

Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH
TECHNOLÓGIÍ

Tímový projekt

Rekonštrukcia 3D scény

Projektová dokumentácia - inžinierske dielo



Vedúci projektu:

Ing. Vanda Benešová, PhD.

Členovia tímu:

Bc. Lukáš Hudec (IS)
Bc. Róbert Birkus (IS)
Bc. Michal Löffler (IS)
Bc. Róbert Karásek (IS)
Bc. Martin Jurík (IS)
Bc. Michal Korbeľ (IS)
Bc. Katarína Janečková (IS)

Názov tímu: R3D (tím č. 5)

Web: <http://labss2.fiit.stuba.sk/TeamProject/2015/team05is-si/>

Kontakt: R3DteamTP@gmail.com

Akademický rok: 2015/2016

Dátum odovzdania: 19. november 2015

Zadanie

Cieľom projektu je návrh a implementácia metód pre rekonštrukciu 3D scény, automatické generovanie jednoduchého sémantického popisu 3D dát, registrácia texturovaných 3D dát získaných stereo rekonštrukciou ako i ich hierarchické spájanie. (Hierarchical 3D Stitching of surface patches.) Výstupom bude funkčný prototyp pre spracovanie 3D dát.

Atraktívna téma z oblasti počítačovej grafiky a počítačového videnia je orientovaná na využitie v praktických aplikáciách a aj vo výskume. Projekt bude vedený v spolupráci so skúseným tímom Machine Vision Applications Group, Institute for Information and Communication Technologies, JOANNEUM RESEARCH Graz, Austria, ktorý pod vedením Dipl.-Ing. Gerharda Paara poskytne reálne dáta, ako i cenné skúsenosti.

V súčasnej dobe existujú rôzne metódy, prípadne i hotové senzory pre získavanie 3D vizuálnych dát. Tieto majú rôzne obmedzenia v rozlíšení, presnosti a pod., prípadne tiež obsahujú rušivý šum. V rámci tohto projektu vytvoríme prototyp pre spracovanie reálnych 3D dát, pričom kľúčové úlohy budú:

- 3D segmentácia je všeobecný problém, my sa sústredíme na generovanie abstraktného popisu.
- Hierarchická registrácia 3D dát. (Hierarchical 3D Registration of surface patches.) Výzvou pri riešení tohto problému budú predovšetkým dáta s rôznym rozlíšením získané z rôznych uhlov pohľadu.
- Hierarchické spájanie 3D dát (Hierarchical 3D Stitching of surface patches) Pri riešení tejto výzvy sa pokúsime rozšíriť registrované 3D dáta o ich hladké textúrovanie spájaním jednotlivých snímok.

Projekt má výskumný charakter, umožňuje rozvinúť vlastné nápady a zároveň je smerovaný pre uplatnenie v konkrétnych aplikáciách. Jedna dôležitá budúca aplikácia by mohla byť napr. aj fúzia 3D stereo dát snímaných na Marse. (US missions MER, MSL, Mars 2020; ESA Mission ExoMars 2018).

Obsah

1	Úvod	3
1.1	Prehľad dokumentu	3
2	Globálne ciele pre ZS	4
3	Celkový pohľad na systém	6
3.1	Architektúra	6
3.1.1	Návrh architektúry systému	6
3.1.2	Modularizácia systému	8
4	Moduly systému	11
4.1	Metóda 1	11
4.1.1	Analýza	11
4.1.2	Návrh	11
4.1.3	Implementácia	12
4.2	Metóda 2	12
4.2.1	Analýza	12
4.2.2	Návrh	13
4.2.3	Implementácia	13
4.3	Metóda 3	13
4.3.1	Analýza	13
4.3.2	Návrh	14
4.3.3	Implementácia	14
4.3.4	Testovanie	14
4.4	Metóda 4	14
4.4.1	Analýza	14
4.4.2	Návrh	15
4.4.3	Implementácia	16
4.4.4	Testovanie	16
4.5	Vizualizácia	17
4.5.1	Analýza	17
4.5.2	Návrh	18
4.5.3	Implementácia	18

Kapitola 1

Úvod

Tento dokument predstavuje projektovú dokumentáciu softvérového systému na rekonštrukciu 3D scény, ktorý je výsledkom zadania na predmete Tímový projekt.

1.1 Prehľad dokumentu

V časti 2 sú spísané ciele, ktoré chceme dosiahnuť počas zimného semestra. Kapitola 3 sa zaoberá architektúrou nášho systému, je v nej tiež uvedený prehľad modularizácie systému (3.1.2) a diagram tried (3.2). Podrobný opis jednotlivých modulov systému nájdeme v kapitole 4.

Kapitola 2

Globálne ciele pre ZS

Medzi hlavné globálne ciele pre zimný semester patrí:

- Oboznámiť sa s problémom 3D rekonštrukcie a 3D registrácie dát
- Analyzovať a navrhnúť základnú architektúru aplikácie pre rekonštrukciu obrazu
 - Vybrať knižnice pre implementáciu
 - Navrhnúť základné triedy a metódy
- Načítať 3D dáta v možných formátoch
 - Načítanie dát pomocou funkcií použitých knižníc
 - Načítanie pomocou C/C++ funkcií a prekonvertovanie do použiteľného formátu
- Preskúmať, oboznámiť sa a vybrať metódy, použiteľné pre rekonštrukciu obrazu
 - Výber metód pre 3D rekonštrukciu a 3D registráciu dát
 - Štúdium a pochopenie zvolených metód
 - Implementácia a otestovanie daných metód
- Vizualizácia implementovaných metód ich prostredníctvom vytvorenej aplikácie
 - Navrhnúť GUI aplikáciu pre interakciu s používateľom
 - Implementovať načítanie súborov a výber metód pre spracovanie vstupných dát
 - Nastavenie parametrov zobrazenia vizualizácie
 - Vizualizovať výstup spracovania pomocou zvolených metód

- Zhodnotenie práce za dané obdobie
- Konzultácia s product ownerom
 - Predvedenie výsledku práce
 - Dohodnutie ďalšieho postupu podľa požiadaviek

Ciele z pohľadu jednotlivých šprintov za zimný semester:

Šprint 1: Štúdium pojmov a odporúčanej literatúry ku knižniciam, analýzy metód, návrh rozhraní projektu a načítanie dát

Šprint 2: Prvé implementácie častí a metód spracovania point cloudov, testovanie datasetov

Šprint 3: Implementovanie prvého prototypu GUI, segmentácia plôch objektov, jednoduchá vizualizácia a pohyby kamerou

Kapitola 3

Celkový pohľad na systém

3.1 Architektúra

3.1.1 Návrh architektúry systému

Náš projekt 3D rekonštrukcie scény prichádza ako samostatná nová aplikácia vyvíjaná naším tímom od prvého riadku zdrojového kódu. To znamená, že pre novú, takto rozsiahlu aplikáciu, na ktorej pracuje 7 členný tím bolo potrebné vytvoriť vhodnú architektúru, aby neskôr nedošlo k problémom priamo z tohto hľadiska implementácie riešenia.

Celkovému návrhu predchádzala podrobná analýza dostupných technológií – knižníc, ktoré sa zaoberajú problematikou 3D rekonštrukcie a vizualizácie 3D dát, najvhodnejšie oblaku bodov. Pre určité metódy sme zobrali do úvahy aj prácu nad 2D dátami. Dostupné¹ a vhodné technológie sa ukázali knižnice PCL „Point Cloud Library“ a knižnica OpenCV. Vzhľadom na komplexnosť problému, veľkosti spracúvaných dát a výpočtovej zložitosti problému sme sa rozhodli pre zrýchlenie na GPU pomocou knižnice CUDA. Z analýzy naďalej vyplýva, že naším cieľom je vytvoriť prototyp desktopovej aplikácie s extrahovateľnými modulmi rekonštrukcie a prípadnej segmentácie objektov.

Problémy, ktoré by mohli nastať z hľadiska implementácie sú ľahko predstaviteľné. Dokonca prvý problém sa vyskytol ešte v prvých fázach a bol okamžite odstránený. Zlý návrh znamenal problém pri implementácii ďalšieho funkčného rozšírenia. Tým, že na vývoji pracuje súčasne 7 ľudí a ich súčasti musia byť systematicky prepájané, je potrebné dbať na určité zásady. K týmto zásadám sa viaže aj metodika písania zdrojového kódu.

Pre dosiahnutie bezproblémovej implementácie a jednoduchej integrácie jednotlivých modulov a vyvíjaných funkčných modelov musí byť architektúra projektu škálovateľná a robustná voči rôznym zmenám, ktoré sa počas

¹Vzhľadom na zadávateľa – výskumné centrum a charakter zadania potrebovali sme Open Source a GNU GPL licencie

vývoja vyskytnú. Najdôležitejšie vlastnosti, na ktoré architekt dával najväčší dôraz sú:

- Dátovo-formátová nezávislosť
 - Vzhľadom na to, že nedisponujeme zariadením na vytvorenie vlastného datasetu a centrum, pre ktoré vyvíjame prototyp nám ešte nedodalo testovací dataset, boli sme nútení nájsť alternatívu vo voľne dostupných datasetoch.
 - Problém takéhoto prístupu je jasný – dostupné datasety sú rôzneho typu, rôzneho formátu a preto je tento problém riešiť už v návrhu architektúry vytvorením modulu parsera dát.
 - Ďalším riešením je vytvoriť si syntaktické dáta – tieto sú však vhodné iba na testovanie a ich formát sa nemusí zhodovať s reálnym formátom, ktorý dostaneme zo snímacieho zariadenia.
- Dátovo-typová nezávislosť
 - Získané dáta môžu byť rôzneho typu – čisté XYZ, ale môžu obsahovať aj hodnotu intenzity či farebnú zložku. Prototyp by si mal vedieť s týmto problémom poradiť automaticky a nežiadať ďalšie pomocné informácie od používateľa.
- Dátovo-funkcionálna nezávislosť
 - Vývojári jednotlivých funkčných modelov potrebujú vyvíjať metódy a nie riešiť problémy s dátovými typmi. Potrebujú len vedieť kde ich nájsť, ako sa volajú a keď potrebujú iný typ ako štandardne volané XYZ.
- Škálovateľnosť, Flexibilita voči zmenám a Modulovateľnosť
 - Podstata celého výskumného prototypu je v tom, že v priebehu vývoja sa implementuje niekoľko rôznych riešení zadaného problému. Tieto riešenia budú pridávané postupne a do rôznych modulov podľa ich charakteristického zamerania a funkčného významu.
 - Architektúru je preto potrebné navrhnuť tak, že jednotlivé moduly budú ľahko doplniteľné a v prípade potreby vymeniteľné za iný modul.
- Nezávislá integrácia externých zdrojov
 - Výstupom nášho prototypu má byť najmä knižnica, ktorej funkcionality je rekonštrukcia 3D dát a prípadne segmentácia objektov. Pre tento výstup je nežiaduce, keby bola táto knižnica viazaná na tematicky odlišnú funkcionality – gui, prípadne vizualizáciu,

prípadne ovládač. Preto je potrebné navrhnuť štruktúru tak, aby nevznikali nežiadúce prepojenia na externé zdroje, s ktorými osobitné moduly pracujú.

- Údržba pamäte a šetrné zaobchádzanie so zdrojmi
 - Tento bod je spojený tiež s druhom dát, s ktorými program pracuje. Tieto dáta nie je možné spracúvať postupne a mať ich uložené na externom úložisku.
 - Tieto dáta musia byť načítané v programe celý čas ich spracovania aj zobrazovania.
 - Problém nastáva keď si uvedomíme, že niektoré datasety majú viac ako 2GB v ASCII formáte a po načítaní cez 500MB.
 - Z tohto dôvodu je potrebné neplytvat pamäťou a zdieľať čo najväčšie množstvo dát, ktoré je možné.

3.1.2 Modularizácia systému

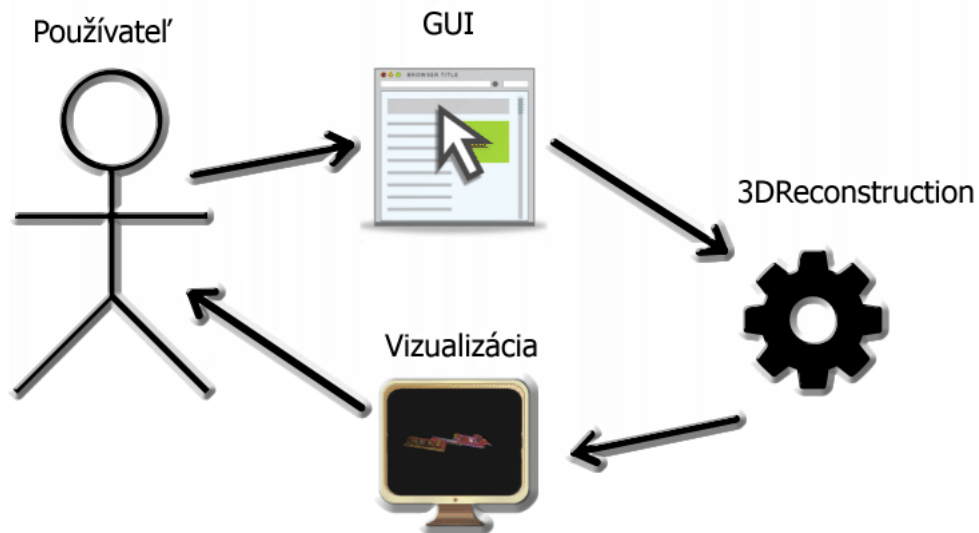
Z architektonického hľadiska nezávislosti modulov vyplynulo rozdelenie problémov na 3 hlavné moduly. Z hľadiska plánovania a rozdelenia projektu tieto moduly predstavujú Epic úlohy evidované v TFS backlog-u. Tieto moduly sú:

- 3DReconstruction - hlavná logika aplikácie
- GUI - Grafické rozhranie pre komunikáciu s používateľom
- Vizualization - Vizualný výstup riešenia ako spätná informácia pre používateľa (Okrem vizualizácie rieši aj interakciu)

Diagram toku dát:

3DReconstruction

Tento modul obsahuje hlavnú našim tímom vyvíjanú logiku rekonštrukcie a segmentácie. Obsahuje hlavnú triedu „ClosedSpace“, ktorá spravuje dáta, nad ktorými sa vykonávajú operácie rekonštrukcie. Okrem toho, bol zámer ju vytvoriť typovo nezávislú, takže dokáže udržiavať dátové typy akéhokoľvek formátu. Ďalšou z hľadiska architektúry významnou triedou je „Reconstruction“ abstraktná trieda, ktorá je rodičom každej triede, ktorá sa venuje rekonštrukcií scény. Tento prístup bol smerovaný najmä z hľadiska modularity a „oop“ polymorfizmu. Z hľadiska správy dát a rekonštruovaných typov sú tiež významné triedy „Object“ a „Primitives“, ktoré spravujú rekonštruované typy a udržiavajú ich dáta. Takto bola dosiahnutá maximálna abstrakcia typov a v ďalších implementáciách objektový polymorfizmus.



Obr. 3.1: Diagram toku dát

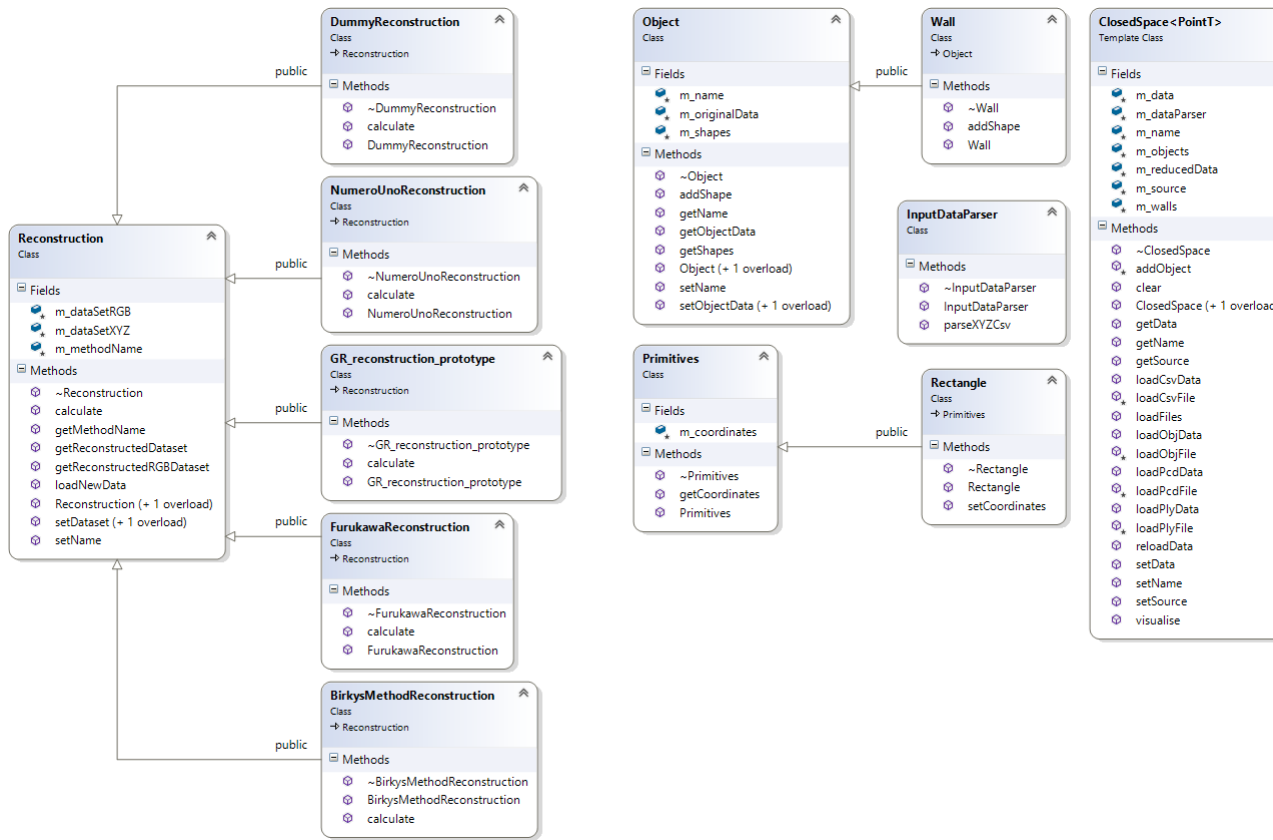
GUI

Každý program potrebuje grafické rozhranie pre interakciu s používateľom. Špeciálne vtedy, keď potrebuje od používateľa aby si vybral metódu, ktorou chce vykonať rekonštrukciu, prípadne dáta ktoré chce vizualizovať. Keďže je to tematicky odlišný modul, bol tak aj vytváraný a ostatné moduly od neho ostali architektonicky nezávislé. Jeho štruktúra je jednoduchá a prepojenie je riešené priamo cez kontroler. Môžeme povedať, že čo sa modulov týka, architektonický vzor použitý na implementáciu sa snažil priblížiť MVC.

Vizualization

Vizualizačný model mohol byť prepojený priamo na GUI, dôvod jeho oddelenia však spočíva v rozdielnych metódach použitých pri ich implementácií – na čo bolo potrebné myslieť už v rámci návrhu architektúry. Rovnako ako vizualizáciu tiež rieši aj interakciu s používateľom a pohyb kamery. Preto tiež aj z hľadiska čiastočnej tematickej odlišnosti bol tento modul oddelený od GUI.

KAPITOLA 3. CELKOVÝ POHLAD NA SYSTÉM



Obr. 3.2: Diagram tried

Kapitola 4

Moduly systému

Rekonštrukcia 3D scény v dnešnej dobe už nie je problém. Existuje viacero metód, ktorými dokážeme vytvoriť modely s miliónmi bodov, s veľkou mierkou a s vysokým rozlíšením. Výzvou však je vytvoriť zjednodušený model, čo možno dokázať použitím určitého stupňa abstrakcie a geometrizáciou. Taktéto modely sú, okrem iného, oveľa lepšie použiteľné. Pracovať so zložitým modelom, ktorý obsahuje niekoľko miliónov bodov, nie je príliš praktické. Hovoríme napríklad o spracovaní, prenášaní, analyzovaní alebo vizualizácii. V našom projekte sme sa zamerali na rekonštrukciu miestnosti. Jednotlivé plochy a objekty v 3D dátach sú reprezentované príliš veľa bodmi. My chceme tieto plochy a objekty rozpoznať a nahradiť ich len niekoľkými bodmi, ktoré budú predstavovať ich zjednodušený geometrický tvar. Prekážkou, ktorú treba rozumným spôsobom prekonať, je nerovnomernosť, zašumenosť a nepresnosť v rekonštruovaných dátach. Rozhodli sme sa analyzovať niekoľko metód, ktoré sú opísané v častiach 4.1, 4.2, 4.3 a 4.4.

4.1 Metóda 1

4.1.1 Analýza

Metóda sa zakladá na článku [1]. V publikácii sa zaoberajú tvorbou zjednodušeného geometrizovaného modelu viacposchodovej budovy. Pri tejto metóde je predpoklad, že vstupný oblak bodov obsahuje body prevažne tvoriace horizontálne a vertikálne plochy.

4.1.2 Návrh

Postup je nasledovný: Najprv chceme nájsť rezy modelu. Rez predstavuje takú časť modelu, ktorú ohraničujú dominantné horizontálne štruktúry. Tieto štruktúry nájdeme tak, že zistíme, ktoré body majú podobnú normálu ako os z a premietneme ich na ňu. Najhustejšie miesta v týchto 1D dátach budú predstavovať hlavné horizontálne štruktúry. Aplikujeme na ne metódu

mean shift, ktorou nájdeme stredy najhustejších oblastí, teda horizontálnych štruktúr a v týchto miestach povedieme rezy modelom. Každý rez premietneme do 2D priestoru a pomocou binárnej segmentácie označujeme pixely ako v rámci objektu alebo mimo objektu. Ohraničíme tie, ktoré do objektu patria a získame 2D plán jedného rezu. Obrisy objektov každého rezu prevedieme do 3D a jednotlivé rezy pospájame. Kvôli nepravidelnostiam medzi rezmi navzájom urobíme s modelom 3D regularizáciu, čím dostaneme výsledný model rekonštruovaných dát.

4.1.3 Implementácia

Implementácia metódy je v začiatkoch, zatiaľ sa podarilo vytvoriť metódu na určenie normál všetkých bodov, nájdenie bodov s normálou podobnou osi z a následné premietnutie týchto bodov na os z .

4.2 Metóda 2

4.2.1 Analýza

Growing region je segmentačná metóda založená na porovnávaní susedných bodov v oblaku a následnom vytvorení vyfiltrovaných oblastí (regiónov) prislúchajúcich bodov. Na začiatku je potrebné si stanoviť tzv. "seed" bod, ktorý považujeme za štartovací bod. Od neho sa vyvíja celý proces segmentácie obrazu. Po stanovení počiatočného seed bodu a vypočítaní normály pre tento bod, môže začať proces segmentácie. Zoberieme susedné body v oblaku a vypočítame ich normály. Následne porovnáme uhol medzi normálou seed bodu a normálou každého susedného bodu. Ak uhol spadá pod stanovenú prahovú hodnotu, susedné body z oblaku patria do novovysegmentovanej oblasti. V opačnom prípade body vynecháme. V ďalšom kroku za seed body teraz považujeme novopridané body do oblasti a opakujeme krok algoritmu. Proces segmentácie a samotné porovnávanie susedných bodov sa taktiež môže uskutočňovať na základe RGB hodnôt jednotlivých bodov v oblaku. Rovnako sa stanoví prahová RGB hodnota a v prípade, že rozdiel dvoch susedných bodov spadá pod túto stanovenú hodnotu, oba body patria do jednej spoločnej oblasti.

Po segmentačnom procese každá oblasť (plocha) obsahuje určitý počet bodov. Takto zostavený model je komplikovaný a je náročnejšie ho vizualizovať, než keby jednotlivé plochy boli prekryté geometrickým útvarom (trojuholník, polygón...). Preto sa snažíme prekryť čo najviac bodov takýmto útvarom. Rovnako je potrebné riešiť body, ktoré nie sú priradené žiadnej ploche, tzv. "outliers", tieto pravdepodobne vymažeme z oblaku, resp. táto otázka bude riešená v neskoršej fáze projektu.

Mnohé výpočty potrebné pre algoritmus sú priamo zahrnuté v `pcl` knižnici, ktorú používame pri riešení projektu, a preto nemusíme riešiť imple-

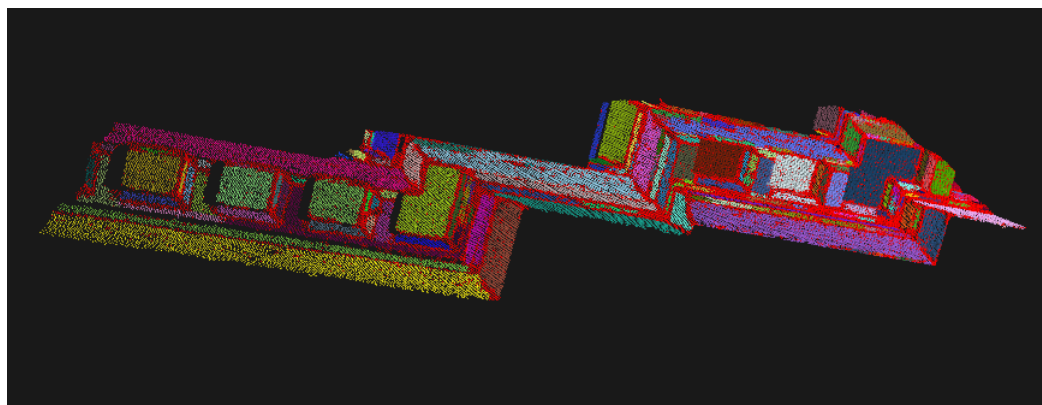
mentácie elementárnych funkcií výpočtov. Napríklad v tomto prípade sú to normály, prípadne výpočet RGB hodnoty z bodov a podobne.

4.2.2 Návrh

Nasledovným krokom bude vysegmentovanie celých plôch a ich následné prekrytie geometrickými útvarmi, čím sa zjednoduší "model oblaku bodov" na "model oblaku plôch". Popri implementácii si metóda rovnako vyžaduje ďalšie štúdium.

4.2.3 Implementácia

V súčasnosti je vytvorený "prototyp" algoritmu, ktorý segmentuje oblak bodov na jednotlivé oblasti (plochy) bodov. Každá takto vytvorená plocha obsahuje body označené jednou príslušnou farbou. Vo vysegmentovanom oblaku nájdeme aj body nespádajúce pod žiadnu plochu, v obrázku 4.1 sú to červené body.



Obr. 4.1: Metóda growing region

4.3 Metóda 3

4.3.1 Analýza

Daná metóda je založená na článku z našej literatúry. Metóda je zo začiatku dosť jednoducho a neopísaná do detailov, preto ju bude potrebné prispôbiť, možno aj zlepšiť v niektorých častiach. Ako prvý bod metódy je potrebné vytvoriť histogram z bodov v smere gravitácie. Po aplikovaní niekoľkých algoritmov nám rozdelí 3D priestor na 2D rezy z ktorých sa vytvorí CSG model. Tie sa neskôr pospájajú a vytvoria nám miestnosť.

4.3.2 Návrh

Postupovať podľa článku a prispôbovať algoritmy, nakoľko nie je tam opísaný do detailov.

4.3.3 Implementácia

Z implementačnej bol zatiaľ vytvorený histogram, ktorý bude neskôr použitý pre rozdeľovanie celkového 3D priestora. Nakoľko z článku nie je jasné ako bol implementovaný, je veľká pravdepodobnosť, že sa počas implementácie môže zmeniť.

4.3.4 Testovanie

Testovanie bolo na našom prvom datasete a testovala sa ešte Houghova transformácia, ktorá bude potrebná pre neskoršiu fázu vývoja.

4.4 Metóda 4

4.4.1 Analýza

Táto metóda je založená na histograme normál bodov. Cieľom je pomocou výpočtu normál jednotlivých bodov (pomocou ich okolia) určiť dominantné smery v oblaku bodov a tým určiť potenciálnu orientáciu stien v analyzovanej miestnosti. Táto metóda by mala úspešne fungovať pri typických miestnostiach s rovnými stenami. Avšak, je dosť pravdepodobné, že táto metóda zlyhá pri atypických miestnostiach s oblúkovitými stenami. Ďalšími problémami tejto metódy môžu byť rôzne objekty v miestnosti, či už na stenách alebo v samotnej miestnosti. Tento problém by sa však mohol riešiť pomocou predspracovania oblaku bodov danej miestnosti, pomocou ktorej by sa tieto objekty odstránili.

Samotná metóda sa skladá z niekoľkých krokov, konkrétne sú to: výpočet normál bodov, vytvorenie histogramu normál, nájdenie dominantných smerov z histogramu normál, označovanie jednotlivých bodov podľa dominantných smerov, získanie bodov jednej roviny(steny), vytvorenie primitív pre steny.

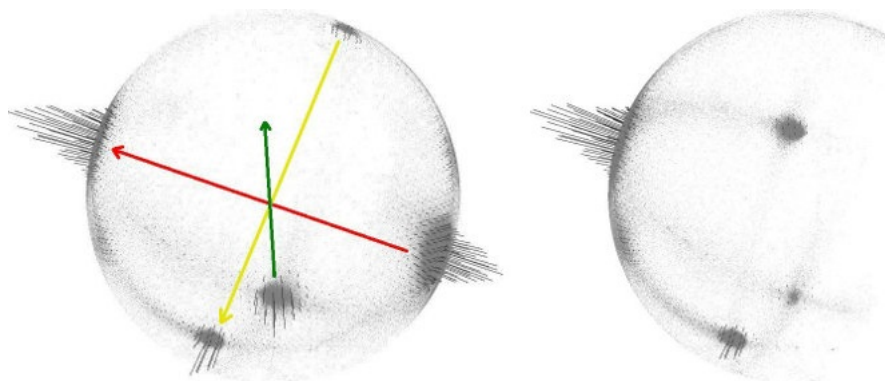
V našom projekte používame knižnicu PCL(Point Cloud Library) a OpenCV, ktoré sa snažíme maximálne využiť pre uľahčenie našej práce. PCL umožňuje výpočet normál bodov. Normály jednotlivých bodov vypočíta pomocou okolitých bodov zo zadaného okolia. Vypočítané normály bodov obsahujú normalizovaný vektor x , y , z a tiež aj zakryvenie. Podrobné vysvetlenia výpočtu normál bodov v PCL je na oficiálnej stránke PCL. Možnosti PCL pre histogram sú vcelku obmedzené, keďže obsahuje iba implementáciu špecifických histogramov pre špecifické úlohy, ktoré pre našu úlohu nie

sú vhodné. Je teda potrebné navrhnuť a naimplementovať vlastný histogram pre normály bodov. Pre určenie dominantných oblastí v histograme bude potrebné si zvoliť vhodný algoritmus, ktorý závisí predovšetkým od navrhnutého histogramu. Predpokladá sa použitie algoritmu mean-shift. Po zistení dominantných smerov v oblaku bodov je potrebné prideliť každý bod k niektorému z dominantných smerov. Ďalším problémom je, že body dvoch alebo viacerých rovnobežných stien budú pridelené k rovnakému dominantnému smeru a pre ďalšie spracovanie potrebujeme, aby sme mali jednotlivé steny zvlášť vysegmentované. Keď už budeme mať všetky body pridelené k jednotlivým stenám, budeme môcť vytvoriť primitívy stien.

4.4.2 Návrh

Pre výpočet normál použijeme existujúce funkcie PCL knižnice, ktoré fungujú na princípe PCA (Principal Components Analysis). V podstate ide o výpočet kovariačnej matice pre každý bod pomocou okolitých susedných bodov.

Keďže PCL neumožňuje vytvárať histogram, ktorý je potrebný pre našu úlohu, navrhli sme vlastný. Pre zjednodušenie histogramu, aby sme nemuseli ukladať všetky tri hodnoty normál (x,y,z) sme si zjednošili úlohu výpočtom dvoch uhlov, ktoré určujú normálu. Sú to uhly alfa a beta. Alfa je uhol medzi x-ovou a y-ovou súradnicovou osou a beta je uhol medzi x-ovou a z-ovou súradnicovou osou. Týmto riešením nám stačí ukladať iba dve hodnoty, čím šetríme pamäť. Taktiež v našom prípade normála a normála k nej inverzná je pre nás takpovediac identická, totiž nám obi dve z hľadiska určenia smeru stien dávajú rovnaký smer steny. Preto by tieto normály mali zahlasovať do histogramu rovnako. Pre lepšiu predstavu si pozrite 4.2.



Obr. 4.2: Všetky normály hlasujúce rôzne (naľavo), normály a normály k nim inverzné hlasujúce rovnako (napravo)¹

¹http://www.pointclouds.org/documentation/tutorials/normal_estimation.php

Histogram normál teda bude 2D histogram, ktorý bude reprezentovať smer normál pomocou dvoch uhlov alfa a beta od 0° po 180° .

Pre určenie dominantných oblastí v histograme je najprv potrebné vizualizovať histogram, podľa ktorého sa bude dať lepšie odhadnúť najvhodnejšiu metódu pre danú úlohu. Vizualizácia sa bude riešiť pomocou knižnice OpenCV, keďže sa jedná o 2D histogram, ktorý sa dá zobrazíť ako obrázok. Avšak, je dosť pravdepodobné, že sa pre nájdenie dominantných oblastí v histograme použije algoritmus mean-shift.

Po získaní dominantných smerov oblaku bodov, pridelieme každý bod k určitému dominantnému smeru, ku ktorému sa normála bodu podobá čo najviac. Normály bodov, ktoré sa budú odlišovať od každého dominantného smeru príliš (prekročia prah) sa nepridelia k žiadnemu. Toto sa vykoná iteráciou cez všetky body a zistením najpodobnejšieho dominantného smeru k ich normál. Po označení jednotlivých bodov podľa dominantných smerov oblaku bodov je ešte potrebné odlíšiť body patriace k rôznym rovinám (stenám). Čiastočne nám to už označovanie podľa dominantných smerov spravilo. Avšak, roviny (steny) ku sebe rovnobežné je ešte stále potrebné od seba oddeliť. To sa vyrieši pomocou výpočtu vzdialeností jednotlivých bodov. Predpokladá sa, že body dvoch rovnobežných rovín budú v istej osy podstatne vzdialenejšie od seba.

A na záver, po pridelení všetkých bodov k jednotlivým rovinám (stenám) bude potrebné vytvoriť primitíva pre jednotlivé steny. Toto chceme docieľiť pomocou zredukovania bodov v oblaku bodov jednotlivých stien a následného vykreslenia plochy. Umiestnenie vykreslenej plochy bude potrebné zkorrigovať podľa aritmetického priemeru alebo mediánu bodov v oblaku bodov.

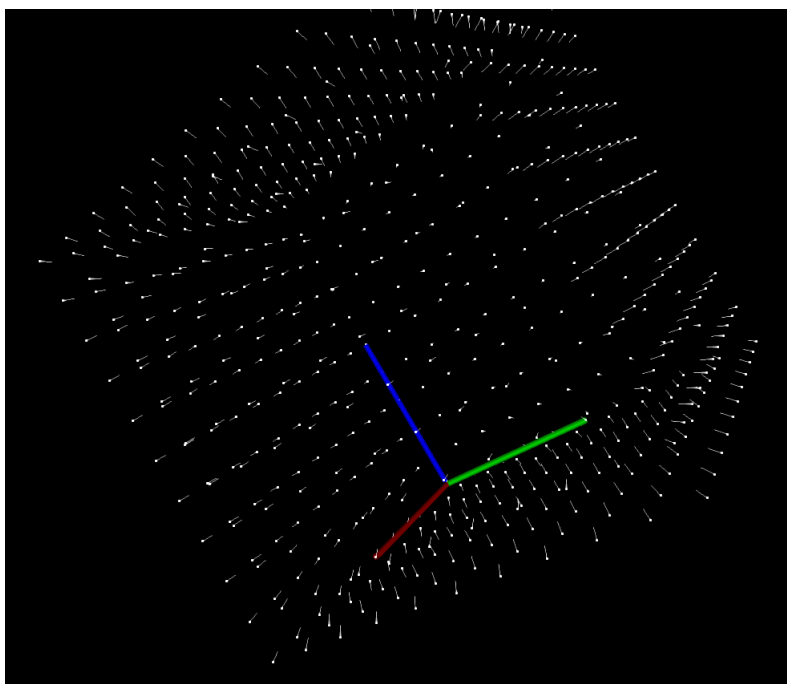
4.4.3 Implementácia

Implementácia prebehla v jazyku C++ pomocou knižníc OpenCV a PCL. Vstupom pre tento modul je oblak bodov a výstupom sú body potrebné na vykreslenie plôch reprezentujúcich steny.

Aktuálne modul podporuje len výpočet normál, vytvorenie histogramu normál a nájdenie dominantných smerov z histogramu normál.

4.4.4 Testovanie

Testovanie prebieha súbežne pre všetky kroky modulu. Prevažne sa jedná o manuálne vizuálne testovanie pomocou vizualizácie knižnice PCL a OpenCV. Pre testovanie jednotlivých krokov modulu používame rôzne vstupné dáta. Pre testovanie sme si vytvorili tiež syntetické dáta. Vytvorili sme oblak bodov pravidelnej kocky, nepravidelnej kocky a kocky s objektmi vo vnútri. Jednotlivé kroky modulu najprv testujeme na týchto syntetických vstupných dátach a následne až na reálnych dátach.



Obr. 4.3: Vizualizácia normál pravidelnej kocky

4.5 Vizualizácia

4.5.1 Analýza

Dôležitým prvkom nášho nástroja je modul vizualizácie, ktorý bude slúžiť na zobrazovanie vstupného oblaku bodov a najmä na prezentáciu výsledných zrekonštruovaných objektov používateľovi.

Vo fáze vývoja aplikácie zároveň dovoľuje vývojárom priebežnú kontrolu funkčnosti jednotlivých rekonštrukčných metód. Môžu tu napríklad jednoducho sledovať, či sa geometrický útvar prekladá bodmi oblaku podľa očakávaní.

Z toho vyplýva požiadavka, aby bolo možné zobrazovať dáta vo forme oblaku bodov ako aj mriežky polygónov.

Úlohou tohto modulu však nebude len staticky vizualizovať, ale aj umožniť interaktívny pohyb scénou, vďaka ktorému bude možné detailne sledovať objekty z ľubovôlej uhlu pohľadu.

Možným rozšírením do budúcnosti bude začlenenie podpory pre 3D myš SpaceNavigator, ktorá nám bola poskytnutá laboratóriom počítačového videnia a grafiky na FIIT STU.

4.5.2 Návrh

Vzhľadom na skutočnosť že pre vývoj nášho rekonštrukčného nástroja používame knižnicu PCL, pre lepšiu integráciu sme sa aspoň pre začiatok rozhodli aj na vizualizáciu využiť vizualizačnú triedu `PCLVisualizer`, ktorá je súčasťou tejto knižnice. Návrh nášho modulu vizualizácie preto zodpovedá návrhu tejto existujúcej triedy. Trieda `PCLVisualizer` je veľmi komplexná a okrem mnohých iných nám poskytuje metódy pre začleňovanie modelov vo forme oblakov bodov alebo mriežky polygónov do scény. O interakciu používateľa so scénou sa vo východzom stave stará trieda `PCLVisualizerInteractorStyle`, ktorá definuje priradenie udalostí zo vstupných zariadení (napr. kliknutie a pohyb myšou) konkrétnym akciám, napr. rotácii kamery.

Práve spôsob interakcie definovaný touto triedou sme vyhodnotili ako nevyhovujúci a preto sa zameriavame na jeho vylepšenie. Medzi hlavné nedostatky patria:

- pohyb kamery po scéne nemožno ovládať klávesnicou, ale iba stlačením kolečka myši a súčasným pohybovaním myšou
- pre rotáciu kamery je potrebné držať stlačené ľavé tlačidlo myši
- rotácia kamery prebieha neprirodzene okolo nejakého bodu umiestneného pred kamerou, navyše vzdialenosť od neho je navyše premenlivá
- počas rotovania sa kamera nekontrolovateľne točí aj okolo osi Z, čo je mätúce.

Navrhnutým a realizovaným riešením je vytvorenie vlastnej triedy interakcie `CustomInteractorStyle`, ktorá bude odvodená od spomínanej `PCLVisualizerInteractorStyle`. V nej budú všetky nevyhovujúce metódy prekonané vlastnou implementáciou. Tým pádom sa zachová štruktúra metód pôvodnej triedy a zmení sa iba ich implementácia.

Pre uplatnenie tohto vylepšeného štýlu interakcie sa výsledná trieda `CustomInteractorStyle` odošle ako parameter konštruktora triedy `PCLVisualizer`.

4.5.3 Implementácia

Nastavenie rotácie kamery okolo jej stredu V knižnici PCL je kamera reprezentovaná triedou `Camera` a jej poloha v scéne je zadefinovaná jej členskou premennou `pos` (`position`), ktorá je polom troch súradníc x,y,z . Bod otáčania kamery je určený zase premennou `focal` (`focal point`), ktorá je podobne polom súradníc. Aby sme dosiahli efekt otáčania kamery okolo vlastnej osi, bolo potrebné pomocou metódy `SetFocalPoint` nastaviť bod otáčania `focal` presne do bodu polohy kamery `pos`. To však ale nebolo možné, pretože z týchto dvoch bodov sa určuje vektor pohľadu kamery a keby boli obidva

tieto body na totožných súradniciach, vektor pohľadu by bol nulový. Preto bol radšej bod otáčania nastavený do veľmi tesnej vzdialenosti pod pozíciu kamery, čím sa dosiahol rovnaký efekt otáčania ako požadovaný a aj vektor pohľadu zostal nenulový.

Uzamknutie rotácia kamery okolo osi Z Pre uzamknutie rotácie kamery okolo osi Z bolo potrebné prekonať metódu `onMouseMove()`. Jej telo bolo jednoducho nahradené rovnakou metódou z triedy `vtkInteractorStyleTerrain` knižnice VTK (Visualization ToolKit), ktorá presne implementuje požadované uzamknutie rotácie okolo osi Z.

Ovládane pohybu kamery klávesnicou Na základe konvencie ovládania pohybov v počítačových hrách sme sa rozhodli, že pohyb kamery vpred a vzad sa bude ovládať klávesami W a S, a pohyby dolava doprava klávesami A a D. Aby bolo možné tieto pohyby kamery zrealizovať, bolo potrebné prekonať metódu `onKeyPressed()`, kde bolo pre každý zo znakov W,S,A,D zadané volanie zodpovedajúcej funkcie vykonávajúcej požadovaný pohyb – konkrétne sú to funkcie `moveForwards()`, `moveBackwards()`, `keyboardPanLeft()` a `keyboardPanRight()`. Samozrejme tieto funkcie sme najprv naimplementovali. Metódy `moveForwards()` a `moveBackwards()` sú založené na premiestňovaní polohy kamery a jej stredu otáčania v smere rovnobežnom s vektorom pohľadu o vzdialenosť rovnú dĺžke tohto vektoru. Metódy pre pohyb do strán sú inšpirované metódami na posun kamery pomocou myši s tým rozdielom, že chýbajúci vektor pohybu myši do strany je nahradený konštantnou hodnotou, ktorá sa vygeneruje pri každom stlačení tlačidla A a D.

Literatúra

- [1] Thomas Holzmann, Christof Hoppe, Stefan Kluckner, and Horst Bischof. Geometric abstraction from noisy image-based 3d reconstructions. *OAGM Workshop 2014*, pages 1–8, 2014.