, které k modelu nepřísluší, tzn. různý šum, částí jiných objektů apod. Pro odstranění se opět požívá metoda k-nejbližších sousedů [59]. Je potřeba ručně specifikovat, kolik procent bodů bude odstraněno. Následně se pro každý bod spočítá průměrná čtvercová vzdálenost ke k nejbližším sousedům. Body se pak seřadí v klesajícím pořadí a prvních x % se odstraní.

```
// percentage of points to remove
const double removed_percentage = 5.0;
// considers 24 nearest neighbor points
const int nb_neighbors = 24;
points.erase(CGAL::remove_outliers(points.begin(), points.end(),
        CGAL::Identity_property_map<Point>(),
         nb_neighbors, removed_percentage),
        points.end());
```

Ukázka kódu 4.5: Ukázka programu po odstranění odlehlých bodů

4.4 Vyhlazení

Pro vyhlazení opět existuje několik možností.

4.4.1 Jet Smoothing

Algoritmus Jet Smoothing (jet_smooth_point_set()) vyhladí mračno bodů pomocí promítnutí každého bodu na hladkou parametrickou rovinu, které se nachází mezi k nejbližšími sousedy.

```
// Smoothing.
const unsigned int nb_neighbors = 24
CGAL::jet_smooth_point_set<Concurrency_tag>(
points.begin(),
points.end(),
nb_neighbors);
```

Ukázka kódu 4.6: Ukázka algoritmu Jet Smoothing

4.4.2 Bilaterální vyhlazení

Bilaterální vyhlezení (bilateral_smooth_point_set()) je schopné vyhladit vstupní mračno bodů pomocí iterativního promítání každého bodu do prostoru mezi k nejbližšími sousedy. [63]

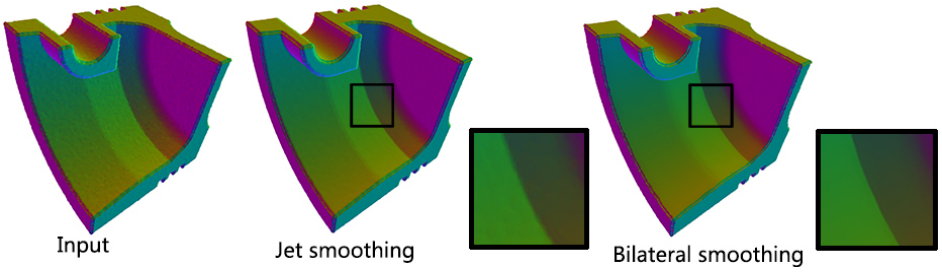
```
// size of neighborhood. The bigger the smoother the result will be.
int k~= 120;
// control sharpness of the result.
double sharpness_angle = 25;
// The bigger the smoother the result will be
// number of times the projection is applied
  int iter_number = 3;

  for (int i = 0; i < iter_number; ++i)
  {
    /* double error = */
    CGAL::bilateral_smooth_point_set <Concurrency_tag>(
          points.begin(),
          points.end(),
          CGAL::First_of_pair_property_map<PointVectorPair>(),
          CGAL::Second_of_pair_property_map<PointVectorPair>(),
          k,
          sharpness_angle);
  }
```

Ukázka kódu 4.7: Ukázka bilaterálního vyhlazení

4.4.3 Srovnání

Výsledek bilaterálního vyhlazení poskytuje ostřejší kontury a zároveň je tento algoritmus rychlejší než algoritmus Jet Smoothing. Jeví se tedy jako lepší. [63]



Obrázek 4.6: Srovnání vyhlazovacích metod. Vlevo je výstup o 250 000 bodech, uprostřed o 197 000 a o 110 000 bodech.[58]

4.5 Výpočet normálových vektorů

Normálové vektory pro každý bod jsou podstatné, pokud se z modelu snažíme sestrojit 3D model, což je náš případ. Jejich výpočet probíhá ve dvou krocích. Nejdříve je nutné odhadnout vektory, protože na začátku žádné nemáme, a následně vypočítat jejich orientaci.

Pro napočítání normálových vektorů používám funkci mst_orient_normals(). Tato metoda používá postup popsaný v [62]. Zjednodušeně – algoritmus sestaví Riemanův graf přes všechny body v mračnu bodů (respektive, přes všechny body v grafu k nejbližších sousedů). Poté propaguje počáteční orientaci napříč minimální kostrou grafu, sestrojenou nad tímto grafem. Výsledkem je orientovaný normálový vektory pro každý vstupní neorientovaný normálový vektor (kromě normálových vektorů, u kterých neproběhla orientace úspěšně).

4.6 Rekonstrukce plochy

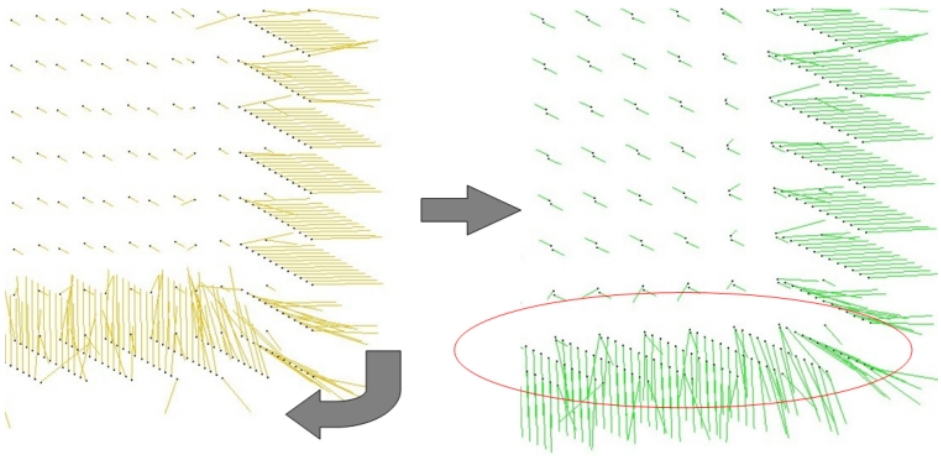
Důležité prerekvizity pro úspěšnou rekonstrukci plochy jsou: Mračno bodů neobsahuje odlehlé hodnoty a současně každý bod musí mít svůj normálový vektor. Ukázka stavu před a po rekonstrukci je vidět na obrázku 4.8.

Vývojový balíček programu MeshLab[57] řeší rekonstrukci povrchu pomocí Poissonova procesu pro aproximovanou indikační funkci odvozené celistvosti, jejíž gradient nejlépe odpovídá vstupním normálovým vektorům. Výstupní skalární funkce je reprezentována oktálovým stromem a je tvarována pomocí adaptivních krychlí.[64]

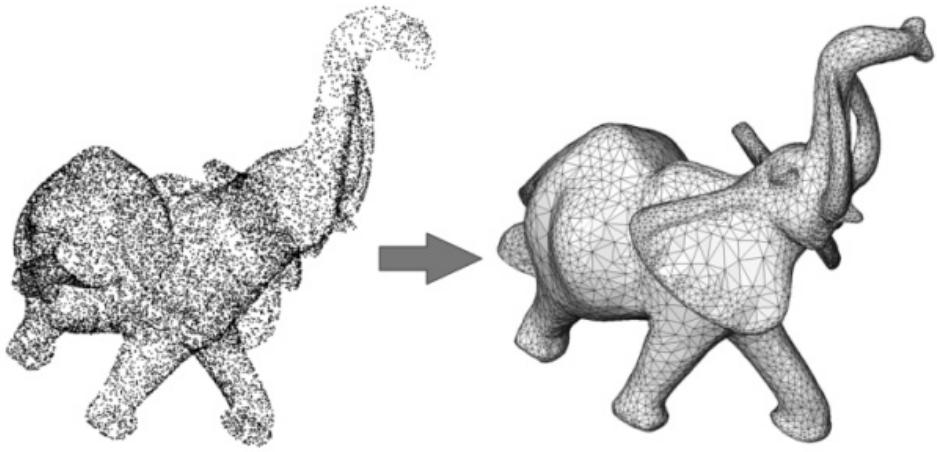
```
// Estimates normals direction.
 const int nb_neighbors = 18;
CGAL::pca_estimate_normals<Concurrency_tag>(points.begin(),
        points.end(),
        CGAL::First_of_pair_property_map<PointVectorPair>(),
        CGAL::Second_of_pair_property_map<PointVectorPair>(),
        nb_neighbors);

// Orients normals.
std::list<PointVectorPair>::iterator unoriented_points_begin =
        CGAL::mst_orient_normals(points.begin(), points.end(),
        CGAL::First_of_pair_property_map<PointVectorPair>(),
        CGAL::Second_of_pair_property_map<PointVectorPair>(),
        nb_neighbors);
```

Ukázka kódu 4.8: Ukázka kódu pro výpočet normálových vektorů



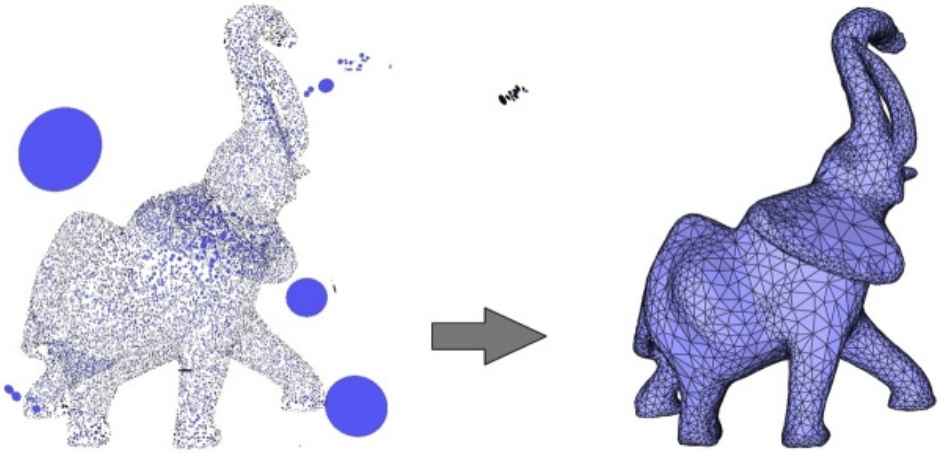
Obrázek 4.7: Orientace normálových vektorů na povrchu krychle. Vlevo neoritentované normálové vektor, vpravo již orientované normálové vektory. Je zde vidět propagace směrem dolů [58]



Obrázek 4.8: Rekonstrukce plochy – vlevo mračno bodu, vpravo model se zrekonstruovanou plochou[58]

4.6.1 Odlehlé body

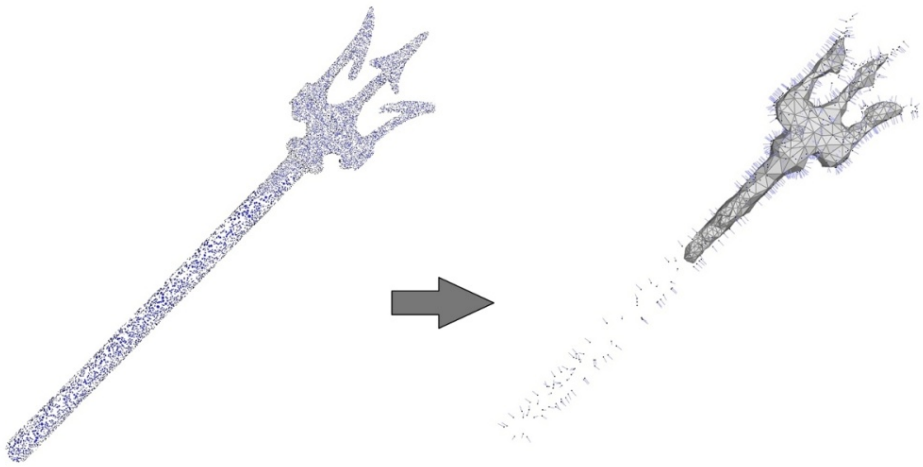
Algoritmus používaný v programu MeshLab umí v případě malého množství ohlehlých bodů tyto body úplně odfiltrovat. Ukázka vstupu s odlehlými body a výstup algoritmu je viditelná v 4.9.



Obrázek 4.9: Rekonstrukce plochy v případě odlehlých bodů[58]

4.6.2 Řídké mračno bodů

Algortimus má známou nevýhodu – vyžaduje relativně husté mračno bodů. V případě řídkého mračna bodů může dojít ke špatné rekonstrukci plochy, jak vidět na obrázku 4.10. Tento problém ale není neřešitelný. Aby proběhla rekonstrukce plochy úspěšně, lze body doplnit dopočítáním.



Obrázek 4.10: Rekonstrukce plochy v případě řídkého mračna bodů[58]

4.7 Shrnutí rekonstrukčního procesu

V této kapitole jsem popsal kompletní proces rekonstrukce. Rekonstrukce byla realizována pomocí vývojového balíčku programu MeshLab. Nyní je potřeba v kapitole 6 Testování otestovat, zda dosáhnu lepších výsledků pomocí vlastní rekonstrukce nebo pokud ponechám rekonstrukci na Intel® RealSense™ SDK.

Kapitola 5

Převod na ševcovské kopyto

V této kapitole se zabývám převodem na ševcovské kopyto, kdy předpokládám, že máme k dispozici naskenovaný 3D model nohy v potřebné kvalitě.

5.1 Situace v průmyslu

V praxi se běžně používájí dva způsoby. Jeden z nich je odebrání naměřených hodnot tradičním způsobem, tedy pomocí obuvnických měřidel a speciálních pomůcek. Ačkoliv se zdá, že je tento způsob nepoužitelný, není tomu tak.

V praxi by to fungovalo následovně. Pomocí kamer ve svém počítači udělám sken své nohy (pravé i levé). Zkontroluji, zda v modelu nejsou viditelné chyby, a model následně pošlu obuvníkovi například na druhý konec světa. Obuvník by na své 3D tiskárně vytiskl moje nohy, ty by změřil[4] podle své potřeby a vyrobil by mi přesně na míru boty, které by mi následně zaslal. Tento přístup by se mohl hodit, pokud by švec vyráběl svá kopyta způsobem, který by byl tajný, nebo kdybych chtěl pár bot, které vyžadují kopyto, na které by se nedal převést sken lidské nohy.

Dalším postupem, který se běžně využívá v průmyslu, je ruční protínání naskenovaného modelu rovinami a manuálnímu zjišťování rozměrů. Výhodou tohoto přístupu je, že si můžete vybrat jakýkoliv rozměr, který vás zajímá, a ten si zjistit. Mezi největší nevýhody tohoto přístupu patří nutnost dalšího manuálního kroku v celém procesu. Naskenovaní není možné automatizovat tak, aby nebylo potřeba člověka, ale převod na kopyto je možné udělat bez manuální práce. Proto jsem se tohoto postupu také nechtěl držet.

5.2 Možná řešení

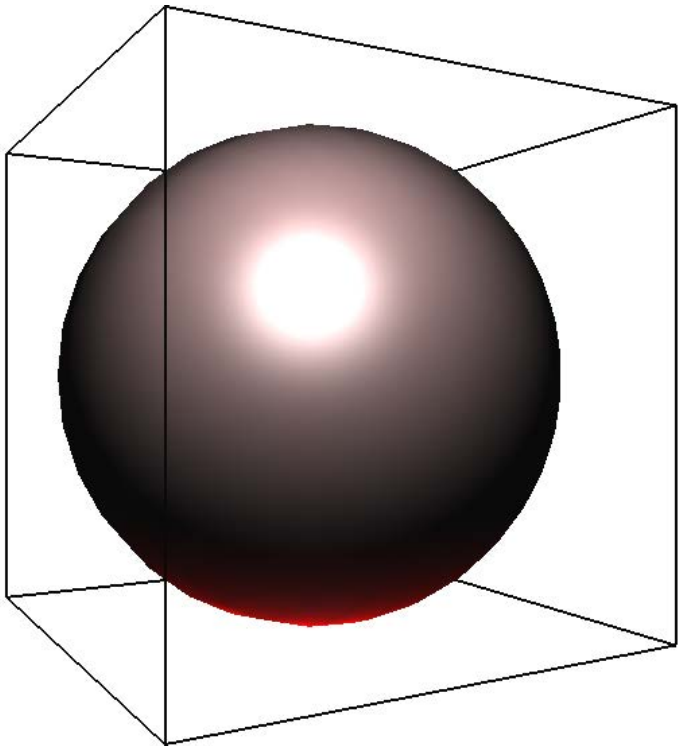
V této části práce představím několik vlastních návrhů řešení problému, ze kterých zvolím to nejvíce vyhovující.

5.2.1 Bounding Box Matching

Předtím, než bude popsán samotný algoritmus, je nezbytné vysvětlit, co je to bounding box.

5.2.1.1 Bounding Box

Bounding box představuje minimální ohraničení okolo modelu, v našem případě okolo 3D modelu. Ve 3D prostoru se jedná o kvádr. Kvádr je sestrojen tak, že se najde globální minimum přes všechny body na osách X, Y, Z, toto minimum označím $X_{min}, Y_{min}, Z_{min}$. Následně je třeba najít i globální maximum přes všechny body na osách X, Y, Z, označím jej $X_{max}, Y_{max}, Z_{max}$. Následně sestrojím dva body, jeden z minim, jeden z maxim. Tyto dva body budou ve výsledném kvádru ty, které bych mohl spojit tělesovou úhlopříčkou.



Obrázek 5.1: Ukázka bounding boxu [65]

5.2.1.2 Postup algoritmu

Postup algoritmu, aby fungoval podle mých potřeb, je následující:

1. mít připravené 3D modely ševcovského kopyta a skenu nohy,

2. oba modely srovnat po všech osách tak, aby byly stejně natočeny,
3. sken nohy seříznout ve stejné výšce jako je ševcovské kopyto,
4. následně srovnat boundig box kopyta tak, aby byl stejný jako bounding box skenu.

5.2.1.3 Posouzení algoritmu

Tento postup jsem zkoušel implementovat, ale nakonec jsem ho zavrh kvůli předpokladu stejného srovnání modelů v prostoru a nutnosti zarovnání modelů na stejnou výšku. Další nevýhodou byly odhlehlé body ve skenu lidské nohy, to znamená body, které nepatři přímo do skenu nohy, ale jsou ve stejném souboru. V případě těchto bodů tento algoritmus naprosto selže. Největší nevýhodou je však úplné vynechání tvaru lidské nohy.

Samozřejmě se dají odhlehlé body lépe či hůře odfiltrovat, ale i tak algoritmus nebyl vhodný kvůli ostatním zmíněným probémům.

5.2.2 Multiple Bounding Box Matching

Další mnou navržený algoritmus, který jsem následně implementoval a testoval, je založený na předchozím algoritmu, avšak lépe reflektuje tvar lidské nohy.

5.2.2.1 Postup

Postup algoritmu je následující:

1. mít připravený 3D model ševcovského kopyta a skenu nohy;
2. oba modely srovnat po všech osách, tak aby byly stejně natočeny;
3. sken nohy seříznout ve stejné výšce jako je ševcovské kopyto;
4. zarovnat střed jejich bounding boxu na souřadnici $[0, 0, 0]$;
5. rozdělit modely na 8 dílů podle os X, Y, Z;
6. rekurzivně opakovat kroky 4 a 5,

   optimální počet kroků rekurze je potřeba vyzkoušet;
7. na konci rekurze srovnat bounding boxy rozřezaných částí modelů;
8. následně při vynořování z rekurze zpětně skládat části kopyta k sobě;
9. vyhlazení celého modelu kopyta, protože obsahuje velké skoky podle počtu zvolených kroků rekurze.

Výhodou oproti předchozímu algoritmu je přiblížení výsledného kopyta tvaru naskenované nohy podle počtu kroků rekurze. Bohužel tento postup má stejné nevýhody jako předchozí algoritmus – nutnost mít modely srovnané v prostoru a možnost rozbití algoritmu odlehlými body. Další nevýhodou je nutnost aplikovat vyhlazovací filtr na celý model, kde nastává riziko rozbití kopyta a jeho znehodnocení pro použití v průmyslu.

5.2.2.2 Posouzení algoritmu

Tento algoritmus představuje lepší variantu než první naivní algoritmus, ale stále není dostatečně vhodný pro daný problém.

5.2.3 Points Pattern Matching

Tento algoritmus není podobný předchozím algoritmům. Hlavní myšlenkou je postavit se k problému jinak a hledat na skenu lidské nohy určité vzory (anglicky patterny)[66] a po jejich zaměření provést mezi těmito body měření. Tímto postupem by mělo být možné docílit až 18 měření [4], které se v případě obuvi na zakázku běžně provádějí.

Následně by bylo možné 3D model ševcovského kopyta zapouzdřit do počítačového modelu s pevně daným rozhraním. Pomocí tohoto rozhraní by bylo možné nastavit jednotlivé délky. Model by samozřejmě musel řešit správné rozpínání a smršťování kopyta v daných rozměrech.

5.2.3.1 Postup

Postup tohoto algoritmu je následující:

1. mít připravený 3D model skenu nohy,
2. mít připravený 3D model ševcovského kopyta, zapouzdřený v počítačovém modelu,
3. na skenu nohy nalézt co nejvíce vzorů,
4. provést měření mezi těmito vzory,
5. zjištěné rozměry přenést do počítačového modelu ševcovského kopyta,
6. modifikovat kopyto podle rozměrů.

Tento algoritmus je velmi dobře rozšiřitelný i o atypická ševcovská kopyta – stačí pouze udělat další počítačový model s pevně daným rozhraním a vše bude fungovat. Zároveň by se dal počítačový model používat i samostatně s manuálním vložením rozměrů.

Pro hledání vzorů na modelech se používají k-dimenzionální stromy (zkráceně kd-stromy)[66].

Kd-stromy
Jde o datovou strukturu pro uložení konečné množiny bodů z k-dimenzionálního prostoru. Umožňuje rychlé vyhledání nejbližšího prvku. Struktura je reprezentována binárním vyhledávacím stromem, kde každý uzel reprezentuje jeden bod z množiny a index souřadnice, podle které se dělí prvky do levého a pravého syna uzlu.

Při vyhledání nejbližšího prvku se najde list náležící cílovému bodu. Nalezený bod je první aproximací nejbližšího bodu. Každý další potenciálně nejbližší prvek musí ležet blíž než první nalezený. Při vyhledávání se algoritmus vrátí k rodiči cílového bodu a pokud je možné, že druhý syn obsahuje bližší bod, tj. vzdálenost hledaného bodu od nadroviny definované synem je menší než vzdálenost od původního cílového bodu, prohledá se druhý podstrom. V opačném případě se druhý podstrom prohledávat nemusí, rovnou se vystoupí o úroveň výše a postup se opakuje.

Při konstrukci stromu se vybere pivotní bod a souřadnice, jež rozdělí množinu prvků, z kterých se konstruuje kd-strom. Prvky se rozdělí do dvou množin – na prvky menší resp. větší než pivot v určené souřadnici. Na obě množiny se aplikuje stejný postup a vytvoří se z nich levý resp. pravý podstrom. Pro vybrání pivotu existuje několik strategií – medián, střed domény dělící dimenze nebo kombinace obou podle rozdělované množiny. Výběr pivotu je důležitý nejen k rovnoměrnému rozdělení bodů do obou podstromů, ale i z hlediska tvaru plochy definované uzlem. Vyhledání prvku vyžaduje minimálně $O(logN)$ a nejhůře $O(N)$, kde N je počet uložených prvků. Přidání prvku do stromu lze provést v čase $O(logN)$. Vyvážení stromu při výběru pivotu kombinovanou strategií mediánu a středu domény vyžaduje čas $O(NlogN)$. Ukázka kd-stromu je vidět na obrázku 5.2. Kód pro jeho sestrojení je vidět v 5.1.

je

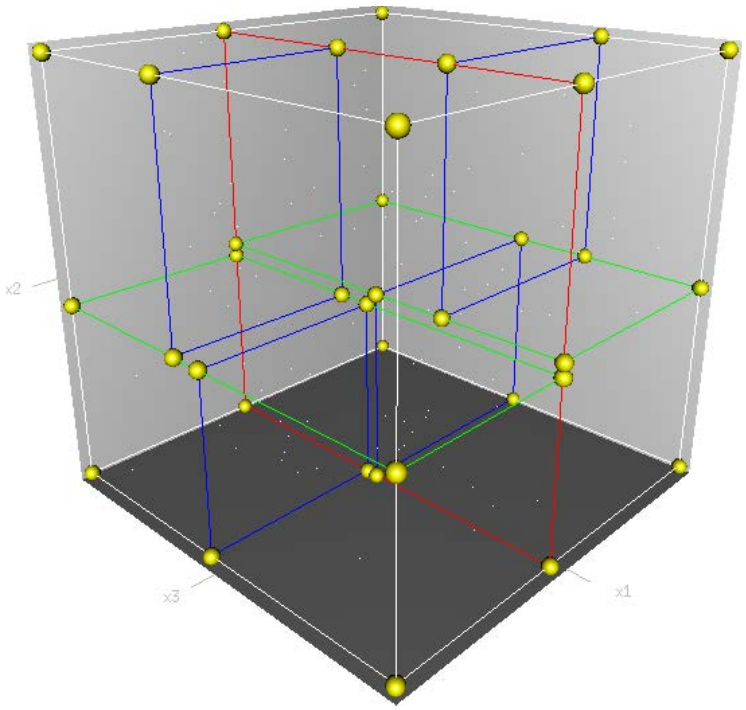
5.2.3.2 Posouzení algoritmu

Algoritmus Points pattern matching se jevil jako nejlepší, dokud jsem nenarazil na problém s přesností, s jakou jsem rozpoznával části lidské nohy. Příkladem úskalí může být rozlišení kotníku z pravé a levé strany nebo rozlišení bodů na lidském nártu, které jsou prakticky identické (kromě své pozice), protože nemají unikátní tvar.

Kvůli těmto problémům neposkytoval algoritmus ani po úpravách uspokojivé výsledky. Pokud se v budoucnu podaří vyřešit, jak přesně rozpoznávat části lidské nohy, bude tento postup funkční a zajímavý.

5.2.4 Iterative Closest Point

Doposud jsem pro práci navrhl mezní algoritmy – příliš naivní nebo přílis složité. Bylo by tedy vhodné navrhnout algoritmus mezi těmito hranicemi. Protože myšlenka z Points pattern matching algoritmu, kdy nebylo potřeba,



Obrázek 5.2: Ukázka kd-stormu[66]

```
function kdtree (list of points pointList, int depth)
{
    // Select axis based on depth so that axis cycles through all valid values
    var int axis := depth mod k;

    // Sort point list and choose median as pivot element
    select median by axis from pointList;

    // Create node and construct subtrees
    var tree_node node;
    node.location := median;
    node.leftChild := kdtree(points in pointList before median, depth+1);
    node.rightChild := kdtree(points in pointList after median, depth+1);
    return node;
}
```

Ukázka kódu 5.1: Ukázka konstrukce kd-stromu

aby modely byly zarovnané v prostoru, může představovat i vstupní podmínku, byla zachována. Při rotování kopytem v prostoru je možné najít takovou pozici, ve které by vzdálenost mezi body kopyta a skenu byla co nejmenší. Takovou pozici lze označit za výchozí a kopyto v této pozici škálovat tak, aby se vzdálenost bodů opět snížila.

5.2.4.1 Postup algoritmu

Postup je tedy následující:

1. mít připravený 3D model skenu nohy a 3D model kopyta,
2. rotovat kopytem v prostoru tak, aby vzdálenost mezi skenerem a kopytem byla co nejmenší,
3. kopyto po všech osách škálovat tak, aby se vzdálenost mezi skenem a kopytem opět snížila,
4. opakování kroků 2 a 3 podle fixního počtu iterací nebo dokud se mezi jednotlivými fázemi nezmenší rozdíl o pevně danou konstantu.

Tento algoritmus splňuje cíle, které jsem si definoval. Navíc není potřeba, aby modely skenu a kopyta byly stejně vysoké nebo aby byly stejně srovnané v prostoru.

Při implementaci jsem objevil open-source program CloudCompare[67], který podobnou metodu má implementovanou ve svém vývojovém balíčku. To mě nasměrovalo na již známý algoritmus který se jmenuje Iterative closest point, který jsem v obecné rovině navrhl výše.

Iterative closest point podrobněji
Jedná se o proces, kdy se snažíme na sebe napárovat dva různé 3D modely nebo mračna bodů. Tento proces se nazývá registrace a funguje následujícím způsobem. Nechť máme dvě nezávisle získané množiny bodů M a D. Množina M reprezentuje známý model velikosti N_m a množina D reprezentuje data, která chceme přidat, jejich velikost označme N_d. Obě množiny reprezentují stejný 3D tvar. Pokud je známá korespondence bodů z dat D k bodům v M, lze jednoduše spočítat transformaci bodů z D do již hotového modelu M. Je tedy nutné určit vzájemně si odpovídající body. Algoritmus ICP předpokládá, že odpovídající body jsou ty nejbližší. Pro ně spočítá transformaci - rotaci R a translaci t, aby se minimalizovala chyba:

$$E(R,t) = \sum_{i=1}^{N_m} \sum_{j=1}^{N_d} w_{i,j} \left\| m_i - (Rd_j + t) \right\|^2$$

kde $w_{i,j}$ představuje váhy pro odpovídající si body. Pokud m_i je nejbližší bod k d_j, je váha $w_{i,j} = 1$, v opačném případě $w_{i,j} = 0$. Nově získaná data se

transformují a použijí k opakovanému určení nejbližších bodů a výpočtu nové transformace. Tento postup se opakuje, dokud není dosaženo minima[68].

5.2.4.2 Posouzení algoritmu

Algoritmus jsem nakonec implementoval s pomocí CloudCompare vývojového balíčku (Anglicky SDK)rozšířeného o možnost škálování. Výsledkem je poměrně robustní řešení, které je schopné škálovat kopyto a párovat je s lidskou nohou i v případě, že je na skenu nohy například noha až po koleno.

5.3 Závěr z možných řešení

Seznámil jsem se se situací řešení problematiky v průmyslu. Následně jsem vysvětlil, proč se chci při řešení ubírat jiným směrem, než je aktuální stav v průmyslu. Poté jsem popsal algoritmy, které jsem navrhl, a rozvedl jsem, proč jsem je nepoužil jako finální. Ke každému algoritmu jsem popsal potřebné datové struktury nebo jiné pojmy.

Jako nejvhodnější jsem vybral algoritmus Iterative closest point, který je rozšířením návrhu, který jsem představil. Algoritmus se jeví jako poměrně robustní díky tomu, že nemá velké předpoklady jako předchozí zmiňované algoritmy.

Kapitola 6

Testování

Během práce jsem vyslovil několik tvrzení, například že mohu zaměnit skutečnou lidskou nohu za vytištěnou nohu na 3D tiskárně, která je potřeba ověřit či dokázat. Zároveň je nutné otestovat vybraný algoritmus pro převod skenu na obuvnické kopyto.

6.1 Skenování skutečné nohy

V kapitole 2.3 jsem popsal důvody a průběh tisku nohy na 3D tiskárně, kterou jsem následně skenoval. Nyní je nutné otestovat, zda práce funguje i pro takové nohy.

Po potřeby testování jsem skenoval svoji nohu (nejdříve levou, pak pravou). Jak jsem zmínil v předchozích kapitolách, používal jsem kameru Intel® RealSense™ SR300. Po dobu skenování jsem měl nohu zafixovanou v pevné pozici a kamerou jsem kroužil okolo nohy.

6.1.1 Ukázka skenů

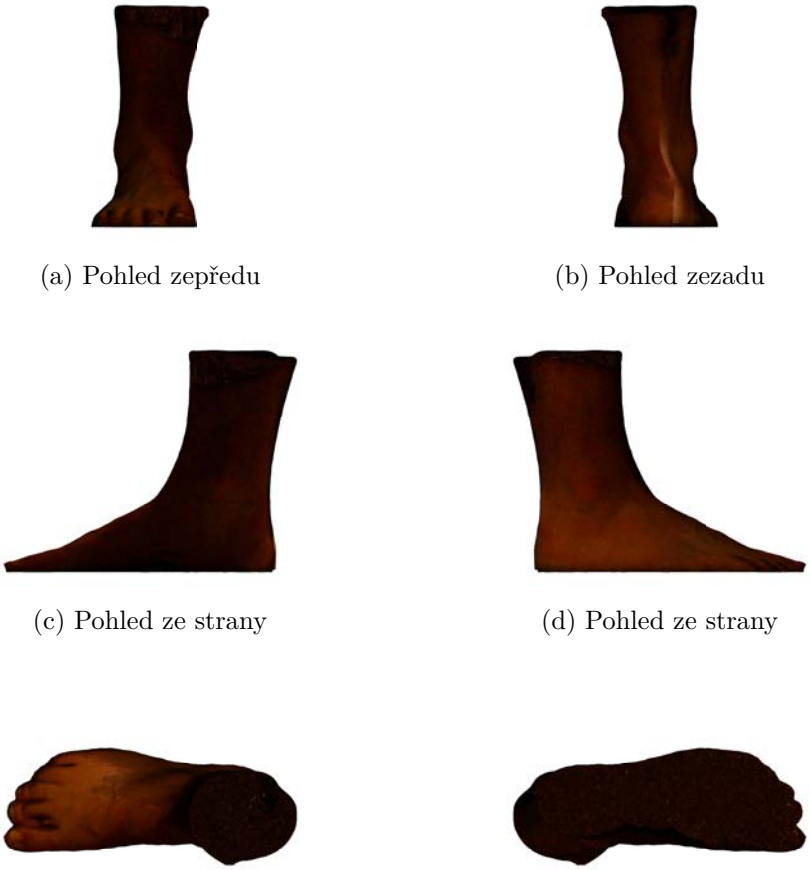
V této sekci je ukázka naskenované lidské nohy. Konkrétně obrázek 6.1 a 6.2.

6.1.2 Závěr ze skenování lidské nohy

Ověřil jsem, že při skenování 3D modelu je možné skenovat i lidskou nohu. Bohužel na renderech vypadá méně kvalitněji než ve skutečnosti. Proto doporučuji prohlédnout si skeny na přiloženém CD.

6.2 Rekontrukce

Dále bylo potřeba otestovat rekonstrukci modelu. Jak jsem zmínil v kapitole 4 je možné buďto z Intel® RealSense™ SDK rovnou vyčíst hotový 3D model nebo pouze mračno bodů. Proto v této sekci provedu s mračnem bodů proces



(a) Pohled zepředu (b) Pohled zezadu

(c) Pohled ze strany (d) Pohled ze strany

(e) Pohled shora (f) Pohled zespodu

Obrázek 6.1: Sken lidské pravé nohy

definovaný v kapitole 4 a na konci srovnám, zda jsem dosáhl výsledků stejných, lepších nebo horších než Intel® RealSense™ SDK.

6.2.1 Rekonstrukční proces

Provedl jsem všechny kroky rekonstrukčního procesu tak, jak jsem si je nadefinoval:

1. analýza,
2. zjednodušení,
3. odstranění odlehlých bodů,
4. vyhlazení,

(a) Pohled zepředu

(b) Pohled zezade

(c) Pohled ze strany

(d) Pohled ze strany

(e) Pohled shora

(f) Pohled zespodu

Obrázek 6.2: Sken lidské levé nohy

5. počítaná normálových vektorů,
6. rekonstrukce plochy.

V následujících obrázcích jsem některé kroky záměrne vynechal, protože neměly viditelný vliv na proces. Jedná se o následující kroky:

- analýza,
- odstranění odlehlých bodů,

  mračno bodů získaná prostřednictvím Intel® RealSense™ SDK neobsahuje žádné ohledhlé body.
- vyhlazení.

(a) Výchozí stav mračna bodů

(b) Mračno bodů po zjednodušení

(c) Napočítání normálových vektorů

(d) Zrekonstruovaná plocha

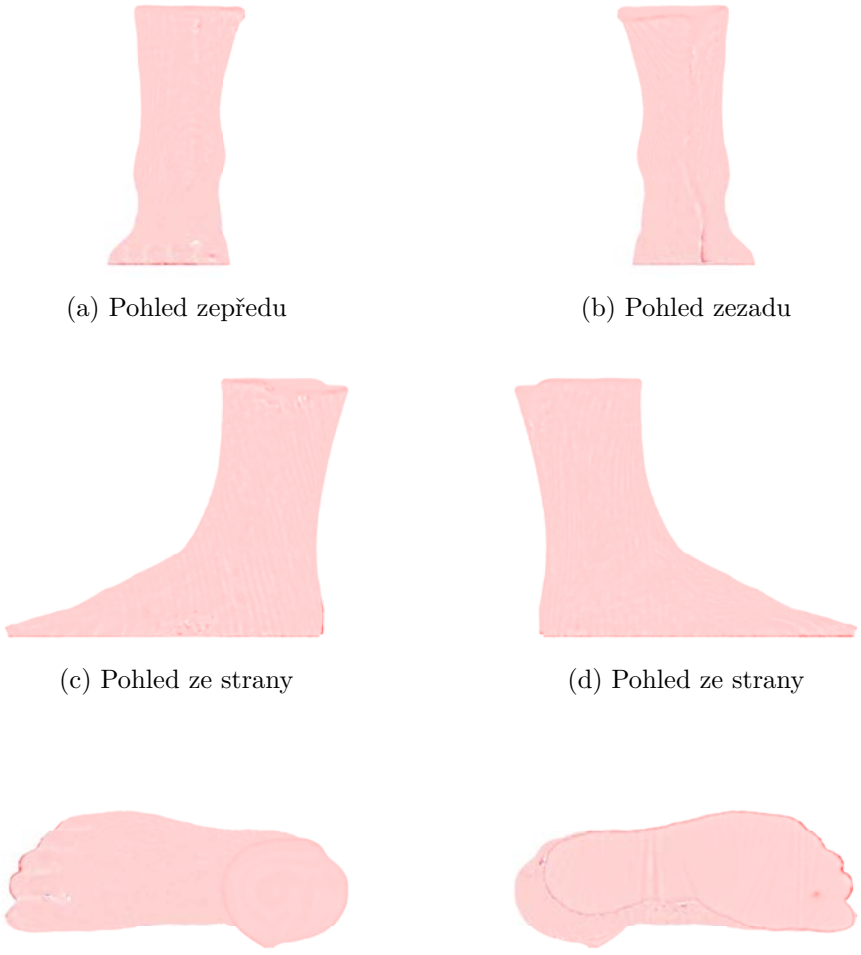
Obrázek 6.3: Ukázka procesu rekonstrukce

6.2.2 Měření vzdáleností

Model získaný rekonstrukcí z mračna bodů jsem následně zkusil srovnat s modelem, který mám přímo z Intel® RealSense™ SDK. Rovnou jsem vykreslil i histogram se vzálenostmi 6.5.

6.2.3 Shrnutí rekonstrukce

Jak je možné vidět na histogramu 6.5, dopadla rekonstrukce velmi podobně jako model získaný přimo z Intel® RealSense™ SDK. Je to především díky kvalitnímu mračnu bodů, které bylo na vstupu rekonstrukčního procesu. Nyní vím, že mužu používat buďto rekonstrukci vlastní, nebo hotový model z Intel® RealSense™ SDK a výsledek bude stejně dobrý.



(a) Pohled zepředu

(b) Pohled zezadu

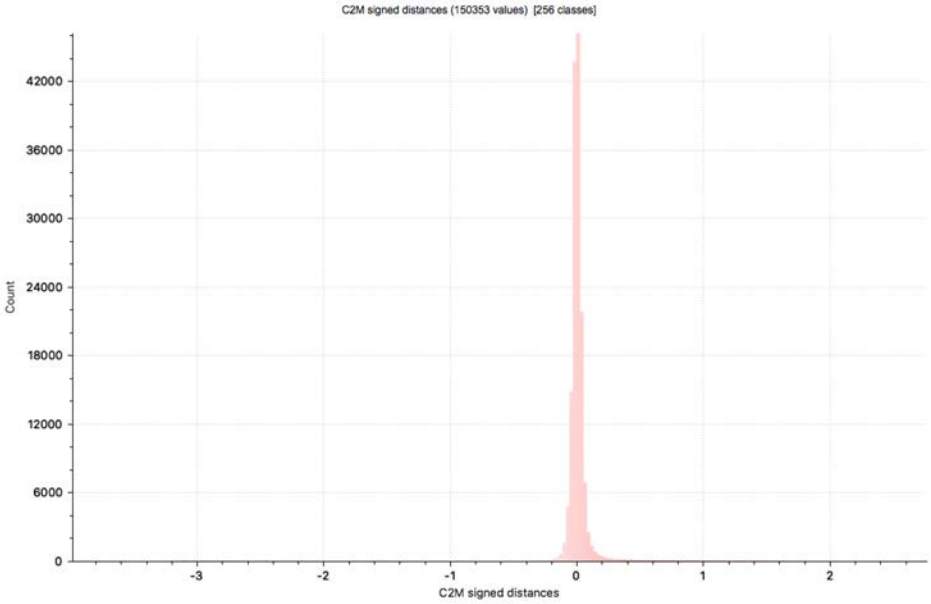
(c) Pohled ze strany

(d) Pohled ze strany

(e) Poohled ze shora

(f) Pohled zespodu

Obrázek 6.4: Srovnání modelu získaného vlastní rekonstrukcí s modelem získaným s Intel® RealSense™ SDK



Obrázek 6.5: Ukázka histogramu se vzdálenostmi

6.3 Testování algoritmu

Algoritmus k otestování je mnou vybraný Iterative closest point.

U samotného algoritmu tvrdím, že sken nohy a kopyto může být různě natočené a v různé výšce v prostoru. Algoritmus dokáže kopyto a 3D sken na sebe správně zaregistrovat. Zároveň tvrdím, že dojde k naškálování obuvnického kopyta tak, aby rozdíl s lidskou nohou byl co nejmenší. Proto bude tako sekce rozdělena na několik dílčích částí.

6.3.1 Testování translace a rotace

Tento bod budu testovat následovně: Do prostoru bude umístěn sken lidské nohy a obuvnické kopyto. Spustím algoritmus a po jeho skončení budu očekávat kopyto registrované na lidskou nohu. Pro testování jsem zvolil pravou lidskou nohu, protože je záměrně naskenovaná s větší částí lýtka.

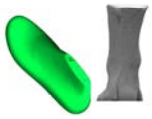
6.3.1.1 Ukázka výchozího a konečného stavu

Na přiložených obrázcích 6.6 a 6.7 je možné vidět výchozí a konečný stav pro tento testovací scénář.

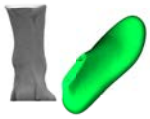
6.3.1.2 Závěr testu translace a rotace

Jako první je potřeba zmínit, že v tomto testu neprobíhalo škálovaní, proto není potřeba porovnávat, jak moc sedí obuvnické kopyto na noze. Lze konstatovat, že ačkoliv obuvnické kopyto začínalo v 3D prostoru na jiných souřadnicích, registrace dopadla velmi zdařile.

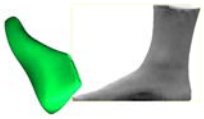
Následně jsem zkoušel různé vzájemné pozice a všechny dopadly stejně jako tento test.



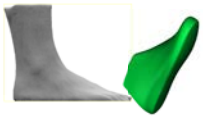
(a) Pohled zepředu



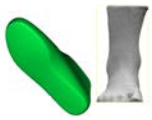
(b) Pohled zezadu



(c) Pohled ze strany



(d) Pohled ze strany



(e) Pohled shora

(f) Pohled zespodu

Obrázek 6.6: Ukázka výchozí konfigurace pro test translace a rotace

(a) Pohled zepředu

(b) Pohled zezadu

(c) Pohled ze strany

(d) Pohled ze strany

(e) Pohled shora

(f) Pohled zespodu

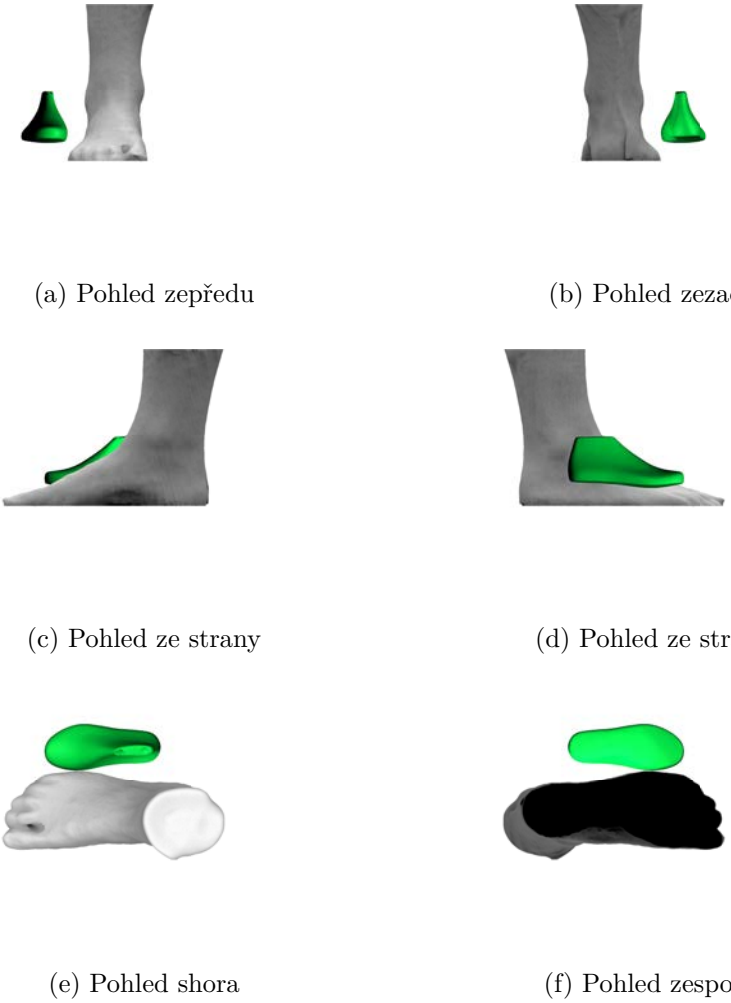
Obrázek 6.7: Ukázka konce testu translace a rotace

6.3.2 Testování škálování

Po ověření registrace obuvnického kopyta ke skenu lidské nohy je nutné ověřit i správné škálování obuvnického kopyta, aby rozdíl vzdáleností byl co nejmenší. V tomto testování se spolu se škálováním provádí i rotace a translace, aby byla zajištěna správná funkčnost.

6.3.2.1 Ukázka výchozího a konečného stavu

Na přiložených obrazcích 6.8 6.9 je možné vidět výchozí a konečný stav pro tento testovací scénář.



(a) Pohled zepředu

(b) Pohled zezadu

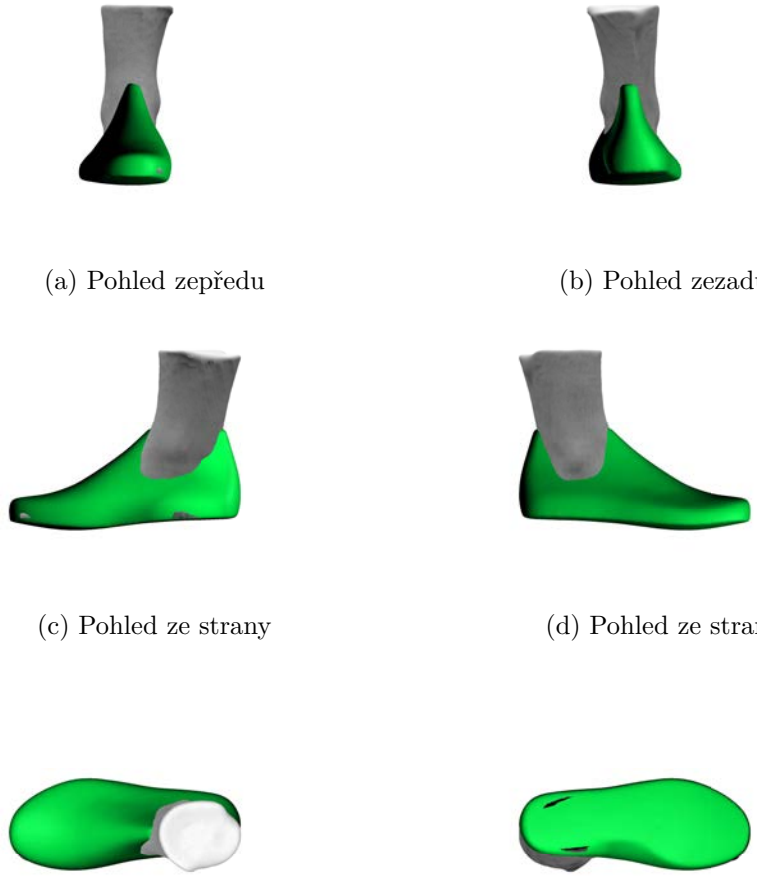
(c) Pohled ze strany

(d) Pohled ze strany

(e) Pohled shora

(f) Pohled zespodu

Obrázek 6.8: Stav před testem škálování



(a) Pohled zepředu

(b) Pohled zezadu

(c) Pohled ze strany

(d) Pohled ze strany

(e) Pohled shora

(f) Pohled zespodu

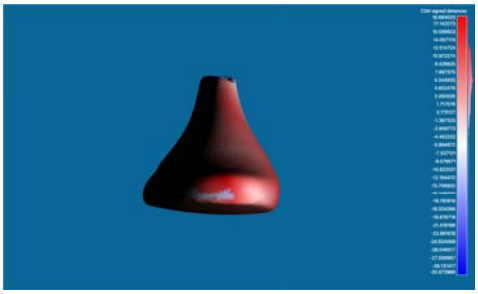
Obrázek 6.9: Stav na konci testu škálování

6.3.2.2 Měření vzdáleností

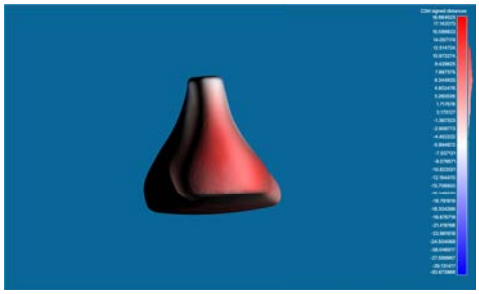
Jak přesně obuvnické kopyto sedí, je nejlépe vidět z obrázků 6.10. Při pohledu shora jsou velmi dobře viditelné obrysy prstů. Horní část kopyta obsahuje sytě modrou barvu, protože zde je kopyto mnohem užší než lidská noha.

Další srovnání modelů je možné pomocí histogramu vzdáleností 6.11, ve kterém jsou červeně znázorněny partie, kde je kopyto větší než sken lidské nohy, a modře naopak partie, kde je kopyto menší. Histogram obsahuje na horizontální ose možné vzdálenosti a na ose vertikální počet výskytů dané vzdálenosti. Každý sloupec představuje rozmezí vzdálenosti. Graf, který by obsahoval obě zmíněné hodnoty, by byl velmi nepřehledný, a tak jsou tato data pro graf ve formátu CSV na přiloženém CD.

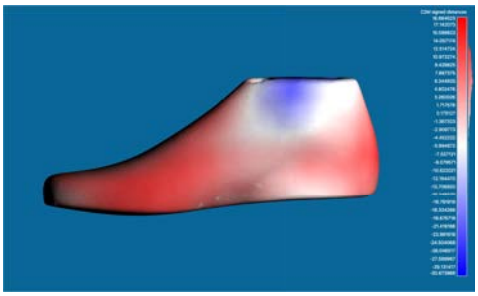
Na histogramu 6.11 lze pozorovat, že většina bodů na obuvnickém kopytu je větší než lidská noha, což je správně. Kdyby tomu bylo naopak, boty by byly ušité podle kopyta, které je menší než lidská noha, a obuv by tak byla zákazníkovi malá.



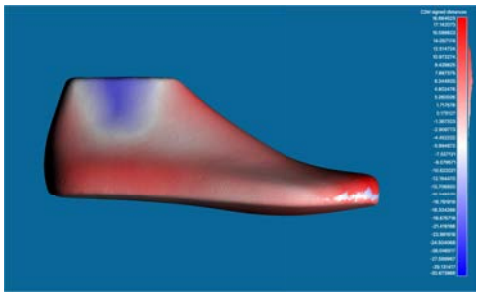
(a) Pohled zepředu



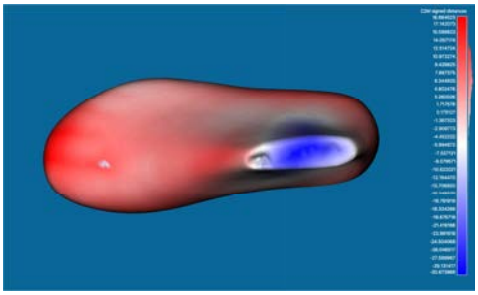
(b) Pohled zezadu



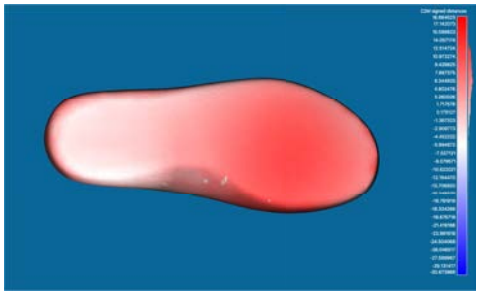
(c) Pohled ze strany



(d) Pohled ze strany



(e) Pohled shora

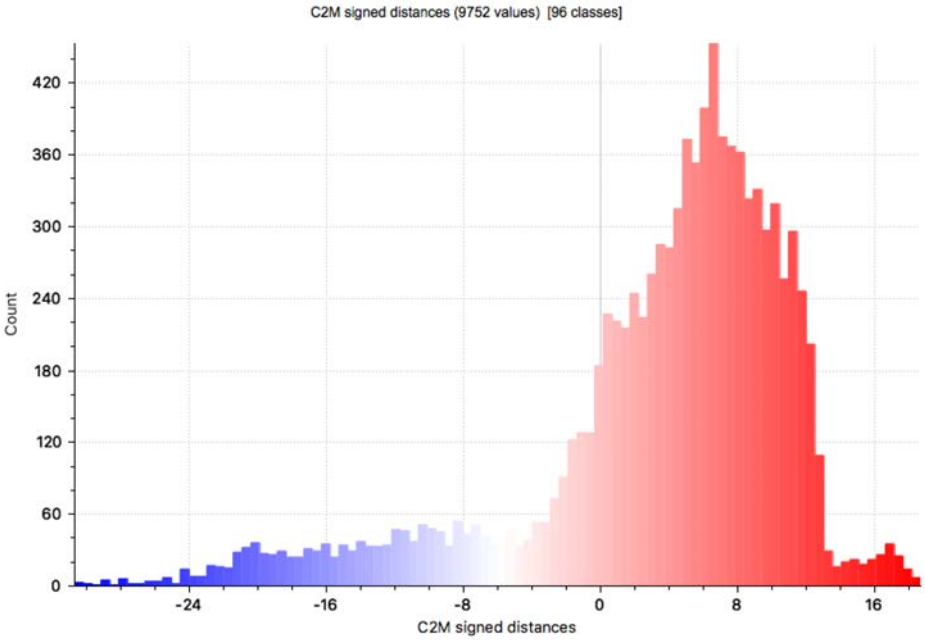


(f) Pohled zespodu

Obrázek 6.10: Ukázka srovnání obuvnického kopyta a 3D skenu

6.3.2.3 Závěr testu škálování

Ukázalo se, že alogoritmus správně škáluje obuvnické kopyto. Zároveň se mi podařilo přehledně graficky zobrazit výsledky testování pro prezentaci.



Obrázek 6.11: Ukázka histogramu se vzdálenostmi

Iterace	Čas pro 1 vlákno	Čas pro 2 vlákna	Čas pro 4 vlákna	Čas pro 8 vláken
10	53 s	55 s	49 s	45 s
20	72 s	62 s	53 s	50 s
40	129 s	78 s	65 s	54 s
80	239 s	169 s	143 s	130 s

Tabulka 6.1: Časy měření

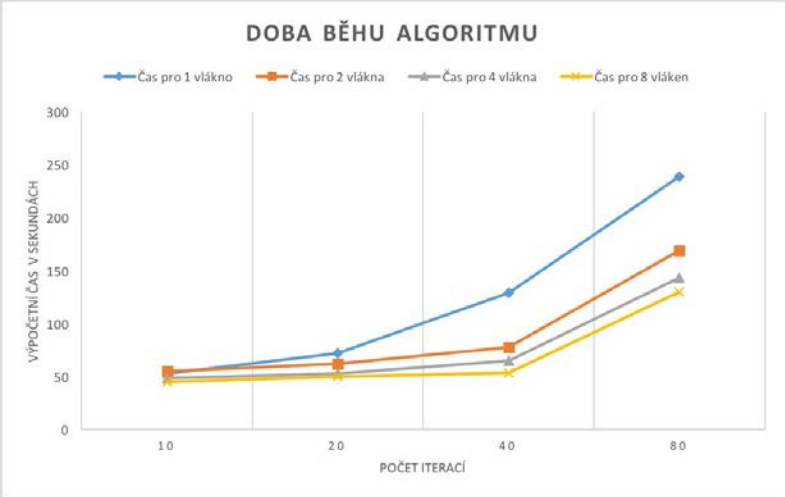
6.3.3 Měření délky běhu algoritmu

Dále je nezbytné otestovat, jak dlouho algoritmus běží a zda má vliv běh na více vláknech. Testování podrobím kompletní algoritmus, tedy rotaci, translaci a škálování. Veškerá měření jsou zaznamenána ve vteřinách. Měření je vidět v tabulce 6.1 nebo grafu6.12. Měření probíhalo na mém počítači, jeho konfigurace je následující:

Počítač MacBook Pro (Retina, Mid 2012)

Procesor 2.3 GHz Intel® Core i7

Operační pamět 8 GB 1600 MHz DDR3



Obrázek 6.12: Měření délky běhu algoritmu

6.3.3.1 Závěr měření délky běhu algoritmu

Ukázalo se, že algoritmus není optimálně paralelní, což pro tuto práci není problém. Cílem práce není implementace plně paralelního algoritmu, a tak je zde možnost navázání a vylepšení.

6.3.4 Závěr testování algoritmu

Algoritmus funguje přesně tak, jak byl popsán v kapitole . Při testování nebylo zjištěno žádné anomální chování a algoritmus dosahuje uspokojivých výsledků.

6.4 Shrnutí testování

Veškeré předpoklady, které jsem v práci zmínil, jsem v této kapitole otestoval. Testováním bylo ověřeno, že skenování funguje stejně dobře na lidské noze i na vytištěné předloze. Rekonstrukce se ukázalo, že je jedno pokud používám svojí, nebo ji nechávam na Intel® RealSense™ SDK. Zároveň algoritmus funguje tak, jak jsem očekával, a poskytuje uspokojivé výsledky. Během měření běhu algoritmu bylo zjištěno, že je možné algoritmus dále vylepšovat v oblasti jeho paralelizace.

Závěr

Provedl jsem rešerši několika různých hloubkových kamer. Srovnával jsem více pozicovaných kamer oproti jedné. Zde jsem narazil na problém vzájemného rušení kamer. Ukázalo se ale, že pro práci to není problém, protože se mi podařilo získávat kvalitní skeny pomocí kamery jedné.

Ukázalo, se že nejlepší kamera z mého srovnání je Intel® RealSense™ SR300. Jelikož je technologie zatím ve stádiu vývoje, měl jsem kameru určenou pro vývojáře. To znamená, že se v současném stavu nedá použít pro jakékoliv řešení v praxi z důvodů licencí. Ale je to technologie, která do pár let bude ve většině nových počítačů, a řešení popsané v této práci bude velmi dobře použitelné.

Dále jsem nastudoval různé metody pro rekonstrukci 3D modelu a převodu 3D modelu na obuvnické kopyto. Vybrané metody jsem implementoval a testoval. Výstupem této práce není jediný ucelený program, ale několik dílčích částí, které je v budoucnu možné spojit do uceleného programu s grafickým uživatelským rozhraním.

Testování ukázalo, že vybrané technologie a algoritmy splňují požadavky, které jsme spolu s vedoucím práce stanovili v zadání.

V průběhu tvorby práce jsem zjistil, že skenování, rekonstrukce modelu a převod modelu na obuvnické kopyto je velmi rozsáhlé téma. Jednotlivé části by (alespoň dle mého názoru) snadno stačily na témata samostatných diplomových prací.

Technologie, se kterou jsem se seznámil, má všestranné využití. Kromě případů, které jsem zmapoval v této práci, se jedná například o integraci Intel® RealSense™ kamer v dronech. To přináší další zlomový okamžik v autonomním řízení dronů a autopilotování.[69]

Přínos práce vidím v tom, že jsem nesrovnával jen aktuálně veřejně dostupné technologie, ale také ty, které budou aktuální v blízké budoucnosti. Zároveň jsou vzniklé postupy použitelné na libovolném počítači, který bude mít zabudovanou kameru Intel® RealSense™. To je aspekt, který výrazně zvyšuje možné využití v praxi.

Navázání na tuto práci

Na tuto práci se dá navázat v mnoha ohledech. V této části rozepíšu jednotlivé možnosti.

Automatizovaná příprava tisku obuvnického kopyta

Je možné vytvořit nástroj, který by automaticky připravil obuvnické kopyto pro tisk na 3D tiskárně. Tento nástroj by mohl mít několik parametrů. Například jaká technologie 3D tisku bude zvolena, a podle toho vybrat, zda je model potřeba rozdělit na více části či nikoliv. V případě rozdělování modelu na více částí by byl zajímavý parametr velikost tiskové plochy a maximální možný úhel tisku. Program by v závislosti na těchto parametrech připravil tiskové pláty.

Implementace a testování Intel® RealSense™ zabudovaných hloubkových kamer

Další možné rozšíření by bylo, počkat až se kamery SR300 (nebo navazující) začnou objevovat v běžných počítačích a vyzkoušet 3D skenování nohou přímo s nimi.

Nepochybuji, že mezitím vývoj kamer urazí ještě velký krok, takže výsledná kvalita bude ještě lepši než ta, které jsme dosáhl nyní. Zároveň by se mohly vyskytnout nové problémy, které jsem teď nemusel s externí kamerou řešit.

Vylepšení algoritmu pro převod skenu na obuvnické kopyto

Existuje řada možností vylepšení samotného algoritmu pro převod 3D skenu na obuvnické kopyto. Například využít neuronové sítě, nebo vyzkoušet něco zcela nového.

Grafický návrh a vznik uceleného programu

Zadáním této práce nebyl vznik grafických návrhů pro ucelenou aplikaci. Je tedy možné navrhnout grafické návrhy, udělat uživatelské testování a spojit více dílčích částí do ucelené aplikace pro laické uživatele.

Budoucí vývoj

V současné chvíli se plánuji zabývat dále kd-stromy a zkusit upravit algoritmus pro převod 3D skenu na obuvnické kopyto tak, aby je používal efektivně.

Literatura

[1] Intel Corporation: Using 3D Imaging to Improve Shoe Sizing. [online], 2016, [Citováno 2016-12-14]. Dostupné z: https://software.intel.com/en-us/articles/when-the-shoe-fits

[2] Hinijo-Pérez J. J., Davia-Aracil M., Jimeno-Morenilla A., Salas F.: Shoe-last design exploration and customization. [online], 2011, [Citováno 2016-12-15]. Dostupné z: https://www.researchgate.net/publication/254251102_Shoe-last_design_exploration_and_customization

[3] Contributors of Wikipedia: Shoe LAST. [online], 2016, [Citováno 2016-12-20]. Dostupné z: https://en.wikipedia.org/wiki/Last#/media/File:PikiWiki_Israel_13497_shoe_trees.JPG

[4] Witana P. Ch., Xiong S., Zhao J., Goonetilleke R.S.: Foot measurements from three-dimensional scans: A comparison and evaluation of different methods. [online], 2006, [Citováno 2016-12-15]. Dostupné z: https://www.researchgate.net/publication/222693176_Foot_measurements_from_three-dimensional_scans_A_comparison_and_evaluation_of_different_methods

[5] Vorum Research Corp.: Vorum - YETI 3D Scanner. [online], 2016, [Citováno 2016-12-10]. Dostupné z: http://vorum.com/footwear/yeti-3d-foot-scanner/

[6] Microsoft: Kinect for Xbox One. [online], 2016, [Citováno 2016-12-10]. Dostupné z: http://www.xbox.com/en-US/xbox-one/accessories/kinect

[7] Contributors of Wikipedia: Kinect. [online], 2016, [Citováno 2016-12-10]. Dostupné z: https://en.wikipedia.org/wiki/Kinect

[8] FabScan: FabScan open source 3D scanner. [online], 2011, [Citováno 2016-12-15]. Dostupné z: http://www.ponoko.com/blog/2011/12/19/fabscan-open-source-3d-scanner/

[9] Method and system for object reconstruction. [online], [Citováno 2016-12-28]. Dostupné z: https://patentscope.wipo.int/search/en/detail.jsf?docId=WO2007043036&recNum=1&maxRec=&office=&prevFilter=&sortOption=&queryString=&tab=PCTDescription

[10] A., M.: How a Depth Sensor Works - in 5 Minutes. [online], 2013, [Citováno 2016-12-28]. Dostupné z: https://jahya.net/blog/how-depth-sensor-works-in-5-minutes/

[11] Andrew McWilliams: How a Depth Sensor Works. [online], 2013, [Citováno 2016-12-10]. Dostupné z: https://jahya.net/blog/how-depth-sensor-works-in-5-minutes/

[12] Ma X., Luximon A.: 3D foot prediction method for low cost scanning. [online], 2014, [Citováno 2016-12-15]. Dostupné z: https://www.researchgate.net/publication/266799310_3D_foot_prediction_method_for_low_cost_scanning

[13] Shi N., Yi S., Xiong S., Jiang Z.: A CAD System for Shoe Last Customization. [online], 2009, [Citováno 2016-12-15]. Dostupné z: https://www.researchgate.net/publication/266799310_3D_foot_prediction_method_for_low_cost_scanning

[14] Rout N., Zhang Y., Khandual A., Luximon A.: 3D foot scan to custom shoe last. [online], 2010, [Citováno 2016-12-15]. Dostupné z: https://www.researchgate.net/publication/228879419_3D_foot_scan_to_custom_shoe_last

[15] Mundo Reader, S.L.: BQ - 3D World. [online], 2016, [Citováno 2016-12-10]. Dostupné z: https://www.bq.com/en/3d-world

[16] Contributors of RepRap wiki: Ciclop. [online], 2016, [Citováno 2016-12-10]. Dostupné z: http://reprap.org/wiki/Ciclop

[17] Logitech: HD Webcam C270. [online], 2016, [Citováno 2016-12-10]. Dostupné z: http://www.logitech.com/cs-cz/product/hd-webcam-c270

[18] ASUSTeK Computer Inc.: ASUS Xtion PRO LIVE. [online], 2013, [Citováno 2016-12-14]. Dostupné z: https://www.asus.com/3D-Sensor/Xtion_PRO_LIVE/

[19] Le N.: A Comparison of Intel® RealSense™ Front-Facing Camera SR300 and F200. [online], 2016, [Citováno 2016-12-14]. Dostupné z: https://software.intel.com/en-us/articles/a-comparison-of-intel-realsensetm-front-facing-camera-sr300-and-f200

[20] Intel Corporation: Technology That Really Matters). [online], 2016, [Citováno 2016-12-14]. Dostupné z: https://software.Intel.com/en-us/forums/realsense/topic/537872#comment-1810928

[21] Intel Corporation: Intel® RealSense™ Technology. [online], 2016, [Citováno 2016-12-14]. Dostupné z: http://www.intel.com/buy/us/en/audience/realsense/

[22] Intel Corporation: Long-Range Intel® RealSense™ Camera. [online], 2016, [Citováno 2016-12-14]. Dostupné z: http://www.intel.com/content/www/us/en/architecture-and-technology/realsense-longrange.html

[23] CatchEye: CatchEye 3D: Supported cameras. [online], 2016, [Citováno 2016-12-14]. Dostupné z: https://catcheye.freshdesk.com/support/solutions/articles/14000035864-catcheye-3d-supported-cameras

[24] ArcSoft, Inc.: ArcSoft® DepthCam™. [online], 2016, [Citováno 2016-12-14]. Dostupné z: http://www.arcsoft.com/depthcam/

[25] Contributors of Developer Forum: Utility for changing laser camera parameters (IVCAM v0.5). [online], 2015, [Citováno 2016-12-14]. Dostupné z: https://software.Intel.com/en-us/forums/realsense/topic/537872#comment-1810928

[26] Samontab: How F200 and SR300 work. [online], 2016, [Citováno 2016-12-14]. Dostupné z: https://software.intel.com/en-us/forums/realsense/topic/537872#comment-1810928

[27] Contributors of Wikipedia: Point cloud. [online], 2016, [Citováno 2016-12-20]. Dostupné z: https://en.wikipedia.org/wiki/Point_cloud

[28] Berger M., Tagliasacchi A., Seversky L. M., Alliez P., Guennebaud G., Levine J. A., Sharf Andrei, Silva C. T.: A Survey of Surface Reconstruction from Point Clouds. [online], 2016, [Citováno 2016-12-20]. Dostupné z: http://gfx.uvic.ca/pubs/2016/reconstar/paper.pdf

[29] Contributors of research project: Building Rome in a Day. [online], 2016, [Citováno 2016-12-20]. Dostupné z: http://grail.cs.washington.edu/rome/col-sfm0.JPG

[30] Contributors of Wikipedia: Model (abstrakce). [online], 2016, [Citováno 2016-12-20]. Dostupné z: https://cs.wikipedia.org/wiki/Model_(abstrakce)

[31] Shene Ch.-K.: NURBS: Definition. [online], 2016, [Citováno 2016-12-20]. Dostupné z: https://www.cs.mtu.edu/~shene/COURSES/cs3621/NOTES/spline/NURBS/NURBS-def.html

[32] Contributors of Wikipedia: Non-uniform rational B-spline. [online], 2016, [Citováno 2016-12-20]. Dostupné z: https://en.wikipedia.org/wiki/Non-uniform_rational_B-spline

[33] Alias Systems: Polygonal Modeling. [online], 2004, [Citováno 2016-12-20]. Dostupné z: https://courses.cs.washington.edu/courses/cse459/06wi/help/mayaguide/Complete/Polygons.pdf

[34] Contributors of OpenSCAD: OpenSCAD User Manual/Using the 2D Subsystem. [online], 2016, [Citováno 2016-12-20]. Dostupné z: https://en.wikibooks.org/wiki/OpenSCAD_User_Manual/Using_the_2D_Subsystem

[35] Hrončok M., Žehra M.: Práce s 3D modely ve formě meshí. [online], 2016, [Citováno 2016-12-20]. Dostupné z: https://edux.fit.cvut.cz/courses/BI-3DT/tutorials/mesh/start

[36] 3D Systems, Inc.: What Is An STL File? [online], 2016, [Citováno 2016-12-20]. Dostupné z: http://www.3dsystems.com/quickparts/learning-center/what-is-stl-file

[37] Contributors of Wikipedia: STL File problem. [online], 2016, [Citováno 2016-12-10]. Dostupné z: https://upload.wikimedia.org/wikipedia/commons/thumb/a/a6/The_differences_between_CAD_and_STL_Models.svg/300px-The_differences_between_CAD_and_STL_Models.svg.png

[38] Murray J. D., Van Ryper W.: *Encyclopedia of Graphics File Formats*. O'Reilly, 1994, ISBN 1-56592-161-5.

[39] Contributors of Wikipedia: Right-hand rule. [online], 2016, [Citováno 2016-12-10]. Dostupné z: https://en.wikipedia.org/wiki/Right-hand_rule

[40] MakerBot® Industries, LLC: Thingiverse - Digital Designs for Physical Objects. [online], 2016, [Citováno 2016-12-10]. Dostupné z: http://www.thingiverse.com

[41] Voodoo Manufacturing, Inc.: Life-Size Body Model. [online], 2016, [Citováno 2016-12-10]. Dostupné z: https://www.thingiverse.com/thing:1615359

[42] Josef Průša: 3D printing handbook. [online], 2016, [Citováno 2016-12-10]. Dostupné z: http://www.prusa3d.com/downloads/manual/prusa3d_manual_175_en.pdf

[43] Filementuim: Lemonesol. [online], 2016, [Citováno 2016-12-14]. Dostupné z: http://fillamentum.com/products/fillamentum-lemonesol

[44] Aleph Objects, Inc.: Lulzbot cura edition. [online], 2016, [Citováno 2016-12-14]. Dostupné z: https://www.lulzbot.com/cura

[45] Aleph Objects, Inc.: Lulzbot TAZ 6. [online], 2016, [Citováno 2016-12-14]. Dostupné z: https://www.lulzbot.com/store/printers/lulzbot-taz-6

[46] Occipital, Inc.: OpenNI. [online], 2016, [Citováno 2016-12-10]. Dostupné z: http://structure.io/openni

[47] Contributors of GitHub: OpenNI2. [online], 2016, [Citováno 2016-12-20]. Dostupné z: https://github.com/OpenNI/OpenNI2

[48] Occipital, Inc.: Structure Sensor – Capture the World in 3D. [online], 2016, [Citováno 2016-12-10]. Dostupné z: https://store.structure.io/store?_ga=1.230077594.769287766.1483130429

[49] Occipital, Inc.: Structure Sensor – Developer. [online], 2016, [Citováno 2016-12-10]. Dostupné z: http://structure.io/developers

[50] Reconstruct Me: ReconstructMe - Digitize Your World. [online], 2016, [Citováno 2016-12-10]. Dostupné z: http://reconstructme.net

[51] Intel Corporation: Intel® RealSense Community. [online], 2016, [Citováno 2016-12-14]. Dostupné z: https://communities.intel.com/community/tech/realsense

[52] Intel Corporation: Cross-platform camera capture for Intel® RealSense™ F200, SR300 and R200. [online], 2016, [Citováno 2016-12-20]. Dostupné z: https://github.com/IntelRealSense/librealsense

[53] Net Applications: Desktop Operating System Market Share. [online], 2016, [Citováno 2016-12-10]. Dostupné z: https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=10&qpcustomd=0

[54] Microsoft: Skeleton Tracking With Multiple Kinect Sensors. [online], 2016, [Citováno 2016-12-10]. Dostupné z: https://msdn.microsoft.com/en-us/library/dn188677.aspx

[55] Microsoft: Media feature pack for Windows 10 N and Windows 10 KN editions. [online], 2016, [Citováno 2016-12-10]. Dostupné z: https://support.microsoft.com/cs-cz/kb/3010081

[56] Zappos: Shoe Size Conversion. [online], 2014, [Citováno 2016-12-10]. Dostupné z: http://www.zappos.com/c/shoe-size-conversion

[57] Contributors of GitHub: MeshLab. [online], 2016, [Citováno 2016-12-20]. Dostupné z: https://github.com/cnr-isti-vclab/meshlab

[58] Pierre Alliez, Clément Jamin, Quentin Mérigot, Jocelyn Meyron, Laurent Saboret, Nader Salman, Shihao Wu: CGAL 4.9 - Point Set Processing. [online], 2016, [Citováno 2016-12-20]. Dostupné z: http://doc.cgal.org/latest/Point_set_processing_3/index.html#Point_set_processing_3NormalOrientation

[59] Bishop Ch. M.: *Pattern Recognition and Machine Learning*. Springer Science + Business Media, 2006, ISBN 978-0387-31073-2.

[60] Hastie T., Tibshirani R., Friedman J.: *The Elements of Statistical Learning*. Springer, druhé vydání, 2011, ISBN 80-200-1062-9.

[61] Huang H., Li D., Zhang H., Ascher U., Cohen-Or D.: Consolidation of unorganized point clouds for surface reconstruction. [online], 2009, [Citováno 2016-12-15]. Dostupné z: http://www.cs.ubc.ca/~ascher/papers/hlzac.pdf

[62] Hoppe H., DeRose T., Duchamp T., McDonald J., Stuetzle W.: Surface reconstruction from unorganized points. [online], 1992, [Citováno 2016-12-15]. Dostupné z: http://hhoppe.com/recon.pdf

[63] Huang H., Wu S., Gong M., Cohen-Or D., Ascher U., Zhang H.: Edge-aware point set resampling. [online], 2013, [Citováno 2016-12-15]. Dostupné z: http://www.cs.ubc.ca/~ascher/papers/hwgcaz.pdf

[64] Kazhdan M., Bolitho M., Hoppe H.: Poisson Surface Reconstruction. [online], 2006, [Citováno 2016-12-16]. Dostupné z: http://hhoppe.com/poissonrecon.pdf

[65] Contributors of Wikipedia: Bounding box. [online], 2016, [Citováno 2016-12-10]. Dostupné z: https://upload.wikimedia.org/wikipedia/commons/c/cd/OpenGL_Tutorial_Bounding_box.png

[66] Li B., Holstein H.: Using k-d Trees for Robust 3D Point Pattern Matching. [online], 2003, [Citováno 2016-12-20]. Dostupné z: https://ai2-s2-pdfs.s3.amazonaws.com/5e05/0d5138f42485df550c18715709654f94883d.pdf

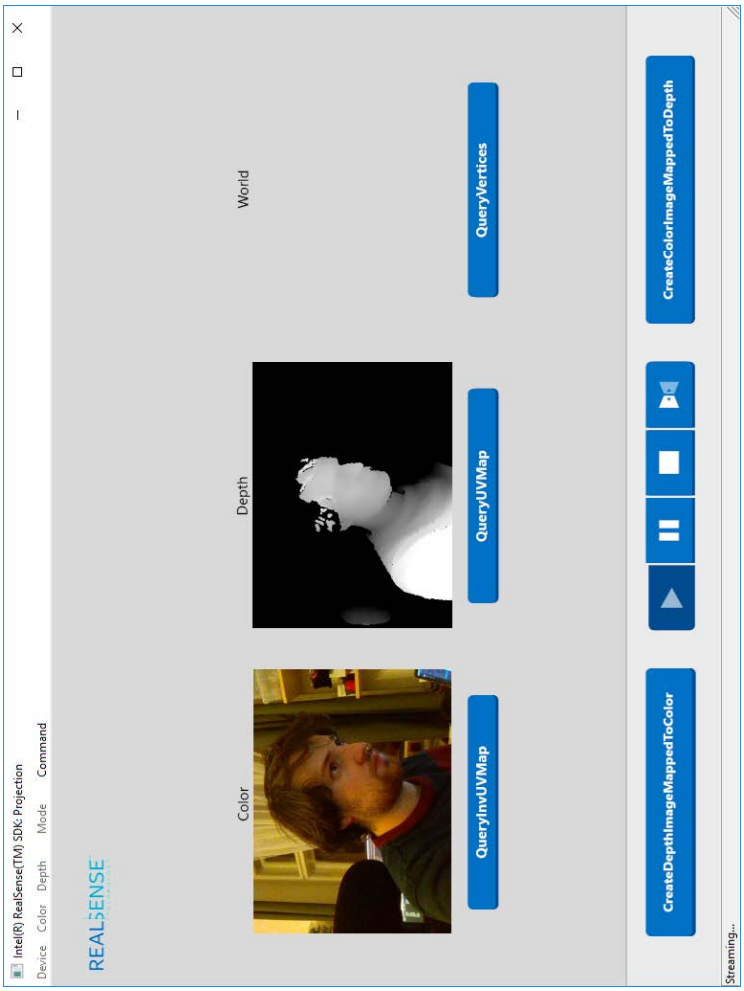
[67] Daniel Girardeau-Montaut: 3D point cloud and mesh processing software Open Source Project. [online], 2016, [Citováno 2016-12-20]. Dostupné z: http://www.cloudcompare.org

[68] Kubík, P.: *Semiadaptivní 3D modeling*. Diplomová práce, Univerzita Karlova v Praze, Matematicko-fyzikální fakulta, 2006, [Citováno 2016-12-10]. Dostupné z: https://is.cuni.cz/webapps/zzp/detail/41211/

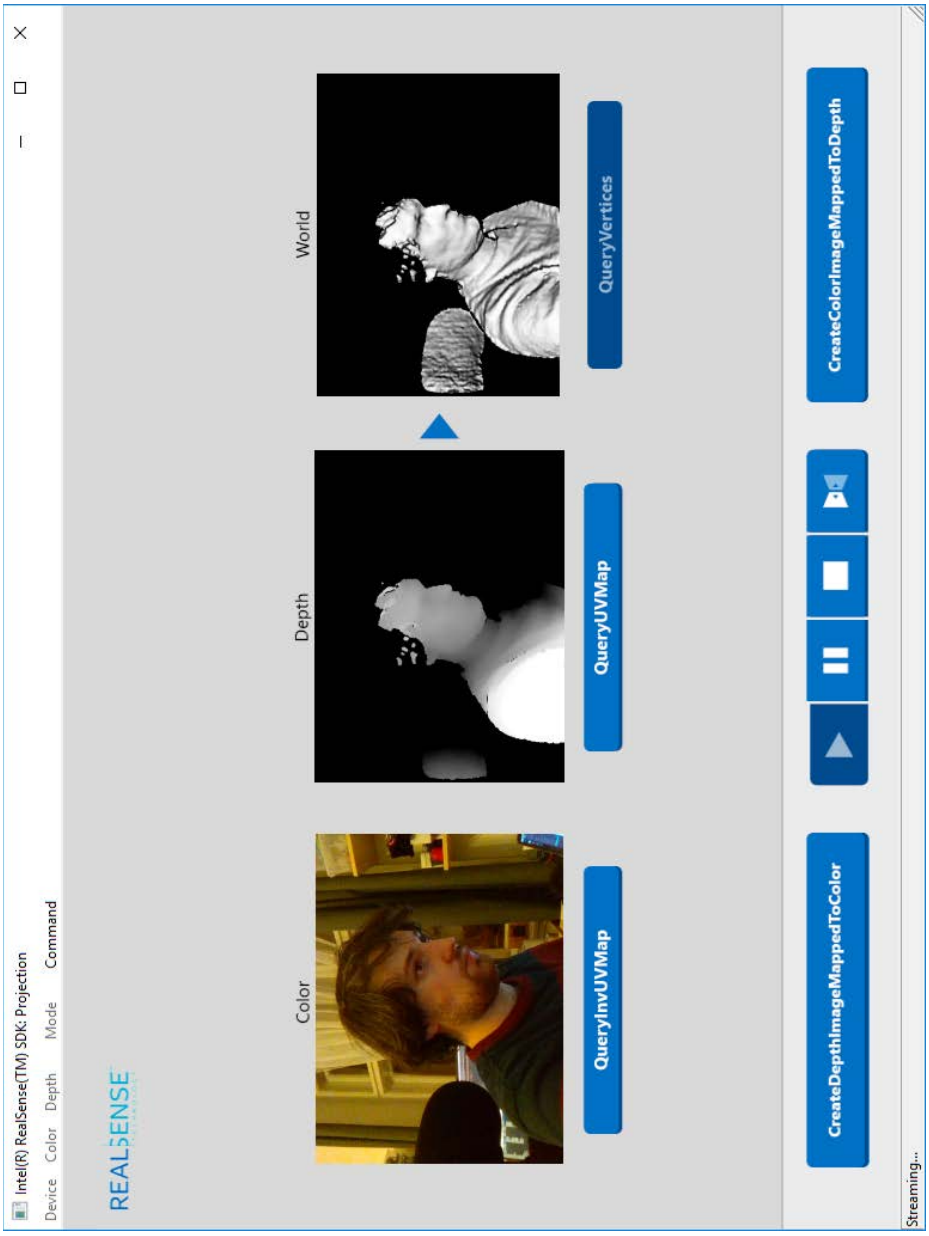
[69] Intel Corporation: Intel® Aero Platform Developer Kits. [online], 2017, [Citováno 2017-01-08]. Dostupné z: http://click.intel.com/intel-aero-ready-to-fly-drone.html

Příloha A

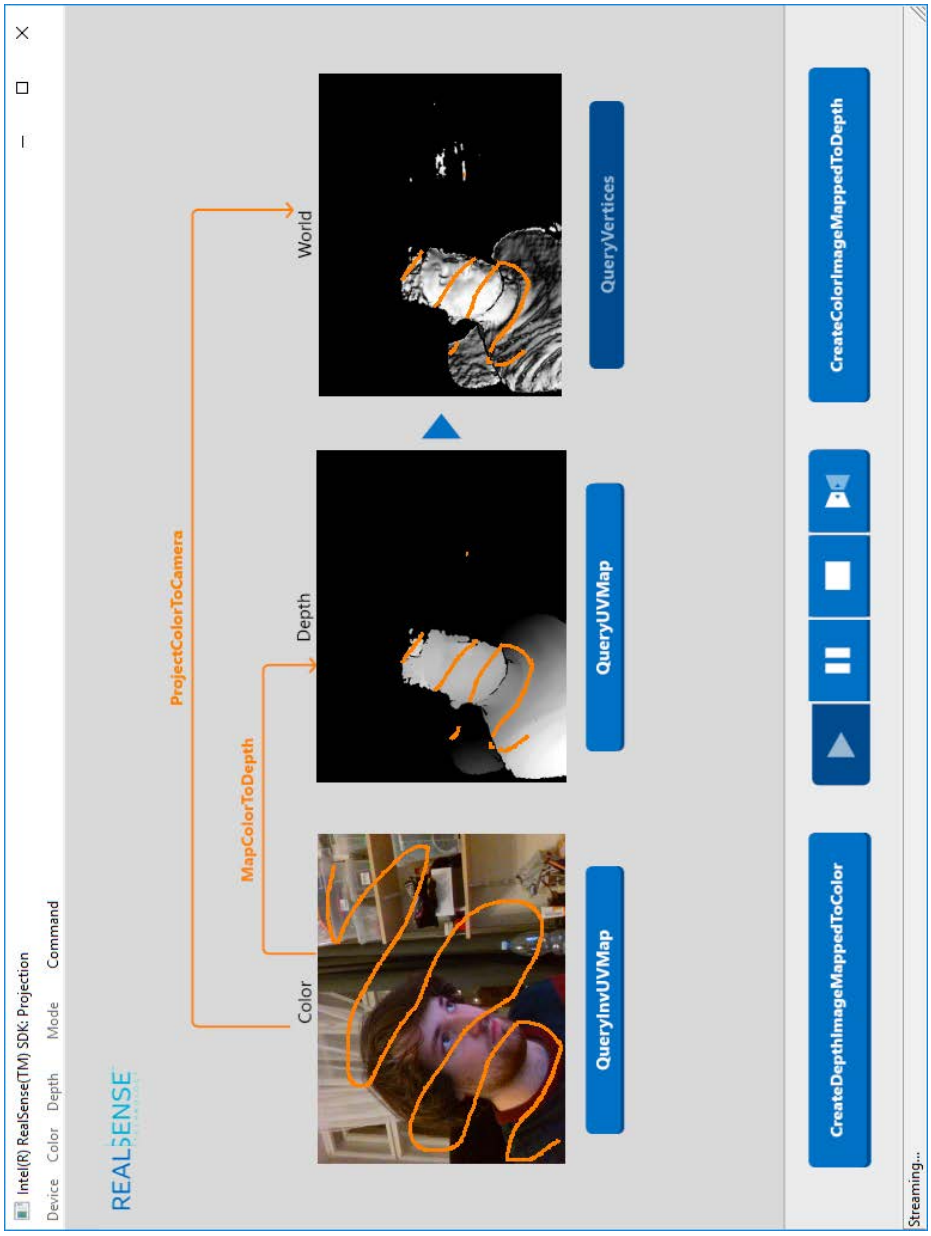
Přílohy



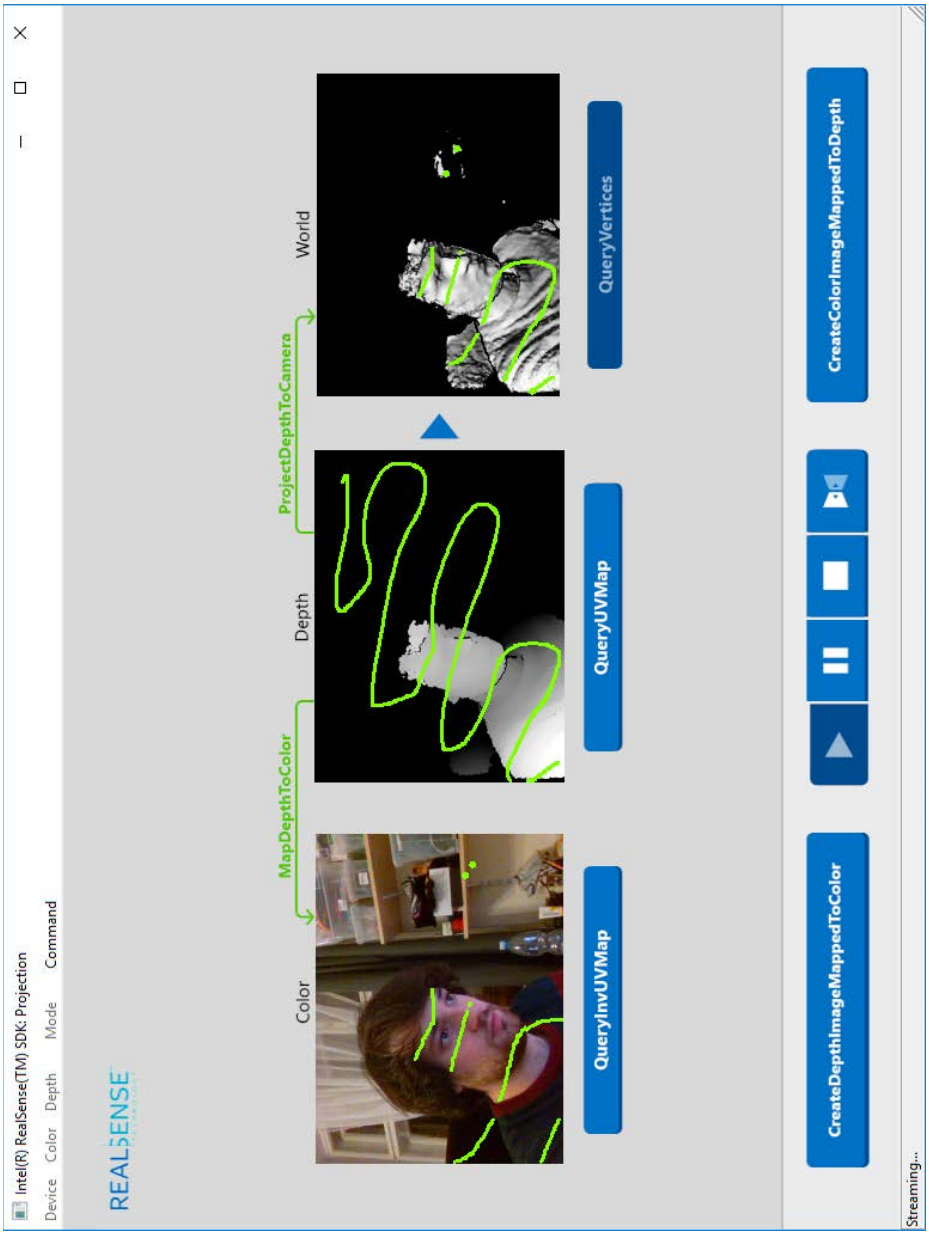
Obrázek A.1: Intel® RealSense™ Demo aplikace – ukázka RGB a hloubkové mapy



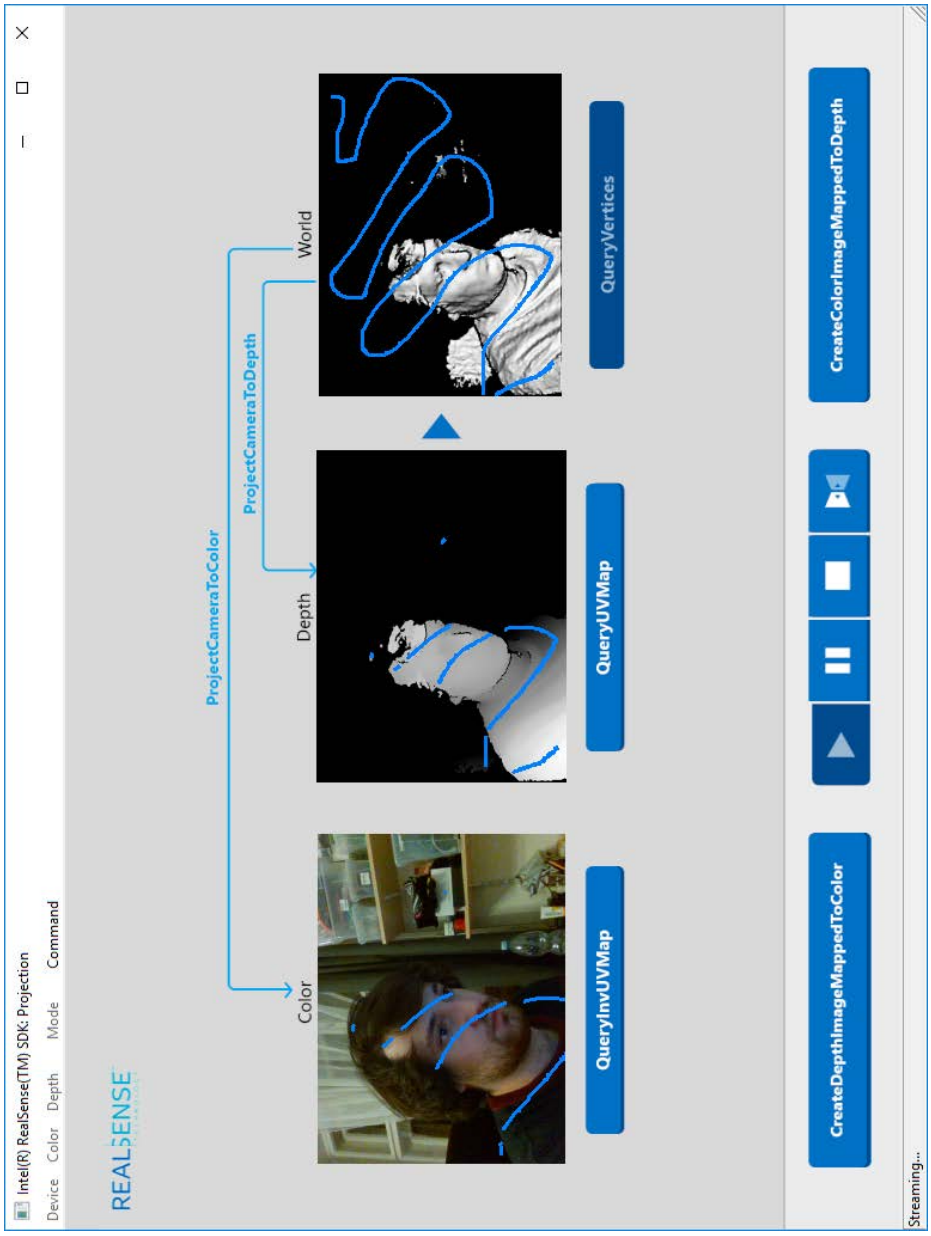
Obrázek A.2: Intel® RealSense™ Demo aplikace – ukázka převodu z hloubkové mapy do 3D modelu



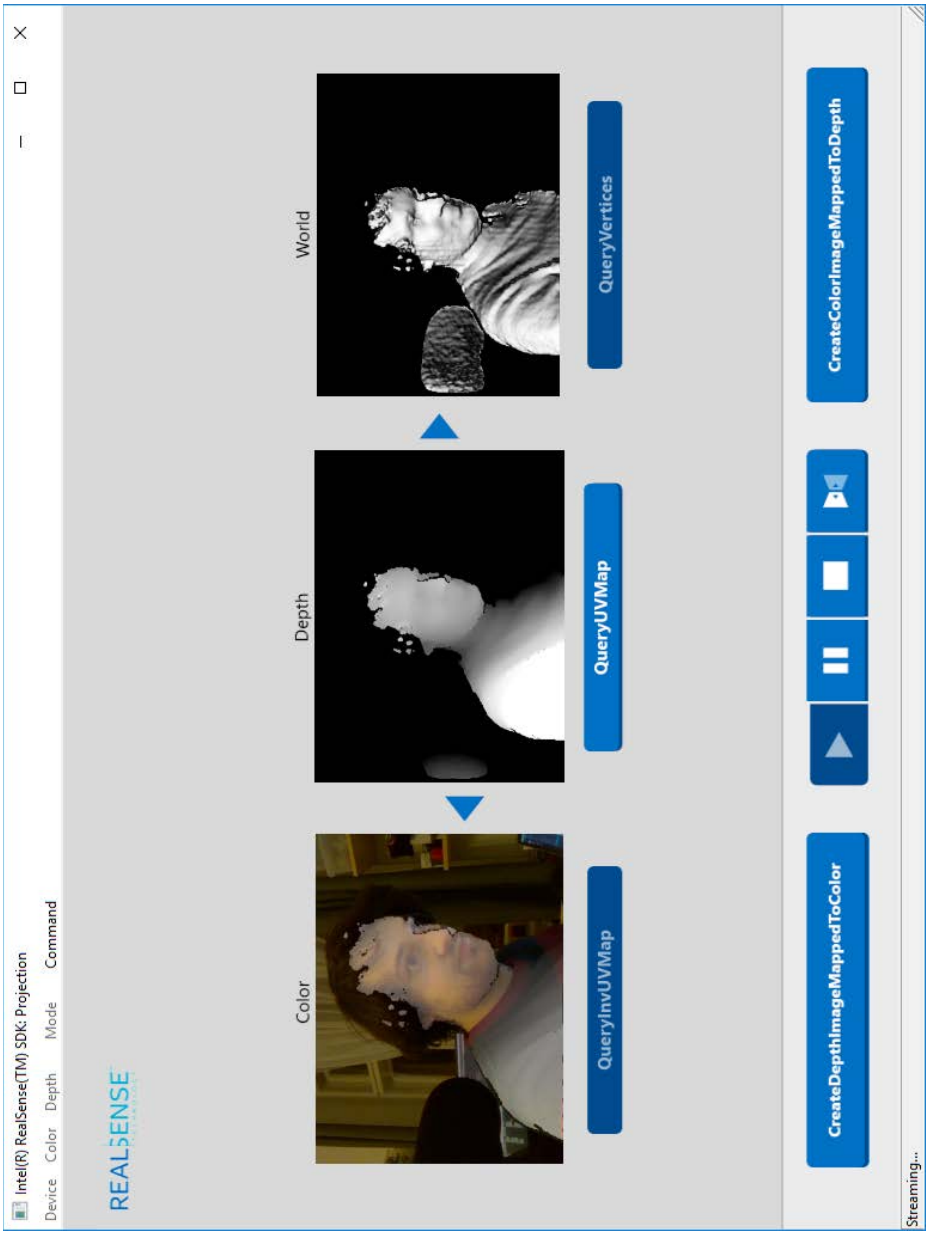
Obrázek A.3: Intel® RealSense™ Demo aplikace – ukázka promítání z RGB kamery do hloubkové mapy a 3D modelu



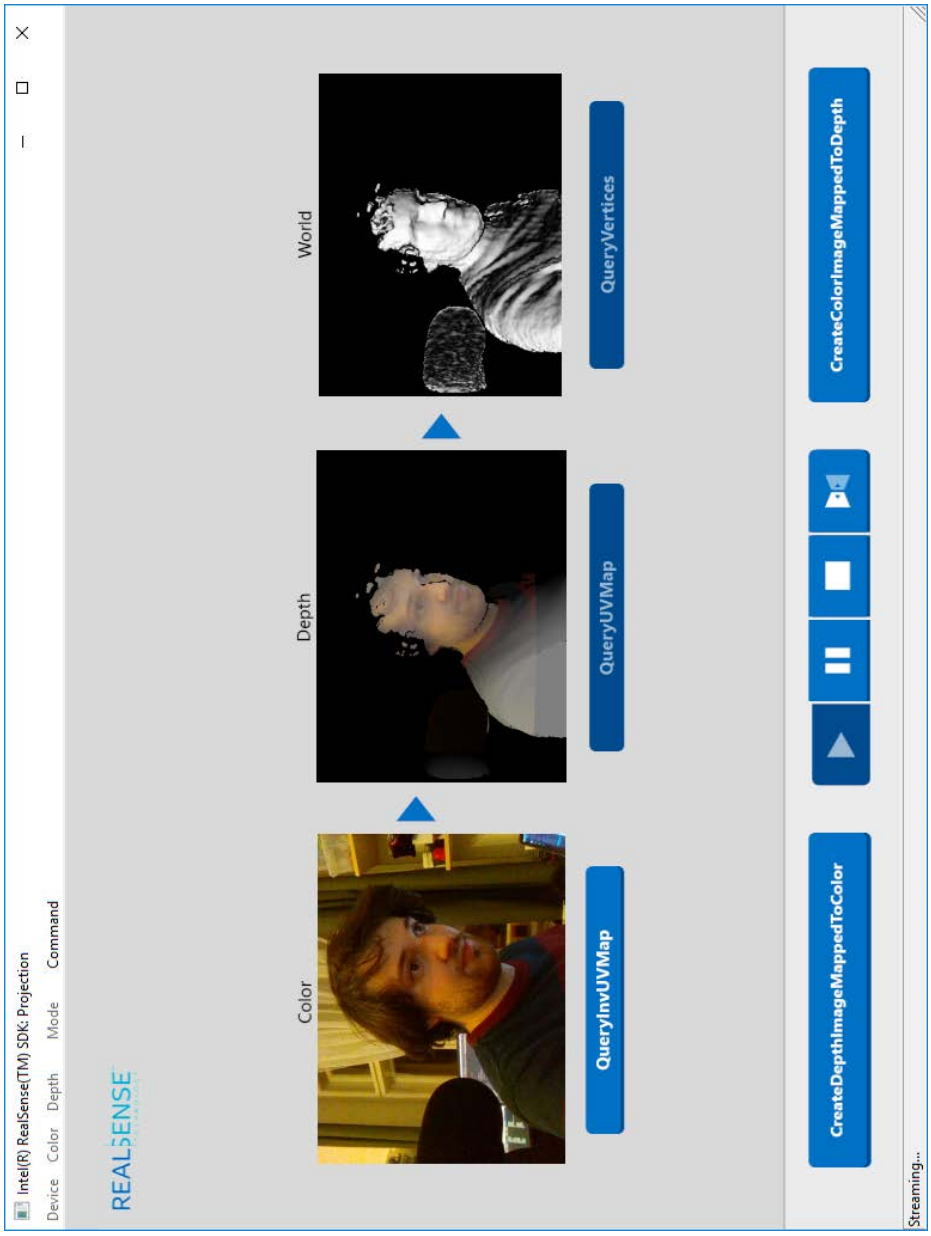
Obrázek A.4: Intel® RealSense™ Demo aplikace – ukázka promítání z hloubkové mapy do RGB kamery a 3D modelu



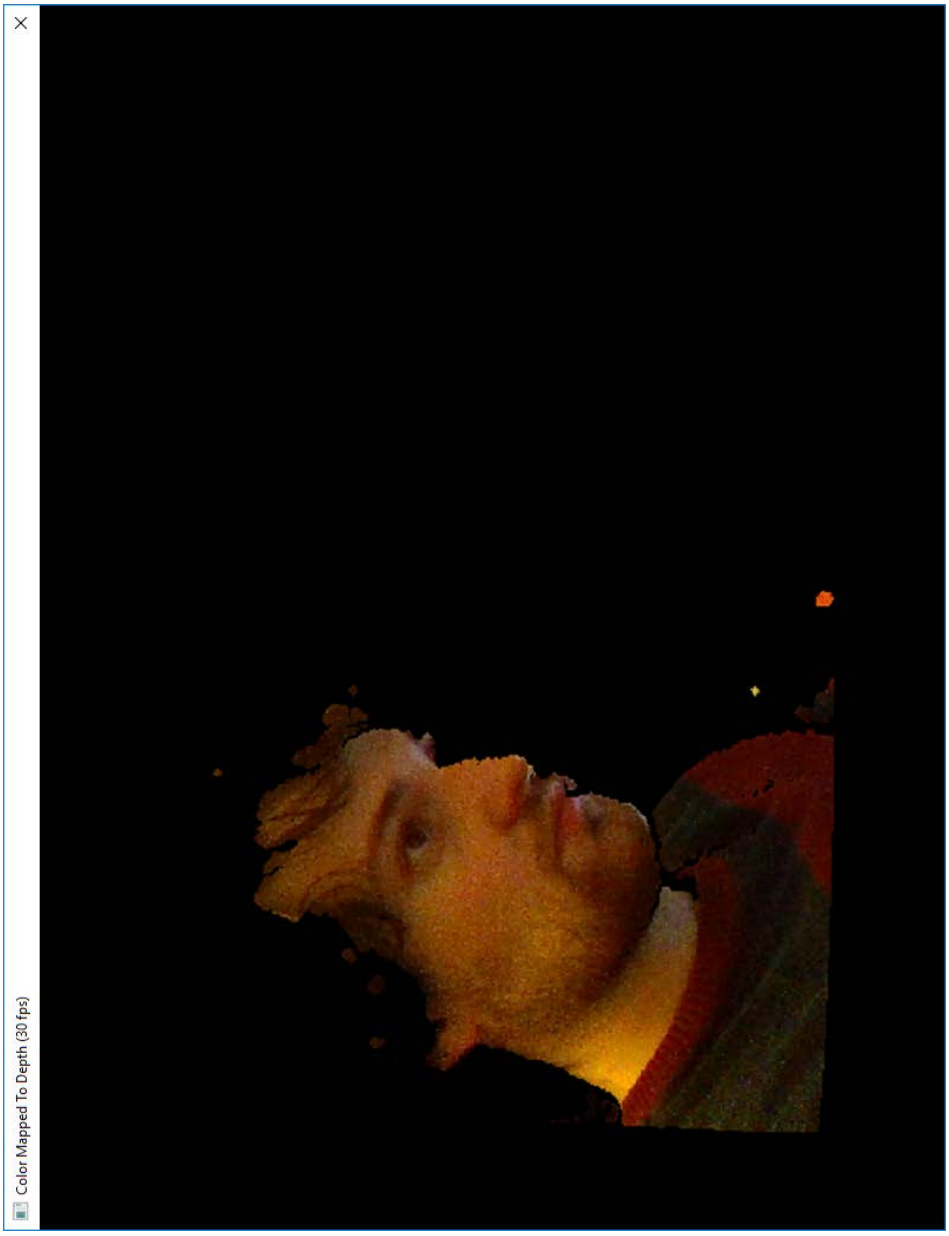
Obrázek A.5: Intel® RealSense™ Demo aplikace – ukázka promítání z 3D modelu do hloubkové mapy a RGB kamery



Obrázek A.6: Intel® RealSense™ Demo aplikace – ukázka maskování z hloubkové mapy do RGB kamery



Obrázek A.7: Intel® RealSense™ Demo aplikace – ukázka maskování z RGB kamery do hloubkové mapy



Obrázek A.8: Intel® RealSense™ Demo aplikace – ukázka maskování z RGB kamery do hloubkové mapy

Příloha **B**

Seznam použitých zkratek

ABS Akrylonitrilbutadienstyren

API Application Programming Interface

ASCII American Standard Code for Information Interchange

CGAL Computational Geometry Algorithms Library

CNC Computerized numerical control

CSG Constructive solid geometry

CSV Comma-separated values

DDR Double data rate

FDM Fused deposition modeling/method

GUI Graphical user interface

HIPS High Impact Polystyrene

ICP Iterative closest point

JPG Joint Photographic Experts Group

KNN k-nearest neighbors

NURBS Non-uniform rational basis spline

PDF Portable Document Format

PET Polyethylene terephthalate

PLA Polylactic acid

PLY Polygon File Format

PNG Portable Network Graphics

RGB Red green blue

SDK Software dev-kit

STL Stereolitography/Standard Tessellation Language

SXGA Super eXtended Graphics Array

USB Universal Serial Bus

UTF UCS Transformation Format

VGA Video Graphics Array

WLOP Weighted Locally Optimal Projection

XML Extensible markup language

Příloha C

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src` ..... adresář obsahující podklady pro jednotlivé části práce
  - `impl` ..... zdrojové kódy implementace
  - `scans` ..... naskenované modely
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `master-thesis-prusa-jakub.pdf` ..... text práce ve formátu PDF