SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Tímový projekt

Rekonštrukcia 3D scény

2015/2016

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: 14. december 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

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 tried rekonštrukcie systému - metód - 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
  - Implementácia ovládania kamery vizualizéra
  - 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, prepojenie jednotlivých lib modulov projektu s exe

Sprint 4: Integrácia OpenCV, spracovanie a vizualizácia histogramov normál, segmentovanie rovín - primitív, skumanie meód na detekciu rezov v 2D, aktualizácia GUI

Sprint 5: Implementácia spracovania VTK súborov, doladenie pohybov kamery, zistenie dominantných oblatí v histograme, implementácia SDK pre ovládanie pomocou 3D myši, aktualizácia požiadaviek pre GUI

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. Okrem toho predpokladáme jej ďalšie využitie v projektoch vedených na FIIT, či už tímových tak aj diplomových.

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. Dostupné a vhodné technológie sa ukázali knižnice PCL „Point Cloud Library“ a knižnica OpenCV. Časom sa ukázalo, že pre prácu v 3D je knižnica PCL viac než dostačujúca. 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, pripravenej pre ďalšiu modularizáciu a možné rozšírenie v budúcnosti.

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 implementácií ďalšieho funkčného rozšírenia. Tým, že na vývoji pracuje súčasne 7 ľudí a ich súčasti musia byť systematicky prepájané, je potrebné dbať na určité zásady.

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

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 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 navrhnúť 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 funkcionalita je rekonštrukcia 3D dát a prípadne segmentácia objektov. Pre tento výstup je nežiaduce, keby bola táto knižnica viazaná na tematicky odlišnú funkcionalitu – gui, prípadne vizualizáciu, prípadne ovládač. Preto je potrebné navrhnúť 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 neplytvať pamäťou a zdieľať čo najväčšie množstvo dát, ktoré je možné.

3.1.2 Modularizácia systému

Z architektonického hľadiska nezávislosti modulov vyplynulo rozdelenie problémov zatiaľ na 4 hlavné moduly. Z hľadiska plánovania a rozdelenia projektu časť týchto modulov predstavuje Epic úlohy evidované v TFS backlog-u.

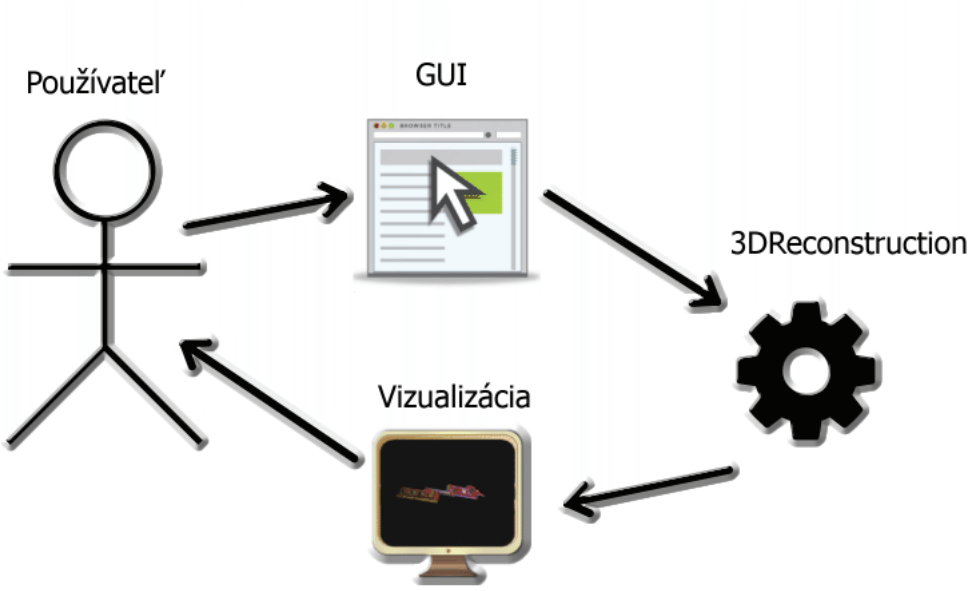
Tieto moduly sú:

- 3DReconstruction - hlavná logika aplikácie, dátové typy poznajú štruktúru jednotlivých načítaných dát
- GUI - Grafické rozhranie pre komunikáciu s používateľom
- Vizualization - Vizuálny výstup riešenia ako spätná informácia pre používateľa (Okrem vizualizácie rieši aj interakciu)
- DataHandling - Správa súborov a načítanie/konverzia dát

Diagram toku dát:

3DReconstruction

Tento modul obsahuje hlavnú naším 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. Okrem správy dát, spravuje spracované objekty. Významnou črtou



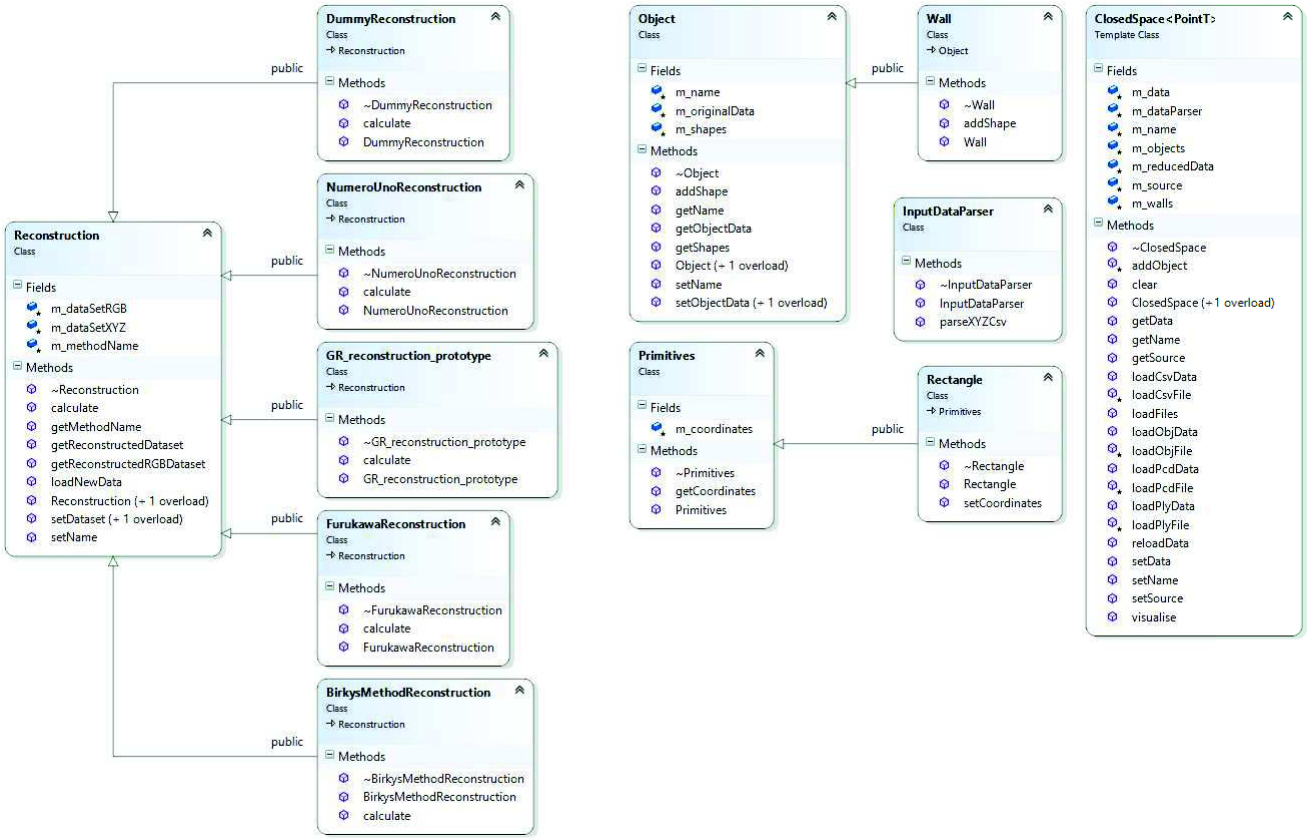
Obr. 3.1: Diagram toku dát

je, že pozná svoje typy objektov a taktiež ich dokáže z hľadiska PCL a VTK vizualizéra zobraziť podľa pravidiel, ktoré nim prislúchajú.

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.

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.



Obr. 3.2: Diagram tried

Opis ovládania a funkcionality GUI

Obrázok 3.3 zobrazuje GUI rozhranie projektu. Obsiahnutú funkcionalitu popisujú jednotlivé body.

- open file - slúži na výber jedného alebo viacerých súborov pre načítanie
- list box vyber metody - umožňuje vybrať jednu z metód pre použitie na spracovanie zvolených a načítaných súborov
- supinka radio buttonov
  voľba zobrazeni - je určená pre výber jedného typu požadovaného point cloudu pre zobrazenie
  original data - pre zobrazenie celého, pôvodného, neupraveného point cloudu
  reconstructed data - body pre rekoštrukciu plôch
  reconstructed shapes - zrekonštruované plochy
  remaining cloud - pre zobrazenie zostatkovej/nespracovanej časti spracovaného point cloudu
- OK button - spustí vizualizér, podmienkou je mať zvolené vstupné súbory, metódu a voľbu zobrazenia
- stop button - zastaví a zatvorí vizualizáciu, následne je možnosť ďalej interagovať s GUI, načítavať iné súbory, zvoliť si iné metódy či volby zobrazní a zase spustiť vizualizáciu s inými nastaveniami

Obr. 3.3: Screen GUI

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. Momentálna implementácia modulu rieši najmä interakciu používateľa so zobrazovanou scénou a pohyb kamery. Preto tiež aj z hľadiska čiastočnej tematickej odlišnosti bol tento modul oddelený od GUI.

DataHandling

Modul, ktorý vznikol z dôvodu lepšej modularizácie a lepších kompilačných časov, vzhľadom na oddelenie niektorých málo menených tried – kódu, ktorý sa od jeho vytvorenia už nemení. Takto sa oddelila funkcionalita načítavania dát z triedy „ClosedSpace“ do osobitného modulu. Čo sa hneď využilo, keď prišli nové dáta – aj nový formát dát, tým pádom vďaka flexibilite bolo pridanie možnosti načítavania bezproblémové a na vyšších vrstvách neboli potrebné žiadne úpravy.

Kapitola 4

Rekonštrukčné triedy 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 abstakcie a geometrizáciou. Takéto 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 horizonrálne 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 označkujeme pixely ako v rámci objektu alebo mimo objektu. Ohraničíme tie, ktoré do objektu patria a získame 2D plán jedného rezu. Obrysy 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 je zatiaľ v štádiu, kedy sa podarilo vytvoriť metódu na určenie normál všetkých bodov, nájdenie bodov s normálou podobnou osi z a následné premietnutie týchto bodov na os z. Ďalej sme implementovali metódu meanshift, pomocou ktorej určujeme umiestnenie horizontálnych rezov modelom.

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. "outlier" body, tieto pravdepodobne vymažeme z oblaku, resp. táto otázka bude riešená v neskoršej fáze projektu.

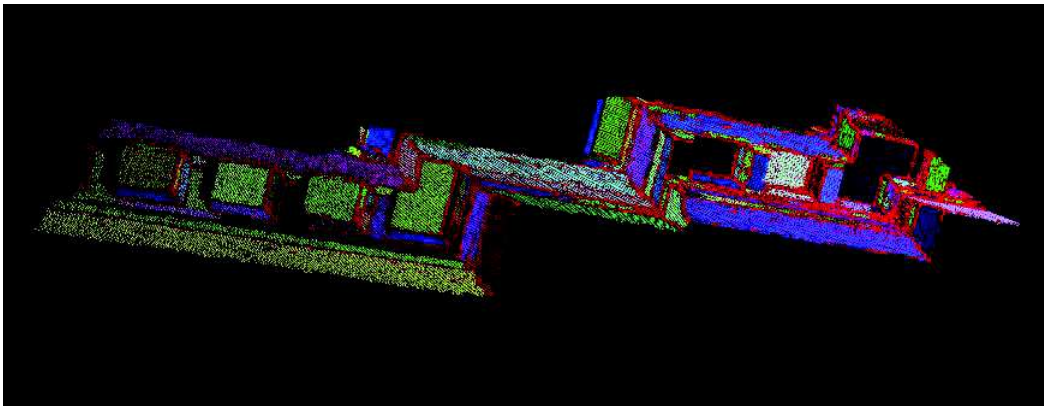
Mnohé výpočty potrebné pre algoritmus sú priamo zahrnuté v PCL knižnici, ktorú používame pri riešení projektu, a preto nemusíme riešiť implementácie elementárnych funkcií výpočtov. Napríklad v tomto prípade sú to normály, prípadne výpočet RGB hodnoty z bodov a podobne.

4.2.2 Návrh

Metóda je implementovaná pomocou triedy GR_reconstruction_prototype, ktorá reprezentuje rozšírenie hlavnej rekonštrukčnej triedy 3Dreconstruction. Metóda dostáva na vstupe oblak dát, ktorý je sekvenčne spracovávaný algoritmom growing region a na výstupe poskytuje zrekonštruovaný oblak dát pomocou tohto algoritmu.

4.2.3 Implementácia

V súčasnosti je vytvorený "prototyp" algoritmu, ktorý segmentuje vstupný oblak bodov na jednotlivé oblasti(plochy) bodov do klastrov. Každý takto vytvorený klaster obsahuje body označené jednou príslušnou farbou. Vo vysegmentovanom oblaku nájdeme aj body nespadajúce pod žiaden klaster, na obrázku 4.1 sú to červené body.



Obr. 4.1: Metóda growing region

Každý z vysegmentovaných klastrov predstavuje jednu plochu, zatiaľ však iba vo forme bodov. Kľúčovým krokom je tieto klastre reprezentovať čo najväčšími celistvými geometrickými útvarmi. Na tento účel sme využili metódu RANSAC, ktorá je dostupná v knižnici PCL. Tá zo vstupného klastru bodov na základe zadanej prahovej hodnoty odstráni "outlier" body a vráti koeficienty parametrického vyjadrenia roviny, v ktorej tieto body ležia. Tým

pádom sme však získali iba množinu neohraničených rovín. Preto je potrebné určiť ich ohraničenia. To sme dosiahli aplikovaním funkcie planeWithPlaneIntersection knižnice PCL, ktorou sme získali priesečníky jednotlivých plôch vo forme priamok (resp. koeficientov v ich parametrickom vyjadrení).

Ďalšia práca:

- hľadanie susednosti klastrov, aby sme vedeli ktoré plochy má zmysel vzájomne pretínať
- vytvorenie obdĺžnikov z rovín a ich pridanie pomocou addWall do zrekonštruovaného modelu

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 prispôsobiť, 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 prispôsobovať 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 miestnostiách s oblúkovitými stenami. Ďalšími problémamy 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.

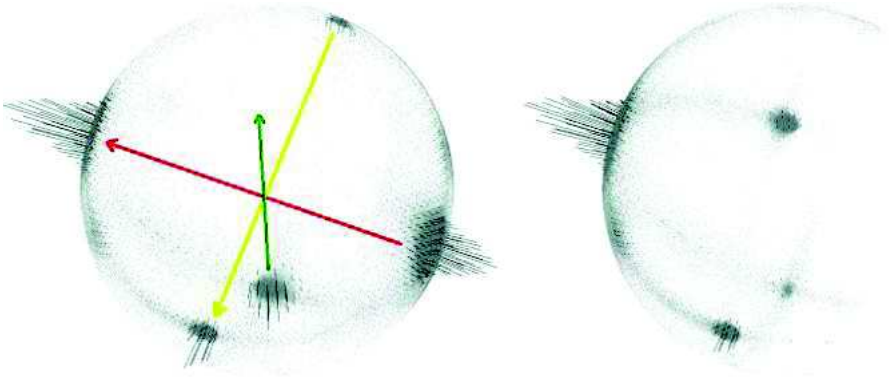
V našom projekte používame knižnicu PCL(Point Cloud Library) a OpenCV, ktoré sa snažíme maximálne využiť pre uľahčenie našej práce. PCL umožňuje výpočet normál bodov. Normály jednotlivých bodov vypočíta pomocou okolitých bodov zo zadefinovaného okolia. Vypočítané normály bodov obsahujú normalizovaný 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 navrhnúť 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. Pre vysegmentovanie bodov prislúchajúcich jednotlivým stenám z množiny bodov prislúchajúcih rovnobežným stenám by sme mohli použiť algoritmus growing-region. Po vysegmentovaní bodov prislúchajúcich k jednotlivým stenám sa môžeme sústrediť na vykreslenie jednotlivých stien. Cieľom je nájsť hlavné body z množiny bodov patriacich jednej steny, pomocou ktorých sa následne vytvorí primitíva steny. Tento problém sa rieši aj v metóde č. 2.

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 ukladávať všetky tri hodnoty normál (x,y,z) sme si zjednošuli úlohu výpočtom dvoch uhlou, 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 pripáde 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]

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 zobraziť ako obrázok. Pre nájdenie dominantných oblastí v histograme normál sme navrhli aj dve metódy. Prvá metóda bola založená na algoritme mean-shift a druhá na hľadaní maxima a následného odstraňovania okolia maxima. Pri mean-shift algoritmu je v našom prípade problém určiť začiatočnú pozíciu okien kvôli čomu nie je táto metóda vhodná pre náš problém. Naopak, druhá metóda vyhovuje našim podmienkam a je schopná nájsť potrebný počet dominantných oblastí v histograme.

Po získaní dominantných smerov oblaku bodov, pridelíme každý bod k určitému dominantnému smeru, ku ktorému sa normála bodu podobá čo najviac. 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

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

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 vyriešime pomocou algoritmu growing-reqion využijúc znalosť, že rovnobežné steny nie sú spojené alebo natoľko blízko seba, aby ich jednotlivé body nebolo možné priestorovo rozlíšiť. Je potrebné vhodne definovať susedné body (maximálnu vzdialenosť).

A na záver, po pridelení všetkých bodov k jednotlivým rovinám (stenám) bude potrebné vytvoriť primitíva pre jednotlivé steny. Toto chceme docieliť pomocou zredukovania bodov v oblaku bodov jednotlivých stien a následného vykreslenia plochy. Umiestnenie vykreslenej plochy bude potrebné zkorigovať podľa aritmetického priemeru alebo mediánu bodov v oblaku bodov. S týmto problém sa zaoberá aj metóda č.2.

4.4.3 Implementácia

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

Aktuálne metóda podporuje len výpočet normál, vytvorenie histogramu normál, nájdenie dominantných smerov z histogramu normál a označenie jednotlivých bodov podľa dominantných smerov.

4.4.4 Testovanie

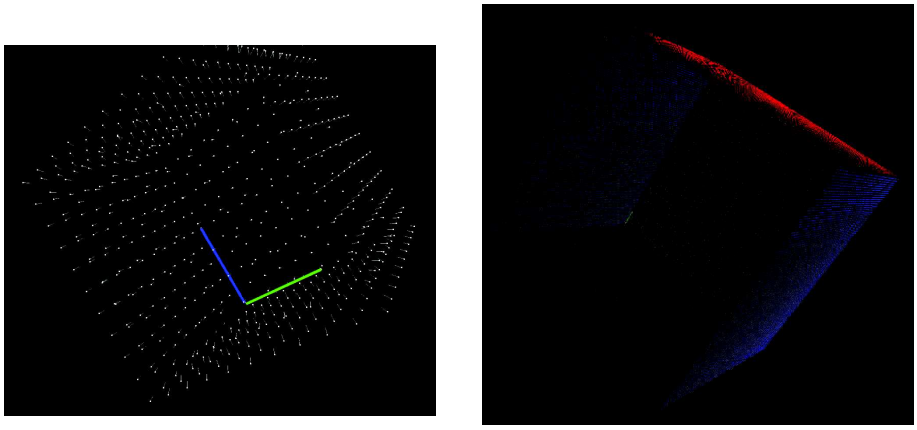
Testovanie prebieha súbežne pre všetky kroky metódy. Prevažne sa jedná o manuálne vizuálne testovanie pomocou vizualizácie knižnice PCL a OpenCV. Pre testovanie jednotlivých krokov metódy používame rôzne vstupné dáta. Pre testovanie sme si vytvorili tiež syntetické dáta. Vytvorili sme oblak bodov pravidelnej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

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.



(a) Výpočet normál

(b) Označenie bodov podľa dominantných smerov

Obr. 4.3: Vizualizácia pravidelnej kocky

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 ľubovolného 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 poľom troch súradníc x, y a z. Bod otáčania kamery je určený zase premennou focal (focal point), ktorá je podobne poľom 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 od pozície kamery, čím sa dosiahol rovnaký efekt otáčania ako požadovaný a aj vektor pohľadu zostal nenulový.

Uzamknutie rotácie 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 (Visualuzation ToolKit), ktorá presne implementuje požadované uzamknutie rotácie okolo osi Z.

Ovládanie 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 doľava 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 zadefinované 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.

Interakcia pomocou 3D myši 3Dconnexion SpaceNavigator
Knižnica PCL využíva na vizualizáciu a interakciu prostriedky knižnice VTK. Tá vo východzom stave podporuje interakciu iba pomocou štandardných vstupných zariadení. Našťastie existuje možnosť jej prekompilovania tak, aby podporovala prijímanie udalostí aj zo vstupných zariadení od výrobcu 3Dconnexion. Je však potrebné nainštalovať softvérovú vývojovú sadu pre 3D myš. Túto možnosť sme preto využili, čím sa nám sprístupnil vstup z 3D myši vo forme udalostí (napr. TdxButtonPressEvent) ktoré knižnica generuje pri manipulácii s myšou.

Knižnica VTK však neobsahuje žiadu interakčnú triedu, ktorá by tieto udalosti interpretovala na skutočný pohyb kamery v scéne. Pre každý typ udalosti generovaný myšou je teda nutné naimplementovať funkciu, ktorá na základe parametrov pohybu myšou vykoná adekvátny pohyb kamery v scéne. Následne stačí pomocou metódy *addObserver()* každú z týchto funkcií namapovať na príslušnú udalosť generovanú myšou.

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.

Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Tímový projekt

Rekonštrukcia 3D scény

Projektová dokumentácia - riadenie

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: 14. december 2015

Obsah

Kapitola 1

Úvod

V tomto dokumente sú opísané postupy, plány, a metódy riadenia tímového projektu stanovené naším tímom pre vývoj prototypu desktopovej aplikácie rekonštrukcie 3D scény z 3D dát. Charakter projektu je výskumne orientovaný, preto aj jeho hlavným cieľom je vytvoriť aplikáciu, ktorá bude spĺňať predpoklady testovacieho prototypu rôznych metód rekonštrukcie a segmentácie objektov. Projekt má predpoklad byť vývojovo a zdrojovo náročný, rovnako ako jeho prototyp náročný výpočtovo a pamäťovo. Preto je potrebné vytvoriť a dodržiavať predpísané pravidlá, aby sa možné komplikácie eliminovali skôr akoby mali skomplikovať ďalší vývoj. Vzhľadom na charakter projektu bolo potrebné upraviť aj metodiky riadenia, plánovanie vývoja a rozdelenie úloh na funkcionalite, aby na ich základe vznikol požadovaný produkt.

Projekt 3DRecon je vyvýjaný pre výskumné centrum Joanneum v Grazi, aj preto nemá charakter klasického produktu pre koncového používateľa, ale skôr výskumne orientovaného prototypu, ktorý môže slúžiť ako základná knižnica pre ďalší vývoj v oblasti rekonštrukcie či segmentácie 3D dát.

1.1 Prehľad dokumentu

V časti 2 sú opísané manažérske činnosti pridelené jednotlivým členom tímu aj s opisom prislúchajúcej zodpovednosti (2.1) a uvedený podiel na jednotlivých častiach dokumentácie k riadeniu aj inžinierskemu dielu (2.2). V časti 3 sú opísané realizované manažmenty v rámci nášho tímového projektu. Sumarizácie doterajších šprintov sa nachádzajú v kapitole 4. V kapitole 5 je prehľad používaných metodík s krátkym opisom, celé znenie jednotlivých metodík sa nachádza v časti 8. Na záver sú v kapitole 9 vložené exporty úloh z nástroja TFS na jednotlivé šprinty.

Kapitola 2

Role členov tímu a podiel práce

2.1 Manažérske činnosti

meno	rola	zodpovednosť
Lukáš Hudec	team leader, manažér architektúry	návrh a implementácia architektúry, správa tímu a rozdelenie úloh, evidencia úloh v TFS, segmentácia
Róbert Birkus	manažér vývoja	vývoj hlavnej aplikačnej logiky, správa implementovaných metód, starostlivosť nad dodržiavaním konvencií
Martin Jurík	projektový integrátor	prepojenie modulov, kontrola funkčnosti, implementácia modulov
Michal Löffler	biznis manažér	analýza požiadaviek, návrh a rozdelenie úloh, komunikácia so zadávateľom, analýza rizík
Róbert Karásek	test manažér	tvorba testov, vykonanie testov, dohľad nad "definition of done"
Michal Korbeľ	manažér propagácie	správa webovej stránky
Katarína Janečková	manažér dokumentácie	udržiavanie dokumentácie a zápisníc zo stretnutí

Tabuľka 2.1: Rozdelenie úloh v tíme

2.2 Podiel práce

časť	**autor**
Úvod	Katarína Janečková
Globálne ciele pre ZS	Martin Jurík
Celkový pohľad na systém	Lukáš Hudec
Rekonštrukčné triedy systému - úvod	Katarína Janečková
Metóda 1	Katarína Janečková
Metóda 2	Michal Korbeľ
Metóda 3	Róbert Karásek
Metóda 4	Róbert Birkus
Vizualizácia	Michal Löffler

Tabuľka 2.2: Podiel práce na dokumentácii k inžinierskemu dielu

časť	**autor**
Úvod	Lukáš Hudec
Role členov tímu a podiel práce	Katarína Janečková
Manažment komunikácie a ľudských zdrojov	Róbert Karásek
Manažment rozvrhu a rozsahu projektu	Lukáš Hudec
Manažment plánovania projektu	Róbert Karásek, Lukáš Hudec
Manažment kvality	Michal Löffler
Manažment rizík	Michal Löffler
Manažment integrácie a podpory vývoja	Martin Jurík
Sumarizácie šprintov	Michal Korbeľ
Používané metodiky	Katarína Janečková
Zoznam kompetencií tímu	Katarína Janečková
Metodika komunikácie	Róbert Karásek
Metodika pre evidenciu zmien zdrojového kódu	Róbert Karásek
Metodika vytvárania úloh v TFS	Lukáš Hudec
Metodika písania zdrojového kódu	Róbert Karásek
Metodika pre technickú dokumentáciu	Róbert Karásek
Export evidencie úloh	Katarína Janečková

Tabuľka 2.3: Podiel práce na dokumentácii k riadeniu

Kapitola 3

Aplikácie manažmentov

3.1 Manažment komunikácie a ľudských zdrojov

Oficiálne stretnutia tímu prebiehali každý týždeň vo štvrtok o 10:00. Ich témou bolo zhodnotenie vykonanej práce za predchádzajúci týždeň a naplánovanie ďalšej. Po stretnutí prebiehala diskusia o spoločnej implementácií, o dokumentácií, čo možno zlepšiť a ďalšie problémy, ktoré sa vyskytli počas týždňa. Prebiehali aj neformálne stretnutia, ktoré sa konali v škole alebo pomocou skype. Ich témou bolo hlavne riešenie menších problémov, ktoré vznikli mimo stretnutí a bolo ich potrebné vyriešiť čo najskôr.

3.1.1 Komunikačné nástroje

Pri výbere nástrojov sme vychádzali zo skúseností a preferencií jednotlivých členov tímu. Analyzovali sme najznámejšie nástroje a dohodli sme sa na nasledujúcich nástrojoch:

Slack

Je to aplikácia, ktorá slúži na komunikáciu a veľa používateľov si ju pochvaľuje aj vďaka zvýšenej produktivite. Dá sa používať aj ako samostatná aplikácia alebo vo webovom prehliadači V aplikácii sme si vytvorili komunikačné kanály pre potrebné témy. Ako príklad jeden kanál je pre GUI ďalší pre datasety a podobne. Umožňuje posielanie súborov, zdieľanie kódov ale aj notifikácie, čo je užitočné hlavne pri naliehavých úlohách. V prvých fázach sa využíval Facebook, nakoľko tam bol registrovaný každký člen tímu.

Nástroj pre zdieľanie obsahu Google Drive

Google Drive používame pre zdieľanie a zálohovanie dokumentov, nie len k dokumentácií ale aj k implementácii. Nachádzajú sa tu aj datasety, logá k našej webovej stránke a pod. Prístup k tomuto priečinku má každý aj vedúci tímu.

3.2 Manažment rozvrhu a rozsahu projektu

Pre správne rozvrhnutie funkčných úloh pre členov tímu je dôkladná príprava a podrobná analýza problému, ktorým sa náš tím zaoberá. Bližšie informácie k tomuto problému poskytla diskusia so zadávateľom, ktorého zastupoval vedúci nášho tímu. Po tomto stretnutí a analýze požiadaviek sa vytvoril zoznam cieľov, ktoré si náš tím rozdelil medzi prioritné a voliteľné, podľa času kedy je ich žiadúce dosiahnuť. Následne bolo potrebné vytvoriť metodiky pre kooperatívne riešenie požiadaviek zadávateľa. Výsledkom tejto analýzy sa stali postupy práce a vývoja. Na základe dodržiavania metodík a postupovaní podľa nami zvolených princípov sme vytvorili sadu štruktúrou elementárnych úloh, na ktorých pracovali vybraní členovia tímu počas dvojtýždňových šprintov. Pre transparentnosť práce na projekte a vnútornú organizáciu jednotlivých úloh náš tím používal TFS – systém na správu projektu.

Práca a rozdelenie úloh medzi členov tímu prebiehalo počas scrum stretnutí na začiatku každého šprintu. Na začiatku sa skontrolovali úlohy v backlogu, podľa progresu na poslednom šprinte a ďalších požiadaviek od zadávateľa sa zhodnotil význam a potreba niektorých úloh. Následne sa podľa analýzy zo stretnutia vytvorili ďalšie „používateľské príbehy“ a priradili sa k nim implementačné úlohy. Tieto príbehy boli následne ohodnocované všetkými členmi tímu, čím sa určila ich odhadovaná obťažnosť. Podľa tejto hodnoty bola jednotlivým úlohám priradená časová rezerva na ich riešenie. Vzhľadom na štruktúru problému a zameranie členov, každý člen dostal priradenú úlohu podľa oblasti, ktorú si vybral.

Po stretnutí sú nové a priradené úlohy a príbehy evidované v nástroji TFS podľa vopred spísanej metodiky vytvárania „taskov“ v TFS. Evidovanie progresu na jednotlivých úlohách prebiehalo tiež prostredníctvom TFS.

3.3 Manažment plánovania projektu

O manažment plánovania projektu sa v našom tíme staral team líder, manažér dokumentácie pričom ich hlavou úlohou v tejto oblasti bolo udržiavanie konvencií a dodržiavanie metodík tvorby úloh.

Keďže pre implementáciu riešenia bolo zvolené vývojové prostredie Visual Studio 2013 (pôvodne 2015, z dôvodu problémom s knižnicami sme prešli na stabilnejšiu verziu) s potrebnými nástrojmi a knižnicami, bolo pochopiteľné že pre správu projektu a jeho verzií, bol zvolený systém TFS. TFS poskytuje pripravenú štruktúru pre tvorbu úloh z hľadiska agilného vývoja, a tak sa po viacerých diskusiách vytvorili konvencie pre tvorbu a udržiavanie úloh v TFS. Vzhľadom na charakter projektu a výskumného zamerania zadávateľa úlohy bolo potrebné pristúpiť k určitým úpravám z hľadiska názvoslovia „Používateľských príbehov“. Každej úlohe na ktorej sa

začalo robiť bolo možné prideliť 3 stavy – „New“ nová, „Active“ aktívna, „Closed“ ukončená. Pre zistené chyby je to o stav „Resolved“ preriešený viac.

Konvencia tvorby, udržiavania a kategorizácie úloh bola dodržiavaná podľa metodiky vytvorenej pre tento účel. Takto vznikli vyššie štruktúry Epics a Features, ktoré predstavovali najväčšie funkčné prvky, ktoré požadoval zadávateľ vo výslednom produkte. Na týchto prvkoch sa postupne pracuje a vytvárajú sa nové podúlohy – „Používateľské príbehy“ (User Stories - UT), ktoré bližšie špecifikujú vlastnosti požadovaných Features. Do každého šprintu sa vybral set UT podľa počtu „story pointov“ a kapacity tímu. Rozdelenie bolo takmer vždy korektné až na ojedinelé prípady keď sa stalo, že sa museli preniesť nedokončené User stories do ďalšieho šprintu, prípadne že člen tímu skončil svoju prácu skôr a do konca šprintu nemal pridelený žiadnu ďalšiu úlohu.

Vedenie agile zasadnutí: Na začiatku každého stand-upu každý člen tímu prezentoval svoj progres za posledný týždeň a predpokladané smerovanie do ďalšieho týždňa. Táto časť bola vždy porovnávaná s agile boardom na TFS. Následne sa v otvorenej diskusií prebrali problémy s jednotlivými úlohami a prichádzalo sa ku konštruktívnym riešeniam. Podľa zistení a nových poznatkov sa podľa potreby vytvorili nové úlohy. Po konzultácií s vedúcim tímu sa navrhli nové „user stories“. Tým sa po ukončení analýzy priradila zložitosť pomocou agile pokrových kariet. Na koniec sa vybrali UT pre jednotlivých členov tímu podľa ich kapacity. Podľa určenej zložitosti sa UT rozdelili na ďalšie implementačné úlohy pre priradeného člena. Bližšia špecifikácia úloh bola vždy obsiahnutá v opise úlohy, takže sa o jej postate dozvedel každý člen tímu aj keď na nej momentálne nepracoval.

Súvisiaca metodika: Metodika vytvárania úloh

3.4 Manažment kvality

Pod manažmentom kvality v našom tíme rozumieme všetky činnosti ktoré vedú k optimalizácii pracovných činností a tým ku zvýšeniu kvality výsledného produktu. Patrí sem: uplatňovanie dohodnutých pracovných a komunikačných metodík, rozdelenie kompetencií jednotlivým členom tímu ako aj používanie kvalitných vývojových nástrojov a knižníc. Súčasťou manažmentu kvality je aj pravidelné neformálne vyhodnocovanie plnenia stanovených úloh a výknonnosti jednotlivých členov tímu, na základe ktorého prispôsobujeme úlohy členov za cieľom zefektívnenia práce. Asi najdôležitejšou činnosťou je však získavanie spätnej väzby od zadávateľa produktu a vyhodnocovanie jeho spokojnosti. Vzhľadom na špecifickú povahu nášho projektu kedy zadávateľom sú výskumníci z ústavu Joanneum Research v Rakúsku, pravidelné a priame stretnutia so zákazníkom nie sú možné. Napriek tomu pre lepšie vyhodnotenie priebežnej spokojnosti a získanie ďalších

požiadaviek sme sa dohodli na stretnutí so zástupcom z Joanneum Research na január roku 2016. Do tej doby sa musíme v ohľade požiadaviek vo veľkej miere spoliehať na vlastnú intuíciu a formálneho vlastníka produktu v osobe nášho pedagogického vedúceho p. Dr. Benešovej.

3.5 Manažment rizík

Jednou z hlavných činností ktoré sme vykonávali v počiatkoch projektu bola identifikácia rizík ktoré môžu ohroziť projekt a hľadanie vhodných protiopatrení. Každý z členov sa snažil identifikovať najmä riziká týkajúce sa jeho pridelených úloh a následne sme ich spoločne vyhodnotili a navrhli riešenia. Vybrané identifikované riziká:

Riziko: odchod člena tímu
Pravdepodobnosť: stredná
Preventívne opatrenie: rozdelenie kompetencií v tíme tak, aby boli do riešenia každého problému zapojení aspoň dvaja členovia tímu
Akcia na zmiernenie dopadu: pridelenie kompetencií odišlého člena jeho zástupcovi

Riziko: zadávateľ projektu nedodá testovacie dáta
Pravdepodobnosť: vysoká
Preventívne opatrenie: nájdenie dát na webe, vytvorenie syntetických dát
Akcia na zmiernenie dopadu: použitie záložných dát vytvorených v rámci prevencie

Riziko: nezvládneme dokončiť niektorý z dôležitých medzičlánkov, čo znemožní dalšiu prácu na projekte
Pravdepodobnosť: nízka až stredná
Preventívne opatrenie: plánovanie úloh a dôkladná analýza možností uplatnenia existujúcich komponentov od tretích strán
Akcia na zmiernenie dopadu: použitie existujúcich komponentov nájdených v rámci prevencie

Riziko: nedostupnosť niektorých kľúčových funkcií pre v knižnici PCL
Pravdepodobnosť: nízka
Preventívne opatrenie: prieskum možností iných knižníc pre spracovanie oblakov bodov, napríklad CGAL, MeshLab

Riziko: nedostatočná kvalita implementácie metód kvôli ich veľkému počtu
Pravdepodobnosť: stredná až vysoká
Preventívne opatrenie: sústredenie sa na 2-3 metódy, pričom na každej

pracujú aspoň dvaja členovia tímu

3.6 Manažment integrácie a podpory vývoja

Pre zabezpečenie konzistentnosti počas riadenia tímového projektu s pohľadu jednotlivých častí a zaistenie požadovanej celistvosti modulov systému máme stanovený postup integrácie a podpory vývoja nasledovne:

Jednotlivý členovia tímu sú zodpovedný za ucelené časti tímového projektu ktoré spracúvajú. Majú na starosti ich kvalitu vyhotovenia a tým padám aj výstup daného modulu a celkové spracovanie. Zodpovedný členovia komunikujú spoločne a aj s integrátorom projektu a podpory vývoja. Tým je dosiahnutá potrebná transparentnosť a lepšia spolupráca. Ďalej integrátor zozbiera od jednotlivých členov ich vytvorené moduly a vytvorí konečnú podobu procesu.

Pre prehľadnú spoluprácu tvorby softvéru, manažovanie práce, úloh a zdieľanie zdrojového kódu sme sa rozhodli použiť TFS - Visual Studio Team Foundation Server 2015, ktorý je jednoducho prepojiteľný a synchronizovateľný s Microsoft Visual Studio. Vďaka tomu dokážeme dodržiavať dohodnuté postupy a konvencie, udržiavať kód prehľadný a zároveň všetkým dostupný. Poskytuje jednoduché aktualizovanie zmien v zdrojovom kóde od všetkých členov tímu.

Kvôli potrebe vytvoriť prehľadnú komunikáciu medzi všetkými členmi tímu sme si zvolili aplikáciu Slack, ktorá Umožňuje triediť si jednotlivé problémy do oddelených konverzácii aby sa medzi sebou nemiešali a taktiež ponúka rôzne možnosti citovania, vytvárania útržkov zdrojového kódu či možnosť nainštalovania na smart zariadenia.

Kapitola 4

Sumarizácie šprintov

4.1 1. šprint

Projekt je postavený na rekonštrukcii 3D scény pomocou rozličných metód. V prvom šprinte sme sa preto sústredili na štúdium literatúry k daným metódam a štúdium pcl knižnice, keďže polovica tímu sa doposiaľ s ňou nestretla. Rovnako bolo potrebné vyriešiť dáta, ich načítavanie z rôznych formátov a najmä ich konverziu do jednej scény, keďže dáta boli tvorené len snímkami z rôznych pohľadov. V neposlednom rade sa riešila vizualizácia vstupných a výstupných dát, rovnako formou štúdia a malej implementácie v podobe pcl vizualizéra.

Po 1. šprinte sme odsúhlasili komunikáciu prostredníctvom komunikačného nástroja Slack. Doposiaľ sme používali na komunikáciu v tíme Facebook chat, avšak väčšina v tíme mala výhrady kvôli tomuto spôsobu. Manažment vytvárania a spravovania úloh v TFS zostal nezmenený, t.j. každý člen si vytvoril svoju vlastnú user-story na ďalší šprint a v nej definoval úlohy.

4.2 2. šprint

V tomto šprinte bola riešená prioritne architektúra celého projektu. Podarilo sa ju úspešne navrhnúť podľa MVC a implementovať. Okrem architektúry jeden člen tímu riešil ešte problém z 1. Šprintu ohľadne konverzie dát z formátu LAS. Úspešne sa mu podarilo prekonvertovať tento formát do PCD. Vytvorili sme grafické rozhranie, pomocou ktorého budeme interagovať s programom. Naďalej sme pokračovali v štúdiu literatúry ohľadne jednotlivých metód, z implementačnej časti sa nám podarilo spraviť výpočet normál a histogram normál. Napokon sme upravili dizajn webového sídla podľa pripomienok vedúcej tímu.

Po konzultáciách na predmete MIS a po jednotnom odsúhlasení sme od nasledovného šprintu, teda šprintu č. 3, zmenili manažment spravovania

úloh projektu v TFS. Doteraz každý člen tímu si pred šprintom vytváral storie na dva týždne a v rámci nej si zariadil úlohy. Všetky požadované úlohy boli rozložené na cca 16 hodín, aby ich stihol splniť. Tento manažment nebol správnym riešením, pretože ako tím, sme jednak ešte dobre nevedeli ohodnocovať jednotlivé úlohy (ohodnotili sme úlohu vysokým číslom a úloha bola ľahká) a na strane druhej tým, že sme mali presne vytvorený počet user-stories na každého člena (člen po dokončení svojej práce nemôže zobrať inú user-story a úlohy v nej zahrnuté a riešiť ich).

Nový manažment správy úloh v TFS poskytne lepší prehľad vykonanej práce a práce, ktorá nás ešte len čaká. Vytvorili sme jednotlivý „epics“, „features“, ktoré predstavujú veľké úlohy a budú otvorené počas celej fázy projektu a tie ďalej členíme do „user-stories“ a malých úloh, ktoré sa riešia počas daných šprintov. Rovnako sme zaviedli vytváranie úloh v rámci každej user-story, ktoré budeme vedieť presne ohodnotiť, či sú vykonané alebo nie, teda tzv. „definition done“ úlohy.

4.3 3. šprint

Tento šprint sme robili už „po novom“. Implementovali sme prototyp growing region metódy, ktorá je jednou zo segmentačných metód. Úspešne sme do vizualizéra pridali ovládanie scény pomocou kláves a rotovanie scény okolo kamery. Mnoho úloh je v tomto čase v rozpracovanom stave. V neposlednom prípade sme dokumentovali súčasný stav projektu. Retrospektíva po treťom šprinte bude vykonaná až na stretnutí tímu.

4.4 4. šprint

Šprint zahŕňal úpravy v architektúre, konkrétne v module grafického používateľského rozhrania, kde sme úspešne implementovali spúšťanie vizualizácie metód a ošetrili chybové stavy. Pre lepšiu interakciu s vizualizovanými metódami sme uvažovali použitie 3D myši SpaceNavigator, ku ktorej sme získali potrebné 3Dconnexion SDK. Podarilo sa nám implementovať ovládanie 3D myšou pomocou knižnice VTK, avšak do projektu táto funkcionalita zatiaľ nebola začlenená, nakoľko ten využíva PCL knižnicu. Napokon sme po spoločnej dohode rozhodli túto problematiku presunúť na letný semester, keďže sa momentálne zaoberáme vývojom jednotlivých metód, kde dáta nie sú natoľko obsiahle, žeby práca s nimi bola náročná na interakciu a prioritne sa musíme zamerať na doladenie architektúry a jej modulov. Pre overenie funkčnosti projektu sme riešili otázku testovania a vytvárania testov. Kvôli štúdiu vytvárania testov a pre mierne komplikácie zatiaľ neboli vytvorené žiadne testy. Okrem architektúry, modulu grafického používateľského rozhrania, problematiky vizualizácie a interakcie a testovania sme naďalej pokračovali v riešení (štúdiu, implementácii) rekonštrukčných metód. Pri

metóde growing region sme zisťovali rovinu z daných bodov v priestore a implementovali sme ju do projektu. Pri metóde č. 4 sme navrhli a implementovali označovanie bodov v oblaku dát a implementovali algoritmus pre nájdenie dominantných oblastí v oblaku dát. Pri metóde č. 1 sme vytvorili extrakciu rezov a implementovali mean shift. V metóde č. 3 sme neurobili žiadny pokrok, keďže osoba, ktorá má pridelenú túto metódu sa zaoberala štúdiom testov.

Migrácia vývojového prostredia Microsoft Visual Studio Community 2015 14.0.23107.0 (VS2015)-> Microsoft Visual Studio Professional 2013 12.0.21005.1(VS2013)

Nakoľko sa vyskytol problém pri kompilovaní knižníc OpenCV verzie 3.0.0 vo vývojovom prostredí VS2015 a navyše niektoré z knižníc OpenCV verzie 3.0.0 si vyžadovali podporu platformy CUDA a zároveň táto ešte nepodporuje VS2015, rozhodli sme sa jednohlasne pre migráciu z VS2015 do VS2013, kde knižnica OpenCV verzie 3.0.0 má plnú podporu.

4.5 5. šprint

V šprinte sme sa zaoberali rozšírením modulu grafického používateľského rozhrania o výber jednotlivých metód rekonštrukcie pomocou checkboxov. Používateľ pri interakcii s aplikáciou má navyše možnosť vidieť pôvodné dáta až potom rekonštruované a pod. Pri implementácii vizualizácie pôdorysu miestnosti z dát sme zaznamenali chybu v projekte, ktorá bude riešená a v prípade vyriešenia odstránená. Pri rekonštrukčnej metóde č. 4 sme pokročili s implementáciou submetódy pre nájdenie dominantných oblastí v histograme pomocou maxím. V rekonštrukčnej metóde č. 2 growing region sme hľadali priesečníky stien, pričom sme našli koeficienty priamky. Implementácia zatiaľ nebola otestovaná pomocou vytvorených automatických testov. Tieto ešte neboli zautomatizované kvôli migračným problémom s vývojovým prostredím.

Technická dokumentácia - Doxygen

Pre potrebu ľahkého dokumentovania kódu projektu sme sa rozhodli zaviesť používanie nástroja Doxygen. Dokumentácia sa generuje na základe komentárov pri jednotlivých úsekoch kódu. Doxygen prepája časti kódu a referencie na dokumentáciu k týmto častiam kódu, čím sa stáva projekt prehľadnejší z dlhodobého hľadiska pre všetkých záujemcov o prácu s projektom.

Kapitola 5

Používané metodiky

názov	**opis**	**autor**
Metodika komunikácie	Pravidlá pre používanie aplikácie Slack na komunikáciu v tímoch. Vytváranie kanálov, odpovedanie na správy a pod.	Róber Karásek
Metodika pre evidenciu zmien zdrojového kódu	Návod na správnu evidenciu zmien a vetvenie kódu podľa dohody v tíme.	Róbert Karásek
Metodika evidencie úloh v TFS	Pravidlá pre vytváranie nových úloh v nástroji TFS. Ktoré polia treba vyplniť a ako, ako správne nastaviť všetky parametre.	Lukáš Hudec
Metodika písania zdrojového kódu	Konvencie pre písanie zdrojového kódu v C++ v rámci projektu.	Róbert Karásek
Metodika pre technickú dokumentáciu	Pravidlá písania technickej dokumentácie pomocou Doxygen	Róbert Karásek

Tabuľka 5.1: Zoznam používaných metodík s ich opisom

Kapitola 6

Globálna retrospektíva

Ako tím sme sa posunuli počas riešenia projektu v zimnom semestri dopredu vo viacerých oblastiach. Ako prvý problém sme riešili komunikáciu v time. Na komunikovanie v rámci tímu sme prestali používať komunikačný nástroj Facebook a začali sme komunikovať prostredníctvom nástroja Slack, čím sme oddelili komunikačný pracovný kanál od iných komunikačných kanálov (zábava, priatelia, ...), ktoré výrazne ovplvňovali úspešnosť dorozumievania v tíme pri riešení pracovných záležitostí.

Ďalším výrazným krokom vpred bola zmena manažmentu evidencie úloh v TFS (Team Foundation Server). Zmena nastala po 2. šprinte, dovtedy každý člen evidoval úlohy v rámci svojej user-story, ktorú si dopredu na stretnutí vytvoril. Časová náročnosť úloh bola rozložená na približne 16 hodín. Avšak po odpracovaní svojich úloh člen môže robiť na ďalších úlohách, v prípade, že má definované user-story a male úlohy. Týmto spôsobom sa to nedalo. Aj ako začiatočníci sme nevedeli dobre ohodnocovať úlohy (ako začiatočníci všeob. v oblasti rekonštrukcie objektov) a mohlo sa stať, že sme ohodnotili úlohu vysokým číslom a úloha bola ľahko riešiteľná (nikto neoponoval). Nový manažment predstavoval definovanie množiny veľkých úloh v podobe “epics”, “features”, ktoré budú otvorené počas celej fázy riešenia projektu a nebudú sa meniť. Tieto ďalej členíme do “user- stories”, čo predstavujú väčšie úlohy riešené v rámci šprintu. Snažili sme sa zvoliť čo najväčšiu množinu týchto úloh, aby každý člen si potom mohol v šprintoch vyberať riešiť danú user-story a definovať k nej malé úlohy. Rovnako sme zaviedli vytváranie úloh v rámci každej user-story, ktoré budeme vedieť presne ohodnotiť, či sú vykonané alebo nie, teda tzv. „definition of done“ úlohy.

V ďalšej fáze projektu sme narazili na problém pri kompilovaní knižníc OpenCV verzie 3.0.0 vo vývojovom prostredí VS2015 a navyše niektoré z knižníc OpenCV verzie 3.0.0 si vyžadovali podporu platformy CUDA a zároveň táto ešte nepodporuje VS2015, preto sme sa rozhodli jednohlasne pre migráciu z VS2015 do VS2013, kde knižnica OpenCV verzie 3.0.0 má plnú podporu.

Pre potrebu ľahkého dokumentovania kódu projektu sme sa rozhodli zaviesť používanie nástroja Doxygen. Dokumentácia sa generuje na základe komentárov pri jednotlivých úsekoch kódu. Doxygen prepája časti kódu a referencie na dokumentáciu k týmto častiam kódu, čím sa stáva projekt prehľadnejší z dlhodobého hľadiska pre všetkých záujemcov o prácu s projektom.

Pre správnosť kódu v poslednom období plánujeme zaviesť testovanie. Testovanie kódu by sa malo uskutočňovať prostredníctvom Google testov. Testovanie pravdepodobne vykonáme až budúci semester nakoľko doteraz neboli naimplementované žiadne väčšie funkčné celky.

Kapitola 7

Zoznam kompetencií tímu

Každý v tíme zohráva dôležitú rolu, hlavné oblasti zodpovednosti sme si rozdelili nasledovne:

- manažér architektúry - Lukáš Hudec
- manažér vývoja - Róbert Birkus
- projektový integrátor - Martin Jurík
- biznis manažér - Michal Löffler
- test manažér - Róbert Karásek
- manažér propagácie - Michal Korbeľ
- manažér dokumentácie - Katarína Janečková

Viac o úlohách a zodpovednostiach členov tímu sa dočítate v kapitole 2.1.

Kapitola 8

Metodiky

8.1 Metodika komunikácie

Komunikácia
Nami zvolený systém pre komunikáciu medzi členmi tímu je Slack.

- Možnosť vytvoriť kanál ma každý člen tímu
- Do komunikačného kanála treba písať vždy k danej téme, ak neviem kde patrí konkrétna otázka, treba sa opýtať v general chate alebo konkrétnej osoby, na ktorú je smerovaná.
- Používať slack anotácie (napr. vloženie kódu do ‘code‘)
- Slack kontrolovať denne

Zdieľané dokumenty
Nami zvolený systém pre zdieľanie dokumentov je Google Drive.

- Štruktúru treba zachovávať ako je, prípadne požiadať o zmenu vedúceho tímu.
- Je dovolené meniť dokumenty ak sa nájdu v nich nezrovnalosti alebo presunúť súbor do správneho priečinka
- Nahrávať súbory vždy načas

8.2 Metodika písania zdrojového kódu

Header Files

- **každý .cc súbor musí mať "pridružený" .h súbor**, okrem unit testov a malých .cc, ktoré obsahujú iba main()
- na konci .h
- **súbory, ktoré začleňujú text, ale nie sú hlavičkovými tak na konci .inc** (napr. ak sa používajú na viacerých miestach v kóde, alebo sú určené pre špecifickú platformu)
- každý by mal obsahovať "header guards"(nižšie), a mal by byť prepojený s ďalšími headermi
- ak je v headeri deklarovaná template alebo inline funkcia, tak definície týchto konštruktorov musia byť v každom .cc súbore, ktorý ich používa
- (As an exception, a function template that is explicitly instantiated for all relevant sets of template arguments, or that is a private member of a class, may be defined in the only .cc file that instantiates the template.)

The #define Guard

- každý header musí obsahovať #define guard
- formát <*PROJECT*>_<*PATH*>_<*FILE*>_H_
  - **napr. súbor v foo/src/bar/baz.h by mal vyzerať nasledovne**

```
#ifndef FOO_BAR_BAZ_H_
#define FOO_BAR_BAZ_H_
        ...
#endif  // FOO_BAR_BAZ_H_
```

Forward Declarations

- ako predísť zbytočným #includes
- Pros / Cons
- ak používam funkciu zadeklarovanú v headeri, vždy #include konkrétny header
- pri použití class template, pre istotu #include jeho header
- ak používam bežnú triedu, spoliehať sa na poprednú dekleráciu je OK ale treba vždy skontrolovať
- **nenahradzovať datové členy s ukazovateľmi len preto aby sa zabránilo #include**

Inline Functions

- **iba funkcie, ktoré majú malý počet riadkov (napr. 10 a menej)**
- kompilátor zavolá funkcie inline, teda neprejdu zvyčajným volacím mechanizmom
- Pros / Cons
- neodporúča sa aby inline funkcia obsahovala cykly alebo switch (može ak sa nikdy nebude vykonávať), nemali by byť ani rekurzívne

Function Parameter Ordering

- **najprv vstupy, potom výstupy** (aj keď pridáme nový vstup, tak na začiatok s ním)

Names and Order of Includes

- všetko od source zložky teda
  - google-awesome-project/src/base/logging.h → #include "base/logging.h"
- ak *dir/foo.cc* alebo *dir/foo_test.cc* ma za úlohu implementovať alebo testovať *dir2/foo2.h,* **poradie includov** bude nasledovné
  - **dir2/foo2.h.**
  - **C system files.**
  - **C++ system files.**
  - **Other libraries' .h files.**
  - **Your project's .h files.**
    - každá sekcia zoradená podľa abecedy

- **Príklad**
  - google-awesome-project/src/foo/internal/fooserver.cc
  - *#include "foo/server/fooserver.h"*

    #include <sys/types.h>
    #include <unistd.h>
    #include <hash_map>
    #include <vector>

    #include "base/basictypes.h"
    #include "base/commandlineflags.h"
    #include "foo/server/bar.h"

- **ak obsahuje podmienky, tak až nakoniec**
- *#include "foo/public/fooserver.h"*

  #include "base/port.h" // For LANG_CXX11.

  #ifdef LANG_CXX11
  #include <initializer_list>
  #endif // LANG_CXX11

Scoping

Namespaces

- nepomenované namespace je v .cc súboroch podporované
- ak chcem pomenovať tak v kombinácii s menom projektu a jeho path
- nepoužívať inline namespace
- (Namespaces subdivide the global scope into distinct, named scopes, and so are useful for preventing name collisions in the global scope.)
- Pros / Cons

- ```
  namespace {                    // This is in a .cc file.
  // The content of a namespace is not indented.
  //
  // This function is guaranteed not to generate a colliding symbol
  // with other symbols at link time, and is only visible to
  // callers in this .cc file.
  bool UpdateInternals(Frobber* f, int newval) {
    ...
  }

  }  // namespace
  ```

- ```
  // In the .h file
  namespace mynamespace {

  // All declarations are within the namespace scope.
  // Notice the lack of indentation.
  class MyClass {
   public:
    ...
    void Foo();
  };

  }  // namespace mynamespace
  ```

- ***// In the .cc file***
  namespace mynamespace {

  // Definition of functions is within scope of the namespace.
  void MyClass::Foo() {
  ...
  }

  } // namespace mynamespace

- *#include "a.h"*

  DEFINE_bool(someflag, false, "dummy flag");

  class C; // Forward declaration of class C in the global namespace.
  namespace a { class A; } // Forward declaration of a::A.

  namespace b {

  ...code for b... // Code goes against the left margin.

  } // namespace b

- **nič** nedeklarovať v namespace std ani popredne deklarované classy z standard lib

Nested Classes

- nerobiť vnorené triedy ak nie su súčasťou interface
- používať nonmember funkcie s namespace alebo static member funkcie namiesto globalnych funkcii

Local Variables

- int i;
  i = f();
- int j = g();

- vector<int> v;
  v.push_back(1);
  v.push_back(2);
- vector<int> v = {1, 2};
- ďalšie ukážky

Static and Global Variables

- **statické alebo globálne premenné typov tried** sú **zakázané**
- ďalšie pravidlá

Classes

- nikdy nevolať konštruktorom virtuálne funkcie
- C++ keyword **explicit** for **constructors callable with one argument**.
- používať delegovanie a dedičnosť konštruktorov, keď znižujú duplicitu kódu (viac)
- poradie **public: pred protected: pred private: , metódy pred dátovými členmi (premenné)**
- Typedefs and Enums
- Constants (static const data members)
- Constructors
- Destructor
- Methods, including static methods
- Data Members (except static const data members)

- ak ma funkcia viac ako 40 riadkov = rozmýšlať či sa nedá rozdeliť

General Naming Rules

premenné pomenovávať zrozumiteľne, slová oddeľovať podčiarkovníkom
int price_count_reader; // No abbreviation.
int num_errors; // "num" is a widespread convention.
int num_dns_connections; // Most people know what "DNS"

mená súborov taktiež s _ alebo -
my_useful_class.cc
my-useful-class.cc
myusefulclass.cc
myusefulclass_test.cc

mená metód, tried.. *CamelCase*
static Pool<UrlTableProperties> pool;*

globálne s prefixom g_ napríklad

funkcie = CamelCase

enum CamelCase

makrá VELKYM_PISMOM

komentáre ako sa dohodneme ale asi klasicky
/* nad metódami */
//v riadku

alebo všetko jedným štýlom

// TODO(kl@gmail.com): Use a "*" here for concatenation operator.
// TODO(Zeke) change this to use relations.

môže sa použiť meno, kto by mal fixnúť ale môže to samozrejme aj niekto iný

komentáre písať zrozumiteľne bez spelling chýb

max dĺžka riadka 80 písmen
nepoužívať taby :(iba space a 2x => nastaviť aby po stlačení tab spravil 2x space

```
bool retval = DoSomething(averyveryveryverylongargument1,
                                              argument2, argument3);
```
argumenty funkcií pri volaní posúvať vždy na úroveň zátvorky

```
if (...) {
  ...
  ...
  if (...) {
    DoSomething(
          argument1, argument2,  // 4 space indent
          argument3, argument4);
  }
```

PODMIENKY

```
if (condition) {  // no spaces inside parentheses
  ...  // 2 space indent.
} else if (...) {  // The else goes on the same line as the closing brace.
  ...
} else {
  ...
}

if (x == kFoo) return new Foo();
```

jedno riadkový for

```
for (int i = 0; i < kSomeNumber; ++i)
  printf("I love you\n");
```

alebo

```
for (int i = 0; i < kSomeNumber; ++i) {
  printf("I take it back\n");
}
```

-viac podmienok

```
if (this_one_thing > this_other_thing &&
    a_third_thing == a_fourth_thing &&
    yet_another && last_one) {
  ...
}
```

POINTRE

```
// These are fine, space preceding.
char *c;
const string &str;

// These are fine, space following.
char* c; // but remember to do "char* c, *d, *e, ...;"!
const string& str;
```

RETURN

```
return result;                  // No parentheses in the simple case.
// Parentheses OK to make a complex expression more readable.
return (some_long_condition &&
          another_condition);
```

```
// Good - directives at beginning of line
  if (lopsided_score) {
#if DISASTER_PENDING          // Correct -- Starts at beginning of line
    DropEverything();
# if NOTIFY                   // OK but not required -- Spaces after #
    NotifyClient();
# endif
#endif
    BackToNormal();
  }
```

NAMESPACE

```
namespace {

void foo() {  // Correct.  No extra indentation within namespace.
  ...
}

}  // namespace
```

každé namespace na novom riadku bez zač. posunu

OPERÁTORY

môže byť aj x*y ale aj x * y, tak najlepšie sa bude dohodnúť (asi s medzerami)

MINIMALIZOVAT vertikálny space

HLAVNE BYŤ KONZISTENTNÝ

Tipy

cpplint nájde chyby vo formátovaní a pod.
nepoužívať exceptrions RTTI
static_cast<>(). Do not use other cast formats like int y = (int)x; or int y = int(x);

Use prefix form (++i) of the increment and decrement operators with iterators and other template objects.

8.3 Metodika pre technickú dokumentáciu

Na vygenerovanie technickej dokumentácie sa používa Doxygen(verzia 1.8.10).

Inštalácia sa nachádza na http://www.stack.nl/ dimitri/doxygen/download.html.

- Každý autor zdrojového kódu je nútený komentovať a napísať hlavičku podľa `http://www.stack.nl/~dimitri/doxygen/manual/docblocks.html`
- Komentáre musia byť stručné a výstižné
- Treba komentovať len potrebné časti kódu
- Netreba spomínať znova zdrojový kód v komentári
- Nezakomentovávať zdrojový kód!!!
- Konfigurácia je vyobrazená na obrázkoch
- Už vygenerovanú dokumentáciu je potrebné ukladať na Drive, rozdelenú do priečinkov podľa formátov

8.4 Metodika evidencie úloh v TFS

Prístup k nástroju je možný cez portál tfs.fiit.stuba.sk, alebo s použitím pluginu vo Visual Studio 2015/2013 po presune do tohto vývojového prostredia. Postup vo vypĺňaní jednotlivých polí je rovnaký a výstup musí spĺňať metodikou zavedenú normu. Pre polia, ktoré obsahujú všetky udalosti platia rovnaké pravidlá.

Momentálne existujú 3 najvyššie celky - Epic, Feature a User Story. Epic a Feature sa vytvoria na naplánujú s vlastníkom produktu a predstavujú hlavné celky požiadaviek. Ďalej sa už len pridávajú User Story do Feature podľa zamerania požiadavky a úlohy s ňou spojenej. Pri vytváraní úlohy sa riadíme najvyšším celkom - User story, jednotlivé úlohy pre členov tímu sú označené ako Task a Bug

User story:

- definovaný používateľský príbeh, abstrahovaná menšia časť požiadavky vlastníka produktu - implementačná oblasť problému
- má svojho zodpovedného vedúceho - ten je pridelený na stretnutí tímu, podľa osobnej žiadosti alebo určený podľa skúseností

- ak nie je určený, user story vytvorí team leader a úlohu nastaví podľa dohodnutých kritérií, zodpovedný vedúci bude následne určený hlasovaním v tíme

Task:

- jednotlivé úlohy používateľského príbehu
- ak sú jasné hneď na stretnutí, vytvorí ich zodpovedný vedúci a pridelení členovia tímu si ich rozdelia podľa vlastného záujmu tak, aby ich stihli za šprint vypracovať

Bug:

- označenie pre úlohu zaoberajúcu sa chybou v programe alebo dokumentácií
- jej vytvorenie má na zodpovednosť ten člen, ktorý chybu odhalil, ale riešenie môže po konzultácií so zodpovedným vedúcim user story ponechať na iného člena - zodpovedného za oblasť, v ktorej bola chyba odhalená

Vypĺnanie polí úloh

Celá dokumentácia a správa projektu je písaná výhradne v slovenčine s použitím diakritiky okrem explicitných výnimiek vyznačených v metodike(pokiaľ nie sú zistené žiadne problémy).

- Meno
  - začiatočné písmeno veľkým, ostatné podľa pravidiel slovenského jazyka
  - stručný a výstižný
- Tagy
  - využitie všeobecne známych a výstižných skratiek
  - bez diakritiky
  - môže byť nastavených viac, podľa potreby
  - v prípade zmeny opísať všetky zmeny, ktoré nastali
- Detail, Descrpition
  - detailnejší popis úlohy, z ktorého bude každému jasné o čo v danej úlohe ide
  - spomenúť možné zmeny v úlohe
- Stav

  - keď je úloha vytvorená, je označená ako “New” (prednastavené, nemožno zmeniť)

- Pridadenie (Assigned to)
  - vyplnené podľa typu úlohy a stavu, v ktorom sa nachádza
- Priorita
  - defaultne nastavená na hodnotu 2
  - upravená podľa dohody na stretnutí a momentálnych požiadaviek
- Iterácia
  - nastavuje sa podľa toho do ktorého šprintu daná úloha patri - táto hodnota musí byť vyplnená korektne, inak úloha nie je správne zobrazená v tabuľke
- Effort - vypĺňajú sa 3 hodnoty - “Original estimate”, “Remaining” a “Completed”
  - Original estimate - odhadovaný čas, ktorý riešenie zaberie; nastavená hodnota od ktorej sa počíta burndown chart
  - Remaining - original estimate - completed (odpracovaný čas) = zostávajúci čas; podľa tejto hodnoty sa počíta pokles burndown chart-u
  - Completed - riešiteľ si zadá koľko času na riešení tejto úlohy strávil

Stav úlohy závisí od toho či je to “User story” alebo “Task”.
Hlavné stavy:

- New - nastavená pokiaľ je priamo vytvorená
- Active - keď riešiteľ začne pracovať na úlohe zmení jej stav na aktívnu “User stories”, na ktorých sa pracuje majú tento stav od začiatku riešenia aj keď sa práve nerieši žiadna z ich taskov
- Resolved - “User story”, ktorej riešenie je hotové a prešlo aj testami (task item tento stav nemá)
- Closed - ukončená úloha, na ktorej sa predpokladá už viac nebude pracovať

Šprinty
Šprinty sa vytvárajú s dvojtýždňovým trvaním, najčastejšie od agile stretnutia po najbližšie stretnutie. V prípade technických alebo problémov v manažmente je možné tieto termíny posunúť - najviac však o 2 dni. Ak

šprint obsahuje úlohy k doplneniu dokumentácie, tieto úlohy sa evidujú v jednej špeciálnej User Story, vzhľadom na to, že dokumentáciu potrebuje aj vlastník produktu, jej štandardné ohodnotenie je 5 SP.

Šprinty vytvára a spravuje biznis manažér, ktorý zastupuje scrum mastera, prípadne tím líder, ktorý má prehľad o dianí v tíme a progrese práce na jednotlivých úlohách.

Vytvorenie úloh Úlohy sa plánujú na agile zasadnutí tímu, na ktorom sa plánuje nový šprint. Na začiatku každého stretnutia sa vykoná stand-up, na ktorom každý člen tímu opíše svoj progres za posledný týždeň, prípadné problémy. Ak sa vyskytnú problémy, ostatní členovia tímu produktívne prispejú s pomocou k riešeniu daného problému.

V prípade stretnutia s plánovaním šprintu, po stand-up scrum master vedie plánovanie nových úloh na nasledujúci šprint. Ak vznikli nové požiadavky vlastníka produktu, najprv sa určí ich priorita a podľa jej výšky sa vyberajú požiadavky s backlogu v kombinácií s novými požiadavkami. Vybrané požiadavky sa vložia do backlogu šprintu a ostatné sa uložia do backlogu a evidujú sa k Epicom a Featurám podľa tematiky. Ak majú vzniknúť nové tím sa riadi vyššie spomenutým postupom vyplnenia úloh. Vybrané úlohy sa ohodnotia pokrom a prebehne analýza, ktorá má pomôcť zodpovednému riešiteľovi vytvoriť postup práce na úlohe - taktiež evidovať svoju prácu v TFS ako Task Items. Bližšie informácie k správe úloh podľa agilných metód sú v dokumente “Manažment plánovania projektu”.

Pokiaľ sú tieto úlohy/Task Items jasné počas stretnutia - zadáva ich do TFS team leader, zodpovední členovia tímu len vyplnia Description. V opačnom prípade si ich vytvárajú členovia postupne ako podľa vlastnej analýzy zistia, aké úlohy ich práca bude obnášať.

Zápisnice

Písanie zápisov zo stretnutia sa eviduje rovnako ako ostatné úlohy a ich zaradenie spadá do každého šprintu avšak bez priradenia k User Story. Zápisy majú svoju formu a predpokladaný obsah, čo je určené šablónou, ktorá sa nachádza v priečinku na Google Drive tímu.

- Názov úlohy je zadaný v tvare - “Zápisnica ‘RRRRMMDD’” (napr. Zápisnica 20151001)
- Description - obsahuje text zápisnice (stačí plain text)
- Attachements - formátovaná pdf verzia zápisnice

8.5 Metodika pre evidenciu zmien zdrojového kódu

Používa sa GIT integrovaný vo Visual Studiu.

- Commit musí obsahovať stručný opis, musí byť pri ňom autor.
- Commit je dovolený len pre skompilovateľnú verziu aby sa zbytočne neobmedzovali ostatní členovia tímu.
- Ohľadne verziovania je povolené použiť všetky možnosti ale zbytočne nekomplikovať verzie.
- Ak sú nezrovnalosti ohľadne commitov, treba kontaktovať integrátora

Kapitola 9

Export evidencie úloh

Tabuľka 9.1: Úlohy pred začatím šprintov

Title	Assigned to	Description	Tags
Zápisnica č. 1 (20150924)	Bc. Katarina Janeckova	Vytvorenie zápisnice z 1. stretnutia	doku
Štúdium literatúry [1]	Bc. Katarina Janeckova	Odporúčaná literatúra [1] k zadaniu témy projektu. Geometric Abstraction from Noisy Image-Based 3D Reconstructions	analyza
Štúdium možností OpenCV 3.0.0	Bc. Lukas Hudec	Zistenie možností knižnice OpenCV z hľadiska 3D Rekonštrukcie a 3D registrácie.	analyza
Metodiká tvorby úloh	Bc. Lukas Hudec	Spísanie metodík pre vytváranie úloh v TFS. Metodiky budú dostupné na google drive v priečinku tímového projektu. Predpokladajú sa ešte zmeny, po odsúhlasení ostatnými členmi tímu	
Analýza spracovania dát	Bc. Lukas Hudec	Analýza a navrhnutie rozhrania pre načítanie a správu datasetov - vstupov programu. Výsledok analýzy a návrh implementácie je priložený v prílohe.	analyza; dev
Zápisnica č. 2 (20151001)	Bc. Lukas Hudec	Vytvorenie zápisnice z 2. stretnutia	doku
Analyzovanie spracovania dat	Bc. Martin Jurik	Anaylzovanie spracovania dat, navrh hlaviciek tried	analyza; dev
Zápisnica č. 3 (20151008)	Bc. Martin Jurik		doku
Studium kniznice ITK	Bc. Martin Jurik	Štúdium možností knižnice ITK	ITK
Implementácia webu (HTML + CSS + JS)	Bc. Michal Korbel		
Štúdium metódy growing region	Bc. Michal Korbel	Analýza možností knižnice PCL, príp iných pre použitie metódy Growing region.	
Nahratie stránky na server	Bc. Michal Korbel	Nahratie všetkých súborov tvoriacich stránku do adresára /var/www/html na našom virtuálnom stroji team05-15.studenti.fiit.stuba.sk. Na nahrávanie je potrebné použiť protokol SFTP (port 22), pretože štandardný FTP port 21 nie je na stroji povolený.	
Zápisnica č. 7 (20151116)	Bc. Michal Korbel		
Zápisnica č. 6 (20151029)	Bc. Michal Loffler	Vytvorenie zápisnice zo 6. stretnutia.	doku
Štúdium možností vizualizácie	Bc. Michal Loffler	Prieskum dostupných knižníc na vizualizáciu dát vo forme oblaku bodov a zhrnutie do dokumentu.	
Inštalácia virtuálneho servera	Bc. Michal Loffler	1. Pripojenie cez VNC pomocou pridelených prihlasovacích údajov 2. Inštalácia, konfigurácia OS Ubuntu Server 14.04.3 (LAMP, sshd) 3. Vytvorenie používateľských kont pre R. Karáska, M. Korbeľa	
Analýza spracovania dát	Bc. Robert Birkus	Analýza a návrh rozhrania pre spracovania dát (Načítanie dát, konverzia dát a ukladanie dá	analyza; dev
Štúdium knižnice Point Cloud Library	Bc. Robert Birkus	Rozbehnutie PCL knižnice vo VS 2015, vyskúšanie features knižnice a zistenie dátových typ	analyza; PCL
Zápisnica č. 4 (20151015)	Bc. Robert Birkus	Vytvorenie zápisnice zo 4. stretnutia.	doku
Zápisnica č.5 (20151022)	Bc. Robert Karasek	Vytvorenie zápisnice z 5. stretnutia.	doku
C++ conventions	Bc. Robert Karasek	vytvorenie poznamok z dokumentu http://google-styleguide.googlecode.com/svn/trunk/cppguide.html	doku
Štúdium literatúry [2]	Bc. Robert Karasek		analyza

Tabuľka 9.2: Úlohy z prvého šprintu

Title	Assigned to	Iteration	Description	Tags
Analýza metódy horizontálnych rezov	Bc. Katarina Janeckova	\Sprint 1	Analyzovať: - spôsob určenia a extrakcie rezu - projekcia bodov s určitou normálou na z-os	analyza
Analýza metódy mean shift	Bc. Katarina Janeckova	\Sprint 1	Analyzovať metódu mean shift, spôsob jej využitia a implementáciu v knižniciach.	analyza
Načítanie súboru formátu PLY	Bc. Lukas Hudec	\Sprint 1	PointCloud data je možné ukladať v rôznych typoch súborov a od toho sa odvíja aj formát akým sú uložené. Jedným možným spôsobom je načítavať dáta z formátu PLY (polygón)	
Načítane súboru formátu OBJ	Bc. Lukas Hudec	\Sprint 1	Ďalším možným formátom pre uloženie dát je formát Obj (wawefront) knižnica pcl obsahuje funkcionalitu pre načítanie aj tohto formátu.	
Analýza dát a konfiguračných súborov	Bc. Lukas Hudec	\Sprint 1	určením pozície kamery v konfiguračnom súbore alebo hlavičke jednotlivých načítavaných súborov.	analyza
Implementácia načítania súborov	Bc. Lukas Hudec	\Sprint 1	Implementácia načítavacieho rozhrania s možnosťou načítať "multiple files" a automatické vyhľadanie konfiguračného súboru.	dev
Implementácia kombinácie snímok scény	Bc. Lukas Hudec	\Sprint 1	Kombinácia snímok do jednej karteziánskej sústavy podľa určenia pozície kamery z konfiguračného súboru, alebo metadát - hlavičky - súborov obsahujúce jednotlivé snímky	dev
Testovanie a doimplementácia pre rôzne vstupy	Bc. Lukas Hudec	\Sprint 1	Ako bolo spomenuté existujú rôzne typy dát a rôzne vstupy - všetky majú ale spoločnú podstatu - pozícia kamery a súradnice snímky - je potrebné preto docieliť aby sa problém s načítavaním týchto dát zovšeobecnil a výstup zjednotil.	dev; test
Nacitanie dat formatu PCD	Bc. Martin Jurik	\Sprint 1	Vyhladanie informacii a nacitanie dat formatu PCD, pokus o jednoduche zobrazenie a nastavenie zobrazenych bodov do kamery	dev
Vyhladanie informacii o formate LAS	Bc. Martin Jurik	\Sprint 1	Vyhladanie informacii o formate LAS, analyza struktury ulozenych dat, moznosti jeho nacitania a konverzie do pouzitelneho formatu	dev
Nacitanie dat vo formate LAS	Bc. Martin Jurik	\Sprint 1	Nacitanie point could-ov vo formate LAS a testovanie	dev
štúdium PCL funkcií vhodných pre GR, výpočet normál bodov pomocou PCL	Bc. Michal Korbel	\Sprint 1		
Vytvorenie hrubého prototypu modulu	Bc. Michal Korbel	\Sprint 1		
Štúdium PCLVisualizer	Bc. Michal Loffler	\Sprint 1	Štúdium triedy PCLVisualizer obsiahnutej v knižnici PCL.	
Vizualizácia vstupných dát	Bc. Michal Loffler	\Sprint 1	Štúdium možností vizualizácie vstupných - point cloud dát	
Vizualizácia výstupných dát	Bc. Michal Loffler	\Sprint 1	Štúdium možností vizualizácie výstupných - point cloud/mesh dát	
Štúdium formátu Autodesk DXF	Bc. Michal Loffler	\Sprint 1	Štúdium v praxi rozšíreného parametrického formátu Autodesk DXF (Drawing eXchange Format), ktorý sa používa na zakresľovanie plánov budov, príp. strojárskych a elektrotechnických výkresov.	
Implementácia intuitívnejšieho spôsobu ovládania pohybu kamery	Bc. Michal Loffler	\Sprint 1		
Analýza a hrubý návrh modulu histogram normál	Bc. Robert Birkus	\Sprint 1	Pre implementáciu histogramu je potrebné najprv zanalyzvať možné spôsoby výpočtu normál a následného návrhu štruktúry histogramu normál.	analyza; design; histogram; normal
Výpočet normál point cloud-u pomocou	Bc. Robert Birkus	\Sprint 1	Cieľom tejto úlohy je naimplementovať výpočet normál point cloudu pomocou PCL funkcií.	dev; normal; PCL
Vytvorenie histogramu normál	Bc. Robert Birkus	\Sprint 1	Cieľom je implementácia histogramu normál podľa vypočítaných normál z point cloudu	dev; histogram; normal
Zistenie smeru stien z point cloud-u pomocou histogramu	Bc. Robert Birkus	\Sprint 1	Cieľom tejto úlohy je pomocou dominantných oblastí v histograme normál zistiť smer stien v naskenovanej miestnosti (point cloud).	analyza; design; dev; histogram; normal
Analýza Data Collection and	Bc. Robert Karasek	\Sprint 1		
Analýza Inverse CSG	Bc. Robert Karasek	\Sprint 1		
Analýza Reconstructing 2D CSG Models	Bc. Robert Karasek	\Sprint 1		
Analýza Hough transformation	Bc. Robert Karasek	\Sprint 1		
Implementácia Hough transformation	Bc. Robert Karasek	\Sprint 1		
Zoznámenie sa s knižnicou a nájdenie vhodných funkcií pre Furukawu	Bc. Robert Karasek	\Sprint 1		

Tabuľka 9.3: Úlohy z druhého šprintu

Title	Assigned to	Iteration	Description	Tags
Práca s normálami	Bc. Katarina Janeckova	\Sprint 2	- prehľad metód v knižniciach - analýza a pochopenie - spolupráca s RB	analyza
Ortogonálna projekcia	Bc. Katarina Janeckova	\Sprint 2	- analýza a pochopenie - prehľad implementácie v knižniciach - edit: projekcia bodov na os z	analyza
Mean shift	Bc. Katarina Janeckova	\Sprint 2	- analýza a pochopenie metódy - prehľad o implementácii v knižniciach	analyza
Implementácia využitia normál	Bc. Katarina Janeckova	\Sprint 2	- implementácia metód pracujúcich s normálami	implementacia
Implementácia projekcie	Bc. Katarina Janeckova	\Sprint 2	Premietnutie bodov na os z.	implementacia
Implementácia metódy mean shift	Bc. Katarina Janeckova	\Sprint 2	Použiť metódy z PCL alebo OpenCV aplikujúce mean shift v našom projekte.	implementacia
Implementácia architektúry	Bc. Lukas Hudec	\Sprint 2	Podľa architektonického návrhu projektu z posledného zasadnutia timu je potrebné vytvoriť nový	dev; implementacia
Návrh architektúry	Bc. Lukas Hudec	\Sprint 2	ktoré budú potrebné pre oo prístup k riešeniu a implementácií aplikácie.	analyza; design; doku
Vytvorenie diagramov	Bc. Lukas Hudec	\Sprint 2	Pre čitateľnosť a vyjadrenie návrhu architektúry sú potrebné UML diagramy. Najvhodnejšie pre naše použitie budú pravdepodobne "class diagramy".	design; doku
Implementácia Base class	Bc. Lukas Hudec	\Sprint 2	Hlavná trieda, resp. "abstraktná trieda", ktorá určí štruktúru tried obsahujúcich metódy a bude slúžiť ako "interface" medzi načítaním dát a vizualizáciou spracovaných dát.	dev; implementacia
Vytvorenie testovacieho datasetu	Bc. Lukas Hudec	\Sprint 2	Existujúci a 100% funkčný a použiteľný dataset je príliš zložitý a rozsiahly, čo je pre potreby vývoja a testovania aplikácie zbytočne priveľa. Z tohto dôvodu je potrebné z tohto datasetu (apartmán) vybrať	dev; implementacia; test
Úprava triedy načítania a správy dát	Bc. Lukas Hudec	\Sprint 2	Upravenie triedy načítavania dát zo súborov. Momentálny stav je viac-menej procedurálneho rázu a pre priblíženie a zdokonalenie objektovo orientovaného prístupu je potrebné zmeniť ráz metód a štruktúry triedy. Táto trieda a jej inštancie budú obsahovať objekty metód 3D rekonštrukcie a pracovať nad dátami, ktoré	dev; implementacia
Doriesenie LAS	Bc. Martin Jurik	\Sprint 2		
Konverzia LAS->PCD	Bc. Martin Jurik	\Sprint 2		
Debugovanie pouzivanych libraries	Bc. Martin Jurik	\Sprint 2		
Vytvorenie prototypu Grafického rozhrania	Bc. Martin Jurik	\Sprint 2	Vytvorenie jednoduchého grafického rozhrania pre testovacie účely aplikácie. Načítanie súborov - multiselect Spustenie jednotlivých metód Prepínanie možností výstupu	
Analýza indexov	Bc. Michal Korbel	\Sprint 2		
Test example rozne PCD subory	Bc. Michal Korbel	\Sprint 2		
Test example rozne thresholds	Bc. Michal Korbel	\Sprint 2		
Komplet uprava webu	Bc. Michal Korbel	\Sprint 2		
Analýza možností knižnice CGAL	Bc. Michal Loffler	\Sprint 2	Zistiť aké sú výhody knižnice CGAL oproti PCL, nájsť komponenty využiteľné pre náš projekt	
Analýza možností manipulácie s kamerou	Bc. Michal Loffler	\Sprint 2	Problém: vo východzom stave sa v PCLVisualizeri kamera otáča okolo nejakého bodu pred kamerou (začiatok súradnicovej sústavy?), čo je trochu neprirodzené.	
Uzamknutie osy kamery	Bc. Michal Loffler	\Sprint 2	Problém: PCLVisualizer defaultne otáča kamerou v smere všetkých osí X,Y,Z Riešenie: uzamknutie rotovania kamery okolo osi Z	

Analýza princípov interakcie s	Bc. Michal Loffler	\Sprint 2	Štúdium tried InteractorStyle, ktoré definujú spôsob interpretácie stláčania kláves a hýbania myšou.	
Implementácia histogramu normál	Bc. Robert Birkus	\Sprint 2	Po analýze a návrhu je potrebné naimplementovať histogram normál.	dev; histogram; normal
Analýza štruktúry pcl::Normal	Bc. Robert Birkus	\Sprint 2	Cieľom je zanalyzovať štruktúru pcl::Normal, ako reprezentuje normály, čo všetko iné obsahuje.	analyza; normal
Analýza možností PCL pre	Bc. Robert Birkus	\Sprint 2	Pred samotnou implementáciou histogramu normál je potrebné analyzovať všetky možnosti knižnice	analyza; histogram; normal
Analýza výpočtu normál v PCL	Bc. Robert Birkus	\Sprint 2	Je potrebné analyzovať možnosti knižnice PCL pre výpočet normál.	analyza; normal; PCL
Návrh histrogramu normál	Bc. Robert Birkus	\Sprint 2	Je potrebné navrhnúť štruktúru histogramu normál.	design; histogram; normal
Analýza rezov pre metódu	Bc. Robert Karasek	\Sprint 2		
Porovnanie implementácií Hougha	Bc. Robert Karasek	\Sprint 2		
Testovanie Hougha na datasetoch	Bc. Robert Karasek	\Sprint 2		

Tabuľka 9.4: Úlohy z tretieho šprintu

Title	Assigned to	Iteration	Description	Tags
Nájdenie bodov s normálou podobnou osi z	Bc. Katarina Janeckova	\Sprint 3	Po nájdení normál pre všetky body vybrať len tie body, ktoré majú podobnú normálu s osou z.	implementacia
Dokončenie implementácie výpočtu normál	Bc. Katarina Janeckova	\Sprint 3		implementacia
Implementácia projekcie bodov na os z	Bc. Katarina Janeckova	\Sprint 3	Projekcia bodov, ktoré majú normálu podobnú s osou z, na os z.	implementacia
Vytvorenie kostry celkovej dokumentácie	Bc. Katarina Janeckova	\Sprint 3	Vytvoriť základný dokument na dopĺňanie pre všetkých členov.	doku
Vytvorenie kapitoly Úvod (inžinierske dielo)	Bc. Katarina Janeckova	\Sprint 3	Napísať úvodnú kapitolu dokumentácie k inžinierskemu dielu.	doku
Vytvorenie podkapitoly Dátový model (inžinierske dielo)	Bc. Katarina Janeckova	\Sprint 3	Do dokumentácie k inžinierskemu dielu vytvoriť dátový model aj s opisom.	doku
Zdokumentovanie metódy 1	Bc. Katarina Janeckova	\Sprint 3	Do dokumentácie k inžinierskemu dielu napísať kapitolu o metóde 1,	doku
Vytvorenie podkapitoly s odkazmi na priložené e-dokumenty (inžinierske dielo)	Bc. Katarina Janeckova	\Sprint 3	Do dokumentácie k inžinierskemu dielu napísať časť, ktorá bude obsahovať zoznam priložených e-dokumentov a ich opis.	doku
Vytvorenie kapitoly Úvod (riadenie)	Bc. Katarina Janeckova	\Sprint 3	Napísanie úvodnej kapitoly do dokumentácie k riadeniu.	doku
Vytvorenie kapitoly Role členov tímu a podiel práce (riadenie)	Bc. Katarina Janeckova	\Sprint 3	Do dokumentácie k riadeniu napísať kapitolu s vysvetlením jednotlivých zodpovedností členov tímu a tiež zapísať ich podiel práce na častiach	doku
Vytvorenie kapitoly Používané metodiky (riadenie)	Bc. Katarina Janeckova	\Sprint 3	Spísať zoznam používaných metodík s krátkym opisom ku každej do dokumentácie k riadeniu.	doku
Vytvorenie kapitoly Zoznam kompetencií tímu (riadenie)	Bc. Katarina Janeckova	\Sprint 3	Do dokumentácie k riadeniu opísať kompetencie tímu.	doku
Vytvorenie kapitoly Export evidencie úloh + urobiť exporty z TFS (riadenie)	Bc. Katarina Janeckova	\Sprint 3	Vytvoriť v dokumentácii k riadeniu kapitolu exportov z TFS.	doku
Zápisnica č. 8 (20151112)	Bc. Katarina Janeckova	\Sprint 3	Vytvorenie zápisnice zo stretnutia č. 8	doku
Vytvorenie syntetických datasetov	Bc. Lukas Hudec	\Sprint 3	Vytvorenie testovacích datasetov pre účely vývoja.	
Vytvorenie podkapitoly Diagram tried (inžinierske dielo)	Bc. Lukas Hudec	\Sprint 3	Do dokumentácie k inžinierskemu dielu vytvoriť diagram tried aj s opisom.	doku

Vytvorenie podkapitoly Architektúra (inžinierske dielo)	Bc. Lukas Hudec	\Sprint 3		
Implementácia oddelenia pointcloudu interiéru	Bc. Lukas Hudec	\Sprint 3	Implementácia metód na vybratie rozdielu dát stien a "zvyšku" pointcloudu	
Identifikácia zhlukov bodov mimo stien	Bc. Lukas Hudec	\Sprint 3		
Rozdelenie vybraných oblastí bodov podľa pozície a veľkosti	Bc. Lukas Hudec	\Sprint 3		
Analýza segmentácie objektov z 3D pointCloudu	Bc. Lukas Hudec	\Sprint 3		
Vytvorenie kapitoly Globálne ciele pre ZS (inžinierske dielo)	Bc. Martin Jurik	\Sprint 3	Do dokumentácie k inžinierskemu dielu spísať ciele, ktoré sme si určili na ZS. Premyslieť ciele, ktoré je možné stihnúť za ZS	doku
Vyriesenie exception s boost library	Bc. Martin Jurik	\Sprint 3	Odstranenie exception odchytenu programom pri return 0;	
implementacia multiselect dialogu a otestovanie	Bc. Martin Jurik	\Sprint 3	Implementacia multiselect dialogu pre otvorenie viacerch suborov a navrat vektora stringov absolutnych ciest pre metody nacitania point cloudov.	
Implementacia eventov a handlerov pre vyber a spustenie metody	Bc. Martin Jurik	\Sprint 3	Implementovanie vyberu metody v gui a naprogramovanie funkcii pre priestor implementacie jednotlivych metod.	
Implementacia spustenia vizualicazie v gui projekte	Bc. Martin Jurik	\Sprint 3		
Implementácia segmentácie podľa vlastností normál	Bc. Michal Korbel	\Sprint 3		
Segmentovanie jednej plochy z objektu podľa vlastností normál	Bc. Michal Korbel	\Sprint 3		
Segmentovanie celého objektu podľa vlastností normál	Bc. Michal Korbel	\Sprint 3		

Vytvorenie kapitoly Sumarizácie šprintov (riadenie)	Bc. Michal Korbel	\Sprint 3	Do dokumentácie k riadeniu napísať kapitolu, v ktorej bude zhrnutie všetkých šprintov. Podľa úloh v TFS spraviť spätný opis/retrospektíva jednotlivých šprintov. Úspechy, neúspechy = "stop doing, start doing, keep doing" z TFS z časti Reporty vytiahnuť dáta o šprintoch (všetko možné čo sa dá ;))	doku
Zdokumentovanie metódy 2 (inžinierske dielo)	Bc. Michal Korbel	\Sprint 3	Do dokumentácie k inžinierskemu dielu napísať kapitolu o metóde 4, ktorá bude obsahovať nasledovné podkapitoly: analýza, návrh, implementácia a testovanie.	doku
Nastavenie stredu rotácie	Bc. Michal Loffler	\Sprint 3	Problém: Knižnice PCL, resp. VTL uplatňujú vo svojich východzích InteractorStyle-triedach správanie, pri ktorom sa kamera rotuje okolo bodu umiestneného v určitej vzdialenosti pred kamerou (focalPoint). Prirodzeným spôsobom je však otáčanie okolo samotného "ťažiska kamery" - podobne ako človek otáča svojou hlavou. Riešenie: Zadefinovať bod otáčania kamery do jej stredu resp. do zanedbateľne malej vzdialenosti pred ňu.	

Vytvorenie kapitoly Aplikácie manažmentov (riadenie)	Bc. Michal Loffler	\Sprint 3	Kapitola do dokumentácie k riadeniu s opisom realizácie jednotlivých činností potrebných pre riadenie projektu, procesu a produktu. Manažment komunikácie a ľudských zdrojov (Róbert Karásek) - stretnutia tímu, komunikačné nástroje, nástroje na zdieľanie obsahu Manažment rozvrhu a rozsahu projektu (Lukáš Hudec) Manažment plánovania projektu (Lukáš Hudec) Manažment kvality (Mišo Löffler) - (zaistenie že projekt jeho výsledky uspokojuje potreby pre ktoré sa vytvoril/inicioval) Manažment rizík (Mišo Löffler) Manažment integrácie a podpory vývoja (Martin Jurík) - (zaistenie že rôzne elementy projektu sú správne koordinované)	doku
Posun doľava a doprava pomocou kláves A a D	Bc. Michal Loffler	\Sprint 3	Problém: Vo východzom nastavení PCL(VTK) je potrebné pre posun (pan) kamery držať stlačené (prostredné/pravé?) tlačidlo myši a súčasne hýbať myšou. Riešenie: Namapovať pohyby doprava a doľava na klávesy A a D.	
Posun dopredu/dozadu pomocou klaves W a S	Bc. Michal Loffler	\Sprint 3	Problém: Vo východzom nastavení PCL(VTK) je potrebné pre pohyb kamery dopredu a dozadu točiť kolečkom myši alebo držať stlačené pravé tlačidlo myši a hýbať ňou dopredu/dozadu. Riešenie: Namapovať pohyby dopredu a dozadu na klávesy W a S.	

Zrušenie nutnosti mať stále stlačené tlačidlo myši pre rotovanie	Bc. Michal Loffler	\Sprint 3	Problém: Ak chceme natáčať pohľad kamery, musíme pritom držať stlačené ľavé tlačidlo myši, čo je trocha nepohodlné. Riešenie: prvým kliknutím sa aktivuje natáčanie kamery pohybom myšou druhým kliknutím sa deaktivuje	
Zdokumentovanie vizualizácie (inžinierske dielo)	Bc. Michal Loffler	\Sprint 3	Do dokumentácie k inžinierskemu dielu napísať kapitolu o vizualizácie, ktorá bude obsahovať nasledovné podkapitoly: analýza, návrh, implementácia a testovanie.	doku
Analýza možností vizualizácie 2D histogramu	Bc. Robert Birkus	\Sprint 3	Cieľom je zistiť možnosti vizualizácie 2D histogramu a zvoliť si ten najvhodnejší pre náš problém.	analyza
Implementácia vizualizácie histogramu	Bc. Robert Birkus	\Sprint 3	Cieľom je implementácia vizualizácie histogramu.	dev
Analýza možností zisťovania dominantných oblastí v 2D histograme	Bc. Robert Birkus	\Sprint 3	Cieľom je zanalyzovať rôzne algoritmy vhodné pre hľadanie dominantných oblastí v 2D histograme.	analyza
Návrh algoritmu pre nájdenie dominantných oblastí	Bc. Robert Birkus	\Sprint 3	Cieľom je navrhnúť algoritmus pre nájdenie dominantných oblastí v histogram. Predpokladá sa použitie mean-shift algoritmu.	design; histogram
Implementácia algoritmu pre nájdenie dominantných oblastí	Bc. Robert Birkus	\Sprint 3	Implementácia navrhnutého algoritmu pre nájdenie dominantných oblastí v histograme.	dev; histogram
Otestovanie správnosti algoritmu pre nájdenie dominantných oblastí	Bc. Robert Birkus	\Sprint 3	Cieľom je otestovať správnosť a vhodnosť algoritmu pre nájdenie dominantných oblastí v 2D histograme.	histogram; test
Nasadenie implementácie výpočtu a vizualizácie histogramu normál na hlavný projekt	Bc. Robert Birkus	\Sprint 3	Cieľom je nasadenie implementácie histogramu na náš hlavný projekt.	deploy; dev
Zdokumentovanie metódy 4	Bc. Robert Birkus	\Sprint 3	Do dokumentácie k inžinierskemu dielu napísať kapitolu o metóde 2,	doku
Zdokumentovanie metódy 3	Bc. Robert Karasek	\Sprint 3	Do dokumentácie k inžinierskemu dielu napísať kapitolu o metóde 3,	doku

Vytvorenie metodiky pre komunikáciu a pre commiting/branching	Bc. Robert Karasek	\Sprint 3	Spísanie metodiky pre komunikáciu členov tímu v aplikácii Slack Metodika pre správny commiting + merging do projektu, formát správ commitu - premyslieť aby to dobre vyzeralo a malo informačnú hodnotu pre ostatných členov tímu.	doku
Priprava Testovacich scenarov	Bc. Robert Karasek	\Sprint 3		
Príprava konvencií pre generovanie technickej dokumentácie	Bc. Robert Karasek	\Sprint 3	Príprava základných konvencií a návodu ako vygenerovať technickú dokumentáciu	
Analýza histogramu Furukawa	Bc. Robert Karasek	\Sprint 3	Analýza histogramu z článku pre metódu Furukawa	
Implementácia histogramu pre Furukawa	Bc. Robert Karasek	\Sprint 3	Implementovanie histogramu, ktorý bude slúžiť na rozdelenie 3D priestoru na 2D rezy	
Testovanie histogramu Furukawa	Bc. Robert Karasek	\Sprint 3	Testovanie histogramu	
Analýza rozdelenia 3D pointcloudu na 2D rezy	Bc. Robert Karasek	\Sprint 3		
Implementácia rozdelenia 3D pointcloudu na 2D rezy	Bc. Robert Karasek	\Sprint 3	Implementácia bude zložitejšia nakoľko v papieroch nie je presný návod a web je nefunkčný, podobnú metódu na rovnakom princípe som nenašiel	
Testovanie rozdelenia 3D pointcloudu na 2D rezy	Bc. Robert Karasek	\Sprint 3		
Implementácia Gaussian smoothing	Bc. Robert Karasek	\Sprint 3		

Tabuľka 9.5: Úlohy zo štvrtého šprintu

Title	Assigned to	Iteration	Description	Tags
Analýza binary inside/outside segmentation	Bc. Katarina Janeckova	\Sprint 4	-pochopenie princípu binárnej inside/outside segmentácie - prehľad o implementácii v knižniciach - urobiť prehľad o danej metóde pre zvyšok tímu	analyza
Implementácia extrakcie rezu	Bc. Katarina Janeckova	\Sprint 4		implementacia
Implementácia metódy mean shift	Bc. Katarina Janeckova	\Sprint 4		
Analýza outline simplification	Bc. Katarina Janeckova	\Sprint 4		
Funkčná integrácia podporných modulov	Bc. Lukas Hudec	\Sprint 4	Prepojiť existujúce a vytvorené moduly s hlavným GUI	
Projekcia 3D primitív na 2D zobrazenie pôdorysu	Bc. Lukas Hudec	\Sprint 4	Pohľad zhora - zmena 3D na 2D zobrazenie a príprava na spracovanie pôdorysu	
Integrácia metód do gui	Bc. Lukas Hudec	\Sprint 4	Integrácia spúšťania metód cez gui.	
Reštrukturalizácia architektúry	Bc. Lukas Hudec	\Sprint 4	Z dôvodu vyššej efektivity a lepšieho rozvrhnutia dizajnu architektúry projektu je potrebné zmeniť implementáciu modulov, rozbiť zložité štruktúry podľa tematického zamerania a znížiť závislosť na includoch.	implementacia
Zistenie roviny z bodov	Bc. Lukas Hudec	\Sprint 4	Pomocou štatistiky a metód analytickej geometrie získať funkčný predpis aproximovanej roviny, ktorá predstavuje na abstraktnej úrovni priestor v ktorom sa nachádzajú body rovnakého príznaku predstavujúce rovinu, ktorú hľadáme.	implementacia
Zápisnica č.9 (20151120)	Bc. Lukas Hudec	\Sprint 4	Spísanie zápisnice z 9. stretnutia a jej sprístupnenie pre celý tím.	
Implementacia spustenia vizualicazie v gui projekte	Bc. Martin Jurik	\Sprint 4	Riesenie vtk erroru a zatvaranie okna vizualizacie z gui	
Analýza binary inside/outside segmentation	Bc. Martin Jurik	\Sprint 4		
Analýza outline simplification	Bc. Martin Jurik	\Sprint 4		
Zápisnica č.10 (20151126)	Bc. Martin Jurik	\Sprint 4		
Zistenie roviny z bodov	Bc. Michal Korbel	\Sprint 4	- Ransac model plane - výpočet koeficientov roviny (ax +by + cz +d = 0) - preloženie roviny v smere plochy	
Oboznámenie sa s 3Dconexxion SDK	Bc. Michal Loffler	\Sprint 4		
Ošetrenie situácie keď je kurzor mimo okna vizualizácie	Bc. Michal Loffler	\Sprint 4		
Implementácia posunu nahor a nadol (elevation)	Bc. Michal Loffler	\Sprint 4		
Implementácia rotácie kamery	Bc. Michal Loffler	\Sprint 4		
Implementácia interakcie pomocou 3D myši (knižnica VTK)	Bc. Michal Loffler	\Sprint 4	1. prekompilovanie knižnice VTK 2. nastavenie vtkRenderWindowInteractor 3. implementácia event observerov	
Analýza možností knižnice VTK pre ovládanie 3D myšou	Bc. Michal Loffler	\Sprint 4		
Navrhnúť spôsob označovania bodov	Bc. Robert Birkus	\Sprint 4		design

Implementácia označovania bodov	Bc. Robert Birkus	\Sprint 4	Implementovanie označovania všetkých bodov v point cloude pomocou dominantných normál. Podľa normály jednotlivých bodov sa priradí bod k najpodobnejšej dominantnej normály	dev
Otestovanie presnosti a správnosti označovania bodov	Bc. Robert Birkus	\Sprint 4	Je potrebné otestovať efektívnosť označovania bodov v pointcloude vizuálne z hľadiska segmentácie bodov reprezentujúcich stenu miestnosti.	test
Analýza možností zisťovania dominantných oblastí v 2D histograme	Bc. Robert Birkus	\Sprint 4	Cieľom je zanalyzovať rôzne algoritmy vhodné pre hľadanie dominantných oblastí v 2D histograme. Po analýze sme sa rozhodli, že najvhodnejší algoritmus pre daný problém bude mean-shift. A keďže knižnice, ktoré používame obsahujú implementáciu mean-shiftu len pre špecifické úlohy je potrebné si navrhnúť a naimplementovať vlastný mean-shift pre náš problém.	analyza
Návrh algoritmu pre nájdenie dominantných oblastí	Bc. Robert Birkus	\Sprint 4	Cieľom je navrhnúť algoritmus pre nájdenie dominantných oblastí v histogram. Predpokladá sa použitie mean-shift algoritmu.	design; histogram
Implementácia algoritmu pre nájdenie dominantných oblastí	Bc. Robert Birkus	\Sprint 4	Implementácia navrhnutého algoritmu pre nájdenie dominantných oblastí v histograme.	dev; histogram
Otestovanie správnosti algoritmu pre nájdenie dominantných oblastí	Bc. Robert Birkus	\Sprint 4	Cieľom je otestovať správnosť a vhodnosť algoritmu pre nájdenie dominantných oblastí v 2D histograme.	histogram; test
Nasadenie implementácie výpočtu a vizualizácie histogramu normál na hlavný projekt	Bc. Robert Birkus	\Sprint 4	Cieľom je nasadenie implementácie histogramu na náš hlavný projekt.	deploy; dev
Vytvorenie stručných pravidiel pre komentovanie	Bc. Robert Karasek	\Sprint 4		
Vytvorenie prvých správne okomentovaných súborov	Bc. Robert Karasek	\Sprint 4	Jedná sa o správne anotácie v komentároch, podľa ktorých sa môžete držať.	
Štúdium Unit Testov	Bc. Robert Karasek	\Sprint 4		

Tabuľka 9.6: Úlohy zo štvrtého šprintu

Title	Assigned to	Iteration	Description	Tags
Práca na dokumentácii	Bc. Katarina Janeckova	\Sprint 5	Spísanie nových vecí v rámci projektu do dokumentácie v častiach, v ktorých najviac prispievam.	doku
Finálne spracovanie dokumentácie k ZS	Bc. Katarina Janeckova	\Sprint 5	Doplnenie potrebných vecí, konečné úpravy, kompletizácia.	doku
Zjednodušenie pridania nového objektu steny	Bc. Lukas Hudec	\Sprint 5	Vzhľadom na implementačnú zložitosť vytvorenia nového objektu steny, je potrebná jeho reštrukturalizácia a zjednodušenie po maximálnu abstrakciu - overloading konštruktora.	
Hľadanie priesečníkov rovín - body ohraničenia steny	Bc. Lukas Hudec	\Sprint 5		
Práca na dokumentácii	Bc. Lukas Hudec	\Sprint 5	Spísanie nových vecí v rámci projektu do dokumentácie v častiach, v ktorých najviac prispievam. Doplnenie a úprava metodiky evidencie úloh Doplnenie a úpravy dokumentácie inžinierskeho diela v časti architektúra podľa posledných zmien Dokumentácia Doxygen komentáre do kódu	doku
Primitívne 3D zobrazenie primitív	Bc. Lukas Hudec	\Sprint 5	prvý pokus pre zobrazenie primitív bude ich jednoduché 3D premietnutie pcl vizualizérom	
Práca na dokumentácii	Bc. Martin Jurik	\Sprint 5	Spísanie nových vecí v rámci projektu do dokumentácie v častiach, v ktorých najviac prispievam.	doku
Implementovanie funkcionality check boxov	Bc. Martin Jurik	\Sprint 5	Implementovanie vyberu point cloudov na zaklade vybranych check boxov	
Rozsirenie GUI o featuru pre Birkyho	Bc. Martin Jurik	\Sprint 5		
Práca na dokumentácii	Bc. Michal Korbel	\Sprint 5	Spísanie nových vecí v rámci projektu do dokumentácie v častiach, v ktorých najviac prispievam.	doku
Nájdenie hraničných čiar objektu pomocou Houghove	Bc. Michal Korbel	\Sprint 5		
Segmentovanie jednej plochy z objektu podľa rezov ro	Bc. Michal Korbel	\Sprint 5		
Práca na dokumentácii	Bc. Michal Loffler	\Sprint 5	Spísanie nových vecí v rámci projektu do dokumentácie v častiach, v ktorých najviac prispievam.	doku
Zmenšenie veľkosti kroku pri pohybe do strán	Bc. Michal Loffler	\Sprint 5		
Zjemnenie plynulosti pohybu do strán	Bc. Michal Loffler	\Sprint 5	Zatiaľ sa neporadilo naimplementovať dôvod: pri použití vtkRepeatingTimer je príliš dlhá odozva na stláčanie kláves. (vtkRepeatingTimer je automatický časovač, ktorý po stlačení klávesy pre pohyb začne v pravidelných krátkych intervaloch generovať pohyb o malý kúsok dopredu/ do strany)	
Výpomoc pri hľadaní priesečníkov rovín	Bc. Michal Loffler	\Sprint 5		

Práca na dokumentácii	Bc. Robert Birkus	\Sprint 5	Spísanie nových vecí v rámci projektu do dokumentácie v častiach, v ktorých najviac prispievam.	doku
Návrh a implementácia metódy pre nájdenie dominan	Bc. Robert Birkus	\Sprint 5	Je potrebné navrhnúť a implementovať metódu pre nájdenie dominantných oblastí v histogram normál pomocou hľadania maxím.	design; dev
Kontrola správneho fungovania metódy	Bc. Robert Birkus	\Sprint 5	Je potrebné overiť, či naimplementovaná metóda vykonáva požadovanú funkcionalitu.	test
Nasadenie fungujúcich častí rekonštrukčnej metódy č.	Bc. Robert Birkus	\Sprint 5	Hotové časti metódy č.4 je potrebné nasadiť do hlavného projektu.	deploy
Nastavenie knižnice OpenCV 3.0.0 v debug móde v hla	Bc. Robert Birkus	\Sprint 5	V hlavnom projekte sa knižnica OpenCV 3.0.0 bije s knižnicou boost a každá z knižníc si vyžaduje iné nastavenia. Je potrebné vyriešiť, aby v projekte fungovali obi dve knižnice naraz. Táto úloha je vykonávaná v spolupráci s Martinom Juríkom	configuration
Zápisnica č.11 (20151203)	Bc. Robert Birkus	\Sprint 5	Spísanie zápisnice z agile stretnutia tímu	doku
Vytvorenie Unit Testov	Bc. Robert Karasek	\Sprint 5		
Build Google Test pre projekt	Bc. Robert Karasek	\Sprint 5	Príprava buildu google testov pre náš projekt	
Práca na dokumentácii	Bc. Robert Karasek	\Sprint 5	Spísanie nových vecí v rámci projektu do dokumentácie v častiach, v ktorých najviac prispievam. Metodika technickej dokumentácie Metodika testovania Príprava a vygenerovanie technickej dokumentácie	doku

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Tímový projekt

Rekonštrukcia 3D scény

Projektová dokumentácia – inžinierske dielo

Vedúci projektu: Ing. Vanda Benešová, CSc.

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. máj 2016

Obsah

1. Úvod

Tento dokument predstavuje projektovú dokumentáciu softvérového systému na rekonštrukciu 3D scény, ktorý je výsledkom zadania na predmete Tímový projekt 1 (TP1) a Tímový projekt 2 (TP2).

1.1. Prehľad dokumentu

V časti 2 sú spísané ciele, ktoré sme chceli v rámci projektu dosiahnuť. Kapitola 3 sa zaoberá architektúrou nášho systému, je v nej tiež uvedený prehľad modularizácie systému (3.1.2) a diagram tried (Obrázok 3, Obrázok 4). Podrobný opis tried systému, respektíve rekonštrukčných metód, sa nachádza v kapitole 4. Inštalačná príručka je v závere dokumentu ako príloha.

1.2. 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 sur-face 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 dôle-žitá budúca aplikácia by mohla byť napr. aj fúzia 3D stereo dát snímaných na Marse. (US missions MER, MSL, Mars 2020; ESA Mission ExoMars 2018).

2. Globálne ciele

2.1. Zimný semester

Medzi hlavné ciele zimného semestra 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 rôznych 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 prostredníctvom vytvorenej aplikácie
    - Navrhnúť GUI aplikáciu pre interakciu používateľa
    - Implementovať možnosť výberu súborov s dátami a výberu metódy rekonštrukcie
    - Nastavenie parametrov zobrazenia vizualizácie
    - Implementácia ovládania kamery vizualizéra
    - Vizualizovať výstup spracovania pomocou zvolených metód
- Zhodnotenie práce za dané obdobie
- Konzultácia s product ownerom
    - Predvedenie výsledku práce
    - Dohodnutie ďalšieho postupu podľa požiadaviek

2.2. Letný semester

- Unit testy
- Konfiguračný súbor
- XML Parser
- Výstup vo formáte CAD
- 3D myš
- Doxygen – technická dokumentácia
- Predspracovanie datasetu
    - Odstrániť šum
    - Štatistiky datasetu (štandardná odchýlka)
- Rekonštrukcia objektov scény
    - Nájsť priesečníky rovín

- Ohraničenie nájdených rovín
- Filtrovanie outliers a ich segmentácia

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. Okrem toho predpokladáme jej ďalšie využitie v projektoch vedených na FIIT, či už tímových tak aj diplomových.

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. Dostupné a vhodné technológie sa ukázali knižnice PCL „Point Cloud Library“ a knižnica OpenCV. Časom sa ukázalo, že pre prácu v 3D je knižnica PCL viac než dostačujúca. 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, pripravenej pre ďalšiu modularizáciu a možné rozšírenie v budúcnosti.

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 implementácií 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 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 navrhnúť 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 funkcionalita je rekonštrukcia 3D dát a prípadne segmentácia objektov. Pre tento výstup je nežiaduce, keby bola táto knižnica viazaná na tematicky odlišnú funkcionalitu – gui, prípadne vizualizáciu, prípadne ovládač. Preto je potrebné navrhnúť 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 neplytvať pamäťou a zdieľať čo najväčšie množstvo dát, ktoré je možné.

3.1.2. Modularizácia systému

Z architektonického hľadiska nezávislosti modulov vyplynulo rozdelenie implementačného riešenia projektu na 5 modulov. Z hľadiska plánovania a rozdelenia projektu časť týchto modulov predstavovala Epic úlohy evidované v TFS backlog-u. Tieto moduly sú:

- 3DReconstruction - hlavná logika aplikácie, dátové typy poznajú štruktúru jednotlivých načítaných dát
- Exporter – spracúvajúci výstup metód rekonštrukcie do zvoleného formátu DXF otvoriteľného v akomkoľvek CAD programe alebo vizualizéri
- GUI - Grafické rozhranie pre komunikáciu s používateľom

- Vizualization - Vizuálny výstup riešenia ako spätná informácia pre používateľa (Okrem vizualizácie rieši aj interakciu)
- DataHandling - Správa súborov a načítanie/konverzia dát

Diagram toku dát:

3DReconstruction

Tento modul obsahuje hlavnú naším tímom vyvíjanú a implementovanú logiku rekonštrukcie a segmentácie. Hlavná funkcionalita rekonštrukčných metód sa nachádza v triedach dediacich od triedy Reconstruction. Od nej dedia z GUI volané metódy calculate, visualise a zapuzdrenie datasetu, nad ktorým operujú. Postupný vývoj nám tiež umožnil rozdeliť metódy segmentácie a vyčleniť elementárne funkcie použiteľné vo viacerých metódach. Deklaráciou v rodičovskej abstraktnej triede je možné ich zavolať v akejkoľvek triede potomka a tiež ich prevážiť vlastnou implementáciou. Dátová trieda, ktorá udržiava vstupné surové dáta je ClosedSpace. Zámer bol vytvoriť ju typovo nezávislú, takže dokáže udržiavať dátové typy akéhokoľvek formátu. Okrem správy dát, udržiava spracované rekonštruované objekty scény. Tiež pozná svoje typy objektov a momentálne dokáže vizualizovať definované objekty stien, resp. rovinné objekty, z hľadiska PCL a VTK vizualizéra a podľa jednoduchého pravidla priradenou náhodnou farbou.

Pre udržiavanie dát a rekonštruovaných objektov sa implementovali abstraktné triedy Object a Primitives. Akýkoľvek ďalší nový definovaný typ primitívy a objektu reálneho sveta dedí od týchto tried. Takto definované sú triedy Rectangle, Polygon a objekt reálneho sveta Wall. Z hľadiska klasifikácie objektov, však ich momentálne využitie možno nie je najvhodnejšie. Nezávislým typom primitívy je trieda Line, ktorá definuje priamku. Všetky tieto analytické primitívy obsahujú svoje matematické vyjadrenie a bodové ohraničenie. Triedy objektov reálneho sveta obsahujú množinu bodov, ktorá im prislúcha a v kompozite všetky primitívy, ktoré ho tvoria. Trieda Wall podľa toho ako sme ju určili obsahuje jeden rovinný polygón. Takto bola dosiahnutá maximálna abstrakcia typov a v ďalších implementáciách objektový polymorfizmus.

Ďalšie triedy, ktoré prispievajú svojou logikou sú podporná trieda so všeobecnou funkcionalitou, trieda pre výpočet štatistík segmentu a generátor farieb pre segmenty. Štatistiky, ktoré sú užitočné pre riešenie problému rekonštrukcie sú smerodajná odchýlka presnosti merania senzora a rozptyl nameraných bodov. Ich výpočet je implementovaný v triede DataStatistics. Pre ďalšie riešenia problému segmentácie malých objektov na nižšej úrovni bola implementované triedy PlanePointsExamination a PlaneSegment. Je zjavné, že všade sa hovorí o rovinných telesách – toto vyplýva z našej špecifikácie problému. Bližšie informácie k spomínaným funkciám sú v technickej dokumentácií a implementačnej časti dokumentu.

Exporter

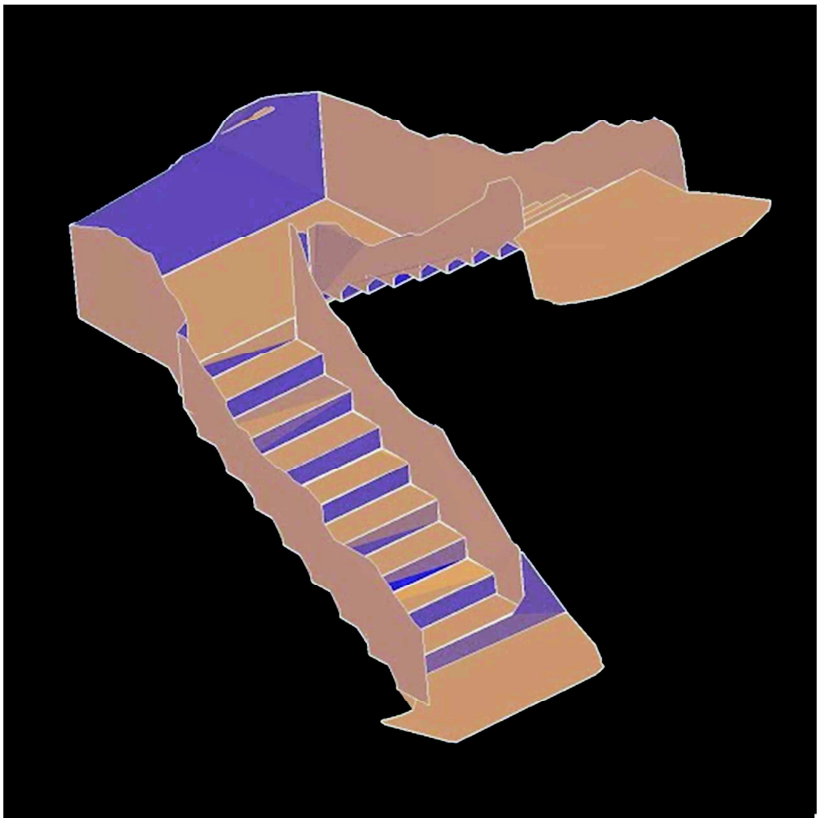
Je dôležité, aby náš nástroj dokázal zrekonštruované objekty perzistentne ukladať do súboru. Umožní to totiž dodatočnú vizualizáciu a manipuláciu aj ľuďom, ktorí tento nástroj nemajú nainštalovaný.

Preto vznikol modul Exporter, ktorý zabezpečuje export objektov do štandardného formátu DXF pre parametrické počítačové modelovanie. Vďaka použitiu tohto formátu je možné s modelom pracovať v množstve zobrazovacích či modelovacích nástrojov, ako napr. AutoCAD.

V súčasnosti tento modul sprístupňuje vývojárovi rekonštrukčných metód dve elementárne funkcie, prostredníctvom ktorých dokážu vyexportovať celú scénu. Sú to funkcie addWalls() na export planárnych objektov v scéne a addMesh() pre všetky ostatné neplanárne objekty vo forme siete trojuholníkov.

Je iba na voľbe programátora ako tieto funkcie použije. Môže pomocou nich umožniť export všetkých identifikovaných útvarov, alebo aj iba ich podmnožinu, napr. iba vertikálne plochy.

Obe spomenuté metódy pridávajú príslušné elementárne útvary do virtuálnej scény, ktorá sa po skompletizovaní zapíše do DXF súboru pomocou funkcie writeDXF(). Táto funkcia je volaná vždy na žiadosť používateľa, ktorý v grafickom rozhraní klikne na tlačidlo "Export DXF".



Obrázok 1 Zobrazenie vyexportovanej scény v programe AutoCAD 2015

GUI

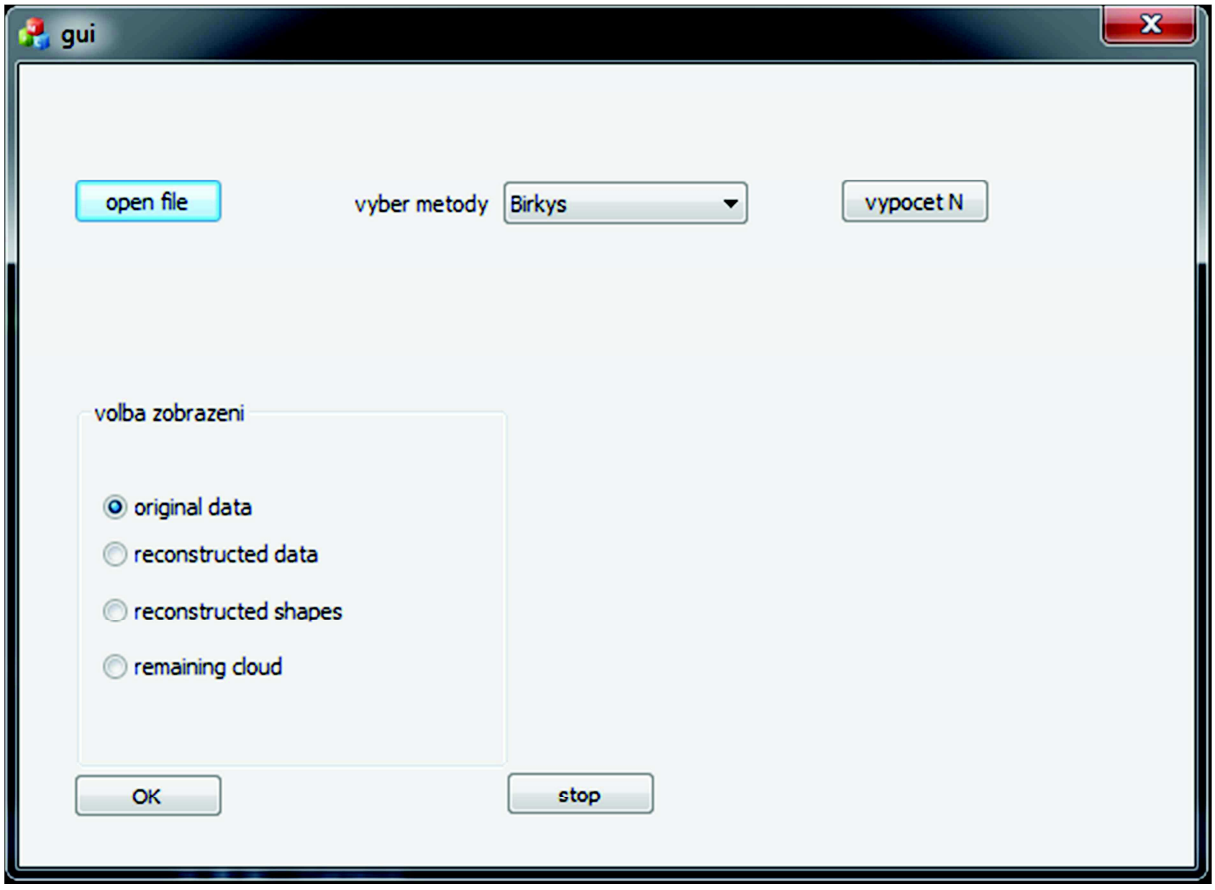
Každý program potrebuje grafické rozhranie pre interakciu s používateľom. Najmä vtedy, keď potrebuje od používateľa aby si vybral metódu, ktorou chce vykonať rekonštrukciu, dáta ktoré chce vizualizovať a aj samotné dáta, ktoré má v pláne spracovať. Keďže je to tematicky odlišný modul, bol tak aj vytváraný 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.

Opis ovládania a funkcionality GUI

Obrázok 2 zobrazuje GUI rozhranie projektu.

- open file - slúži na výber jedného alebo viacerých súborov pre načítanie
- list box vyber metódy - umožňuje vybrať jednu z metód pre použitie na spracovanie zvolených a načítaných súborov
- skupinka radio buttonov: voľba zobrazení - je určená pre výber jedného typu požadovaného point cloudu pre zobrazenie
    - original data - pre zobrazenie celého, pôvodného, neupraveného point cloudu
    - reconstructed data - body pre rekoštrukciu plôch reconstructed shapes - zrekonštruované plochy
    - remaining cloud - pre zobrazenie zostatkovej/nespracovanej časti spracovaného point cloudu
- OK button - spustí vizualizér, podmienkou je mať zvolené vstupné súbory, metódu a voľbu zobrazenia
- stop button - zastaví a zatvorí vizualizáciu, následne je možnosť ďalej interagovať s GUI, načítavať iné súbory, zvoliť si iné metódy či volby zobrazní a zase spustiť vizualizáciu s inými nastaveniami



Obrázok 2 Screen GUI

Vizualization

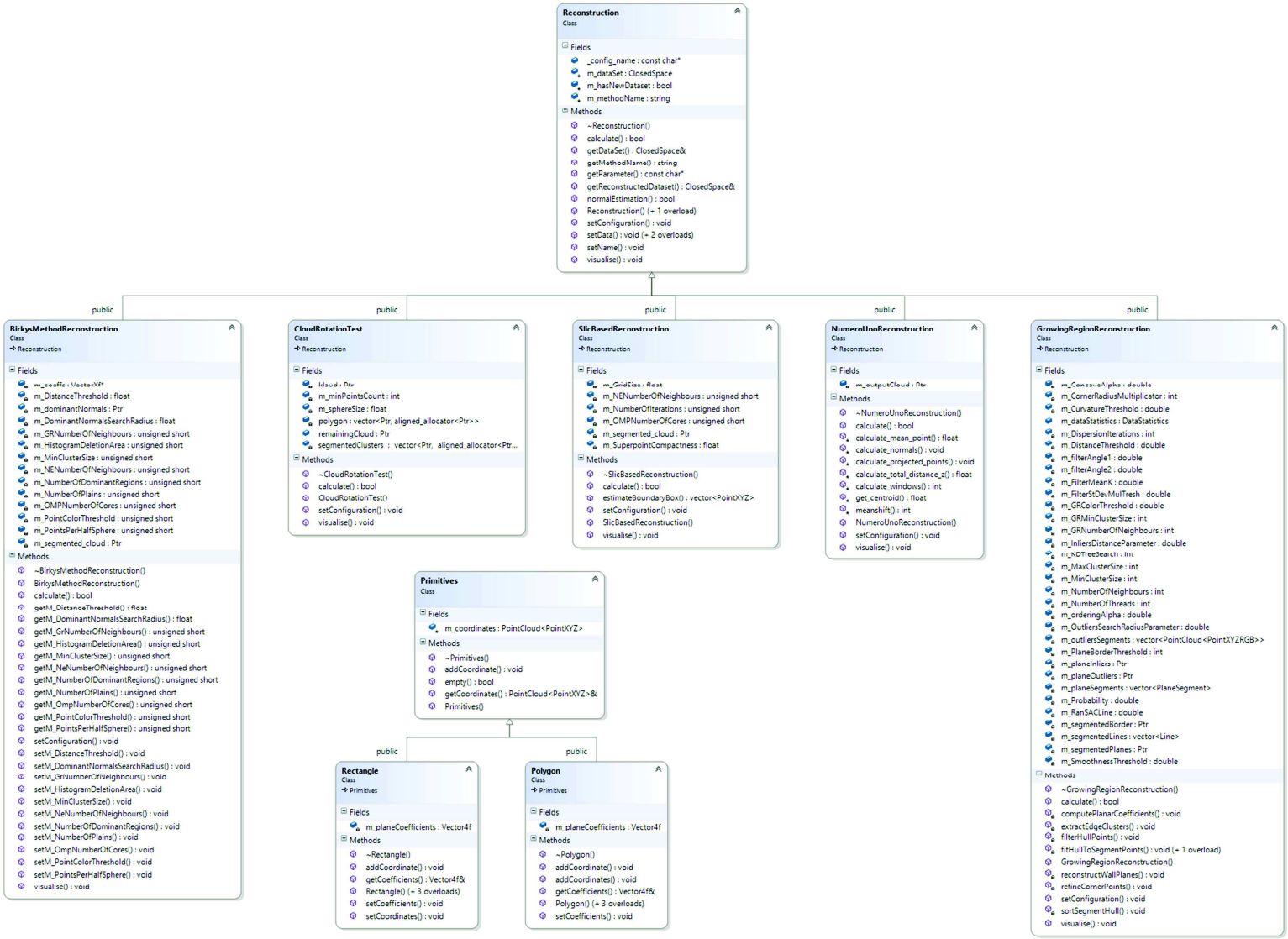
Vizualizačný model mohol byť prepojený priamo na GUI, dôvod jeho oddelenia však spočíva v rozdielnych metódach použitých pri ich implementácií na čo bolo potrebné myslieť už v rámci návrhu architektúry. Momentálna implementácia modulu rieši najmä interakciu používateľa so zobrazovanou scénou a pohyb kamery. Preto tiež aj z hľadiska čiastočnej tematickej odlišnosti bol tento modul oddelený od GUI.

DataHandling

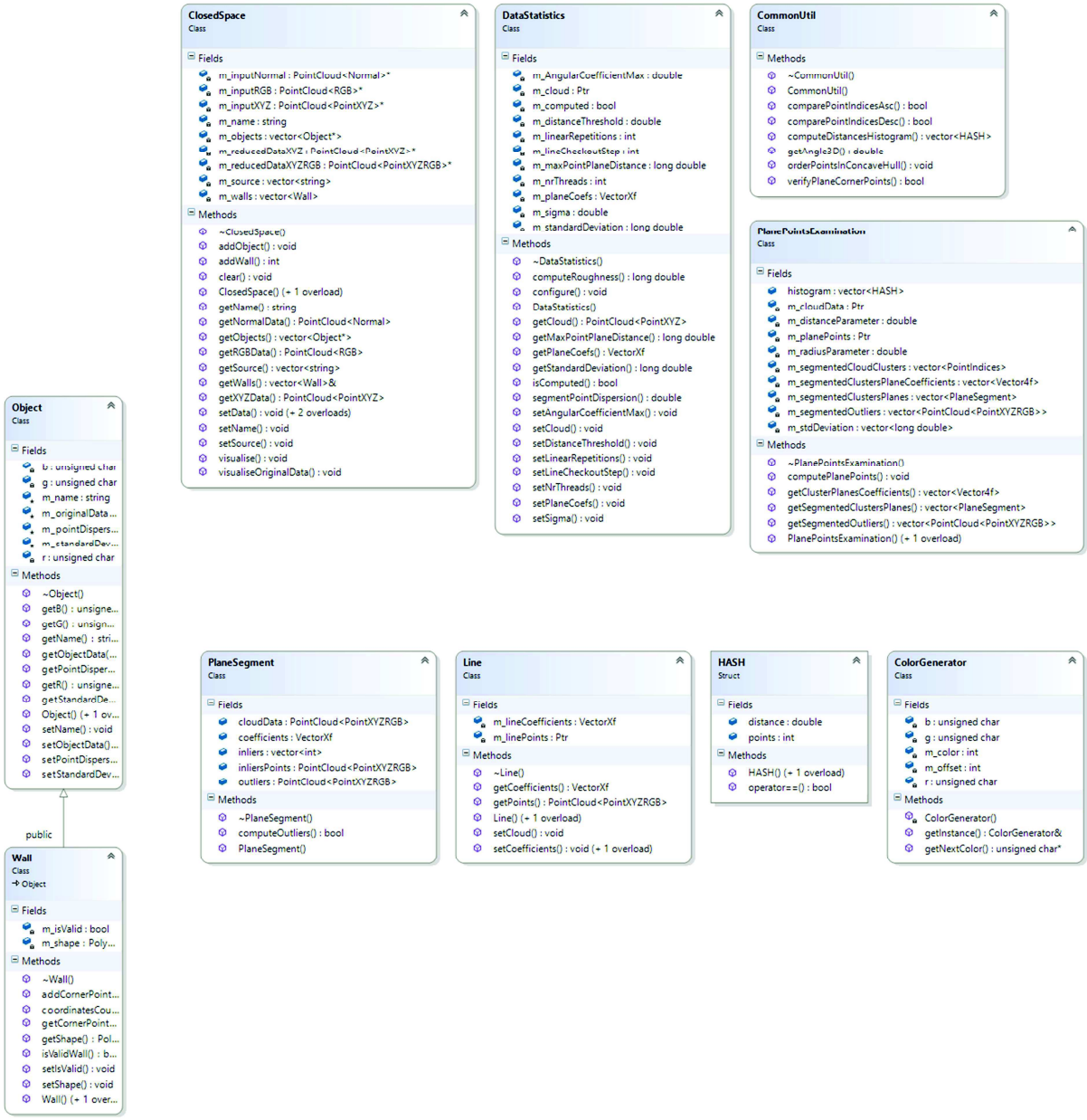
Modul, ktorý vznikol z dôvodu lepšej modularizácie a lepších kompilačných časov, vzhľadom na oddelenie niektorých málo menených tried – kódu, ktorý sa od jeho vytvorenia už nemení. Takto sa oddelila funkcionalita načítavania dát z triedy „ClosedSpace“ do osobitného modulu. Čo sa hneď využilo, keď prišli nové dáta – aj nový formát dát, tým pádom vďaka flexibilite bolo pridanie možnosti načítavania bezproblémové a na vyšších vrstvách neboli potrebné žiadne úpravy. Ďalšia oddelená funkcionalita bol XML parser pre čítanie konfiguračného súboru. Osobitnou kategóriou je metóda odstránenia šumu z dát – ich predspracovanie.

3.1.3. Segmentačná a rekonštrukčná logika –metódy rekonštrukcie

Pôvodný výskumný a implementačný plán nášho tímu bol vytvoriť knižnicu s viacerými rekonštrukčnými metódami. Do určitej úrovne sa nám tento cieľ daril plniť. V určitom momente však v dvoch metódach začalo dochádzať ku konvergencií krokov a ďalšie metódy vyžadovali pre nás nedostupné množstvo zdrojov. Preto sa celkový prístup zmenil. Architektúra ostala, ale nie všetky rekonštrukčné metódy sú úplne rekonštrukčné. Zhodnotili sme, že pre úspešnú implementáciu a dosiahnutie našich prioritných cieľov plne postačuje jedna optimálna rekonštrukčná metóda. Takto sa postupne zredukoval vývoj zo 4 metód rekonštrukcie na jednu s tým, že sa využili úspešné kroky z jednotlivých už implementovaných krokov. Takto sa rozdelilo naše tímové zloženie na segmentačnú, rekonštrukčnú a „outliers“ divíziu. Spojením najlepších dostupných implementácií sme tak vytvorili úspešný dátový tok, ktorý spĺňa stanovené ciele.



Obrázok 3 Diagram tried (part 1)



Obrázok 4 Diagram tried (part 2)

4. Rekonštrukčné triedy 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. Takéto 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 nasledujúcich častiach.

4.1. NumeroUnoReconstruction - pozastavená

4.1.1. Analýza

Metóda sa zakladá na článku …. . 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 taká 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 označkujeme pixely ako v rámci objektu alebo mimo objektu. Ohraničíme tie, ktoré do objektu patria a získame 2D plán jedného rezu. Obrysy 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 je v štádiu, kedy sa podarilo vytvoriť metódu na určenie normál všetkých bodov, nájdenie bodov s normálou podobnou osi z a následné premietnutie týchto bodov na os z. Ďalej sme implementovali metódu mean shift, pomocou ktorej určujeme umiestnenie horizontálnych rezov modelom.

4.2. GrowingRegionReconstruction

4.2.1. Analýza

Rastúci región je segmentačná metóda založená na porovnávaní susedných bodov v oblaku a následnom vytvorení filtrovaných oblastí (regiónov) prislúchajúcich bodov. Na začiatku je potrebné si stanoviť počiatočné body, ktoré považujeme za štartovacie body segmentácie. Podľa ich rozloženia sa vyvíja celý proces segmentácie obrazu. Po stanovení počiatočného 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 počiatočného bodu a normálou každého susedného bodu. Ak uhol spadá pod stanovenú prahovú hodnotu, susedné body z oblaku patria do novo vysegmentovanej oblasti. V opačnom prípade body vynecháme. V ďalšom kroku za počiatočné 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. “outlier” body, 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 implementácie elementárnych funkcií výpočtov. Napríklad v tomto prípade sú to normály, prípadne výpočet RGB hodnoty z bodov a podobne.

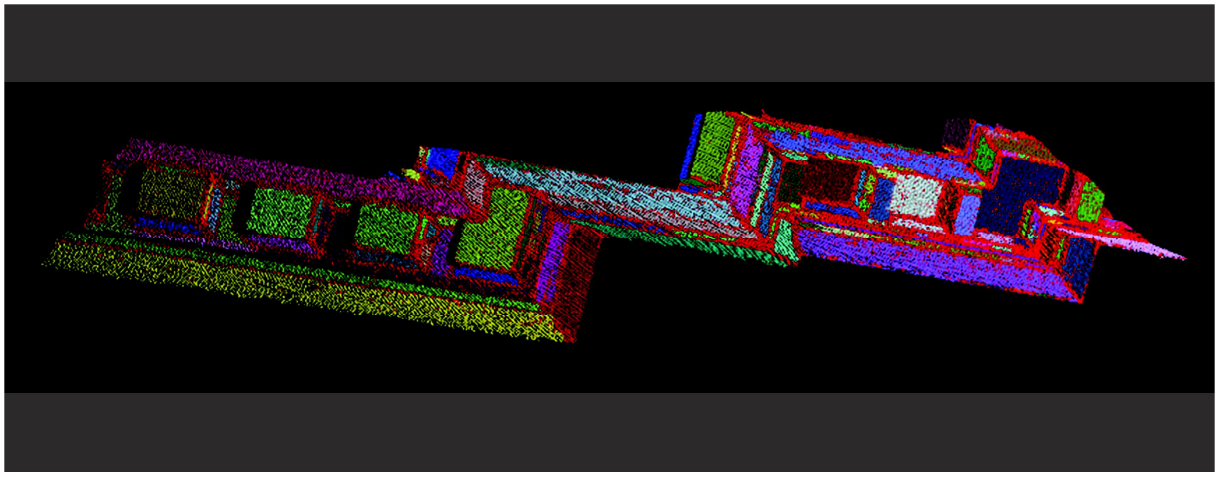
4.2.2. Návrh

Metóda bola pôvodne implementovaná pomocou triedy GrowingRegionReconstruction, ktorá reprezentuje rozšírenie hlavnej rekonštrukčnej triedy 3Dreconstruction. Metóda dostala na vstupe oblak dát, ktorému sa najprv vypočítali normály a bol spracovávaný algoritmom rastúci región. Na výstupe poskytoval zrekonštruovaný oblak dát pomocou tohto algoritmu.

4.2.3. Implementácia

V súčasnosti bol vývoj tejto vetvy zrušený z dôvodu príliš veľkej časovej náročnosti na segmentáciu. Vytvorený prototyp algoritmu, ktorý segmentoval vstupný oblak bodov na jednotlivé roviny z bodov do klastrov. Každý takto vytvorený klaster obsahoval body označené jednou príslušnou farbou. Problém bol aj v tom, že nie všetky body sa pridelili nejakej rovine. Takto ostávali miesta, ktoré predstavovali malé objekty, ale vďaka rastúcemu regiónu, ktorý bol príliš prísny na zmeny normál sa tieto oblasti stratili. Body nespadajúce pod žiaden klaster sú znázornené na obrázku 5 ako červené body.

Každý zo segmentovaných klastrov predstavoval jednu rovinu vo forme bodov. Kľúčovým krokom bolo tieto klastre reprezentovať čo najväčšími celistvými geometrickými útvarmi. Na tento účel sme využili metódu RANSAC, ktorá je dostupná v knižnici PCL. Tá zo vstupného klastru bodov na základe zadanej prahovej hodnoty odstráni "outlier" body a vrátila koeficienty parametrického vyjadrenia roviny, v ktorej tieto body ležia. Tým pádom sme však získali iba množinu neohraničených rovín. Preto je potrebné určiť ich ohraničenia. To sme dosiahli aplikovaním funkcie planeWithPlaneIntersection knižnice PCL, ktorou sme získali priesečníky jednotlivých plôch vo forme priamok (resp. koeficientov v ich parametrickom vyjadrení).



Obrázok 5 Metóda growing region

4.3. BirkysMethodReconstruction

4.3.1. Analýza

Celkovým cieľom tejto segmentačnej metódy je vysegmentovať oblak bodov na jednotlivé segmenty reprezentujúce objekty v 3D scéne (steny, dvere, skrine, atď.). Metóda je založená na histograme normál bodov. Cieľom je pomocou výpočtu normál jednotlivých bodov (na základe 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, či už na stenách alebo v samotnej miestnosti. Tento problém by sa však mohol riešiť predspracovaním 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, vysegmentovanie oblaku bodov na základe priestorovej vzdialenosti a predchádzajúceho označenia bodov (rastúci región).

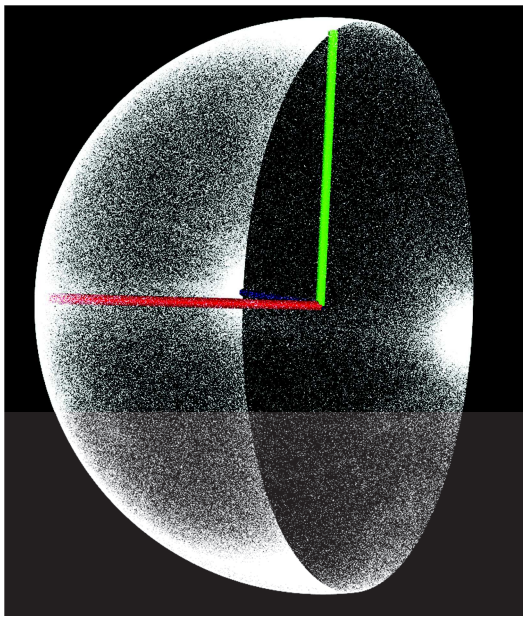
V našom projekte používame knižnicu PCL (Point Cloud Library), ktorú sa snažíme maximálne využiť pre uľahčenie našej práce. PCL umožňuje výpočet normál bodov. Normály jednotlivých bodov vypočítava pomocou susedných bodov zo zadefinovaného okolia.

Vypočítané normály bodov obsahujú normalizovaný vektor normál x, y, z a tiež aj zakrivenie. Podrobné vysvetlenia výpočtu normál bodov v PCL je na oficiálnej stránke PCL. Možnosti PCL pre výpočet histogramu sú vcelku obmedzené, keďže obsahuje iba implementáciu špecifických histogramov pre špecifické úlohy, ktoré pre našu úlohu nie sú vhodné. Bolo teda potrebné si navrhnúť a naimplementovať vlastný histogram pre normály bodov. Pre určenie dominantných oblastí v histograme bolo potrebné si zvoliť vhodný algoritmus, ktorý závisí predovšetkým od navrhnutého histogramu. Po zistení dominantných smerov z histogramu normál je potrebné prideliť každý bod k najbližšiemu z dominantných smerov. Ďalším problémom je, že body dvoch alebo viacerých rovnobežných rovín budú pridelené k rovnakému dominantnému smeru a pre ďalšie spracovanie potrebujeme, aby sme mali jednotlivé roviny vysegmentované zvlášť. Preto je potrebné ešte vysegmentovať oblak bodov viac a oddeliť od seba rovnobežné roviny v 3D scéne.

4.3.2. Návrh

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

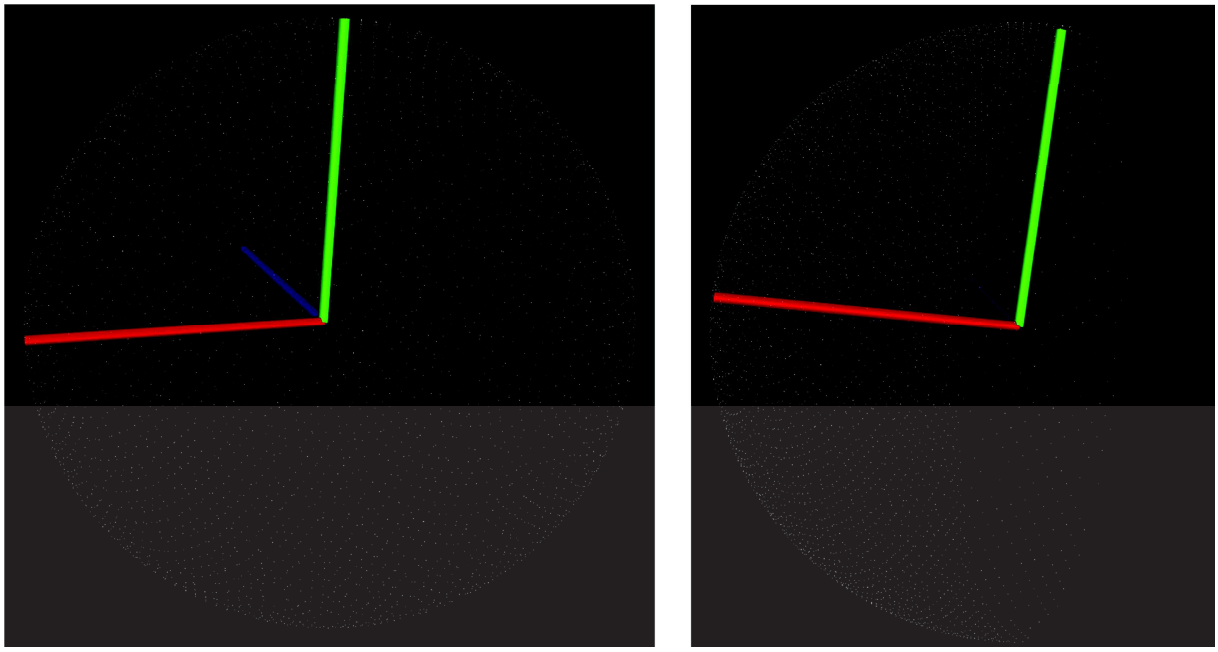
Keďže PCL neumožňuje vytvárať histogram, ktorý je potrebný pre našu úlohu, navrhli sme si vlastný. Vytvorili sme si 3D histogram normál, ktorý je v podstate oblak bodov vykresľujúci jednotkovú guľu. Táto jednotková guľa je vykresľovaná bodmi podľa normál bodov vstupného oblaku bodov. Keďže v našom prípade inverzné body berieme ako rovnaké, všetky normály so zápornou x-ovou normálou premietneme do opačného smeru čím nám vznikne iba pól guľa (viď. Obrázok 6).



Obrázok 6 3D histogram normál.

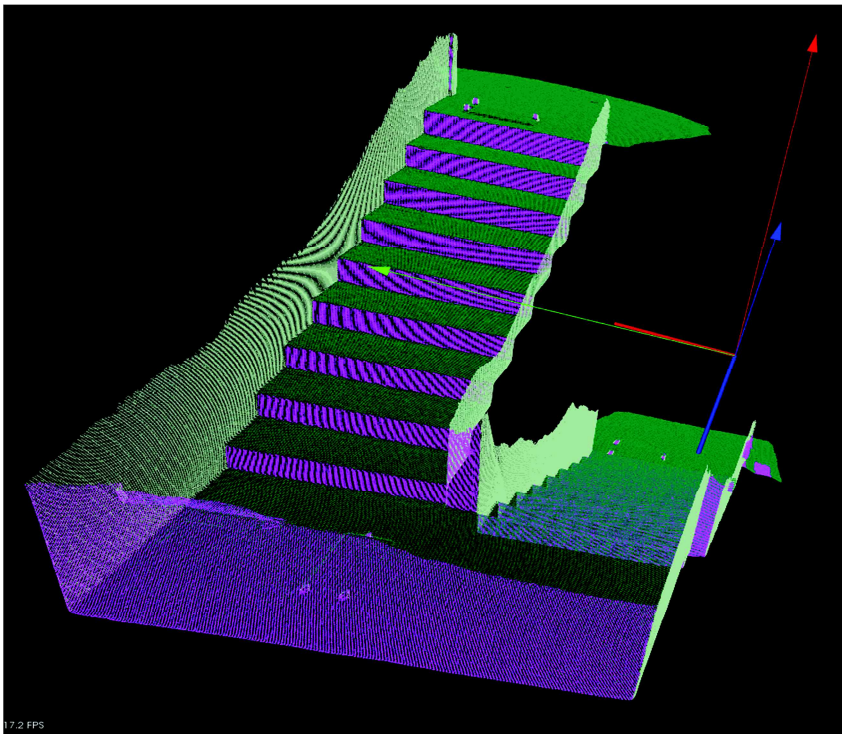
Hľadanie dominantných smerov sme riešili hľadaním najhustejších bodov v 3D histograme normál. Spôsobom, že sme pre každý bod zistili hustotu jeho okolia a hľadali sme bod (normálu) s najhustejším okolím, ktorý sme následne označili ako dominantný smer. Následne

sme okolie tohto bodu z histogramu normál odstránili (aby sme nenašli dva krát ten istý bod) a znovu sme hľadali bod s najhustejším okolím (ďalší dominantný smer). Toto sme opakovali až kým sme nenašli požadovaný počet dominantných smerov. Keďže v samotnom vstupnom oblaku bodov môže byť aj vyše milión bodov, čiže aj vo vytvorenom histograme normál môže byť premietnutých vyše milión bodov, ktoré môžu byť často skoro na úplne rovnakej pozície, nemá zmysel zisťovať hustotu okolia každého bodu (nebolo by to efektívne). Kvôli tomu, sme sa rozhodli dané vyhľadávanie dominantných smerov optimalizovať a to nasledovne. Vytvorili sme si oblak bodov so zadaným počtom bodov (presnosťou) rovnomerne rozložených po povrchu jednotkovej gule (viď Obrázok 7 vľavo) a pri hľadaní dominantných smerov sme prechádzali len bodmi tohto oblaku bodov. Čím bol počet bodov v tomto oblaku bodov väčší, tým bola presnosť hľadania dominantných smerov väčšia, ale zároveň aj časovo náročnejšia. Avšak, keďže v našom prípade inverzné normály (smery) berieme ako rovnaké, stačí nám vytvoriť iba pól guľu (viď Obrázok 7 vpravo). A pri počítaní hustoty okolia bodu budeme pripočítavať aj hustotu okolia jeho inverzného bodu.



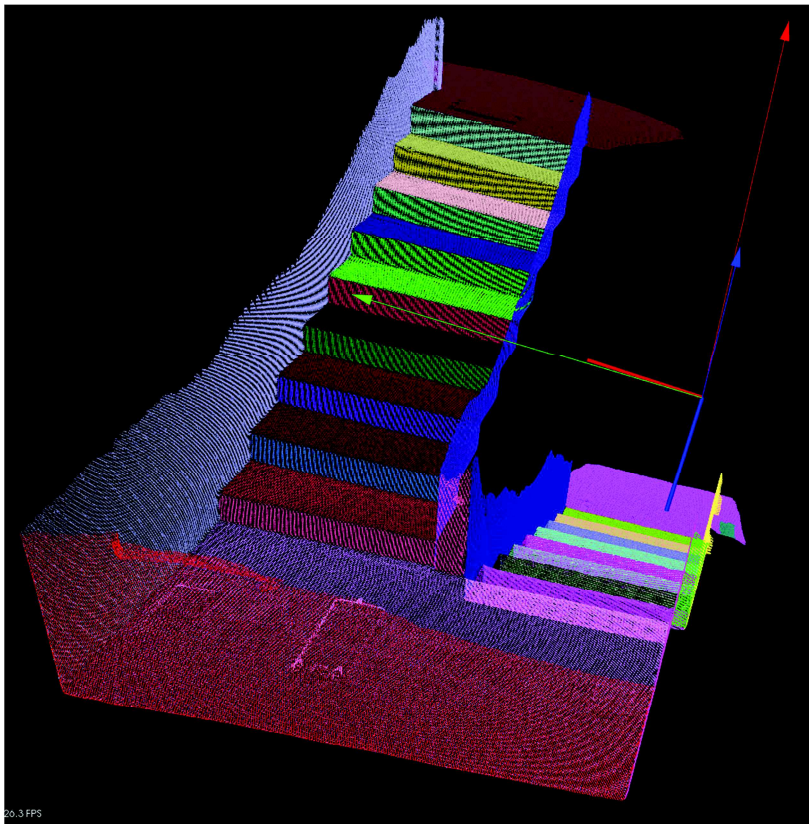
Obrázok 7 Oblak bodov v tvare gule (vľavo) a oblak bodov v tvare pól gule (vpravo).

Po získaní dominantných smerov oblaku bodov, pridelíme každý bod k jednému dominantnému smeru, ku ktorému je normála bodu najbližšie. Všetky body sa zároveň farebne označia podľa toho, ku ktorému dominantnému smeru sa pridelili (každý dominantný smer inou farbou). Na Obrázku 8 môžete vidieť ukážkové ofarbenie bodov podľa dominantných smerov.



Obrázok 8 Ofarbenie bodov podľa dominantných smerov.

Po označení jednotlivých bodov podľa dominantných smerov oblaku bodov je ešte potrebné odlíšiť body patriace k rôznym objektom (jednotlivé steny, dvere, skrine, atď.). Čiastočne nám to už označovanie podľa dominantných smerov spravilo. Avšak, segmenty objektov ku sebe rovnobežné je ešte stále potrebné od seba odlíšiť. Tento problém sme riešili pomocou algoritmu rastúceho regiónu, ktorý jednak bral do úvahy pri segmentovaní priestorovú vzdialenosť (susednosť bodov) a jednak farbu bodov ako hraničný parameter. PCL knižnica poskytuje implementáciu algoritmu rastúceho regiónu, ktorý sme aj použili. Ukážkový výsledok segmentácie je možné vidieť na Obrázku 9.



Obrázok 9 Výsledná segmentácia schodov.

4.3.3. Implementácia

Implementácia prebehla v jazyku C++ pomocou knižnice PCL v programátorskom prostredí Microsoft Visual Studio 2013 C++. Vstupom pre tento modul je oblak bodov 3D scény a výstupom sú segmenty bodov reprezentujúce objekty v 3D scéne (steny, dvere, skrine, posteľ, atď.).

4.3.4. Testovanie

Testovanie prebieha súbežne pre všetky kroky modulu. Prevažne sa jedná o manuálne vizuálne testovanie pomocou vizualizácie knižnice PCL a následne, v prípade nájdenia chyby počas vizualizácie, ladenie zdrojového kódu cez použité programátorské prostredie. 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 kocky. Jednotlivé kroky modulu najprv testujeme na týchto syntetických vstupných dátach a následne až na reálnych dátach.

4.4. Vizualizácia

4.4.1. Analýza

Dôležitým prvkom nášho nástroja je modul vizualizácie, ktorý slúži 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 oblak bodov tvoriaci strop miestnosti prekladá podľa očakávaní správne zostrojeným mnohouholníkom.

Úlohou tohto modulu nie je len staticky vizualizovať, ale aj umožniť interaktívny pohyb scénou, vďaka ktorému je možné detailne sledovať objekty z ľubovoľného uhlu pohľadu.

4.4.2. Návrh

Vzhľadom na skutočnosť že pre vývoj nášho rekonštrukčného nástroja používame knižnicu PCL, pre jednoduchú integráciu sme sa rozhodli na vizualizáciu využiť triedu PCLVisualizer, ktorá je súčasťou tejto knižnice.

Návrh nášho modulu vizualizácie sa preto odvíja od návrhu tejto triedy.

Trieda PCLVisualizer je veľmi komplexná a okrem mnohých iných poskytuje metódy pre vizualizáciu oblakov bodov, mnohouholníkových sietí, alebo jednoduchých geometrických útvarov ako guľa, kváder, kocka.

O interakciu používateľa so scénou sa vo východzom stave stará objekt triedy PCLVisualizerInteractorStyle, ktorý definuje priradenie udalostí zo vstupných zariadení (napr. kliknutie a pohyb myšou) konkrétnym akciám, napr. rotácii kamery.

Východzí spôsob interakcie definovaný touto triedou sme vyhodnotili ako nevyhovujúci a preto sme sa zameriavali na jeho vylepšenie.

Medzi hlavné nedostatky patrili:

- pohyb kamery po scéne nemožno ovládať klávesnicou, ale iba stlačením kolieska myši a súčasným pohybom myši
- 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.

Tieto problémy boli odstránené implementáciou vlastnej triedy interakcie CustomInteractorStyle, ktorá bola odvodená od spomínanej PCLVisualizerInteractorStyle. V nej sú všetky nevyhovujúce metódy prekonané vlastnou implementáciou. Tým pádom sa zachovala štruktúra metód pôvodnej triedy a zmenila sa iba ich implementácia. Pre použitie tohto vylepšeného štýlu interakcie je potrebné vytvoriť objekt triedy CustomInteractorStyle a použiť ho ako parameter konštruktora pri vytváraní objektu PCLVisualizer.

4.4.3. 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 poľom troch súradníc x,y,z. Bod otáčania kamery je určený zase premennou focal (focal point), ktorá je podobne poľom súradníc v priestore. 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 pozície kamery, čím sa dosiahol rovnaký efekt otáčania ako požadovaný a aj vektor pohľadu zostal nenulový.

Uzamknutie rotácie 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 (Visualuzation 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 doľava 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 zadefinované 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.

Interakcia pomocou 3D myši 3Dconnexion SpaceNavigator

Vizualizačná trieda PCLVisualizer je naimplementovaná pomocou prostriedkov knižnice VTK. Tá vo východzom stave podporuje interakciu iba pomocou štandardných vstupných zariadení. Existuje možnosť jej prekompilovania tak, aby podporovala prijímanie udalostí aj zo vstupných zariadení výrobcu 3Dconnexion. Potom by bolo potrebné nainštalovať softvérovú vývojovú sadu pre 3D myš, aby bolo možné spracovávať vstup z 3D myši vo forme udalostí (napr. TdxButtonPressEvent) ktoré knižnica generuje pri manipulácii s myšou.

Knižnica VTK však neobsahuje žiadnu interakčnú triedu, ktorá by tieto udalosti interpretovala na skutočný pohyb kamery v scéne. Pre každý typ udalosti generovaný myšou by bolo teda nutné naimplementovať funkciu, ktorá by na základe parametrov pohybu myšou vykonávala adekvátny pohyb kamery v scéne. Následne by stačilo pomocou metódy addObserver() každú z týchto funkcií namapovať na príslušnú udalosť generovanú myšou.

Tento spôsob interakcie nakoniec nebol zrealizovaný, pretože sa objavila možnosť použiť 3D myš v existujúcom nástroji AutoCAD, ktorý má podporu pre ňu zabudovanú.

4.5. Export rekonštruovaných objektov

4.5.1. Analýza

V našom nástroji bol potrebný modul, ktorý by umožnil export rekonštruovaných útvarov do CAD formátu DXF (Drawing eXchange Format), s ktorým je možné ďalej pracovať vo väčšine modelovacích nástrojov, ako napríklad AutoCAD, Draftsight, či Rhinoceros.

Tento modul je realizovaný triedou Exporter. Poskytuje elementárne funkcie pre zápis objektov do virtuálnej scény, ktorej obsah je možné exportovať do súboru.

Keďže výstupom rekonštrukčných metód sú geometrické útvary reprezentované objektami knižnice PCL, trieda Exporter musí v prvom rade zabezpečiť konverziu týchto útvarov do formátu exportovateľného použitou knižnicou sgCore.

4.5.2. Návrh

Modul exportu je navrhnutý ako samostatná trieda Exporter, ktorá poskytuje tri metódy:

bool addWalls(std::vector<r3d::obj::Wall> walls)

- táto metóda slúži na export planárnych objektov. Jej parametrom je vektor objektov typu Wall, ktoré reprezentujú požadovaný objekt.

bool addMesh(pcl::PolygonMesh mesh)

- pomocou tejto metódy je možné uložiť všetky objekty reprezentované sieťou mnohouholníkov, ktoré získame napríklad po triangulácii oblaku bodov. Použitie tejto metódy má zmysel vtedy, keď nedokážeme nejaký objekt reprezentovať žiadnym elementárnym geometrickým útvarom.

bool writeDXF(const char filename)*

- táto metoda prečíta všetky objekty vo virtuálnej scéne vytvorené pomocou metód addWalls() a addMesh() a uloží ich do súboru s požadovaným názvom.

4.5.3. Riešenie

Trieda Exporter bola naimplementovaná pomocou knižnice sgCore, ktorá umožňuje vytváranie modelov a ich export na disk.

Hlavnou úlohou bolo teda vyriešiť konverziu z formátov pre opis objektov používaných v našom nástroji do formátov knižnice sgCore.

V prípade metódy addWalls() bolo nutné preiterovať cez každý rohový bod steny typu pcl::PointXYZ a zostrojiť z nich kontúru mnohouholníka (sgCContour) spájaním vždy dvoch nasledujúcich bodov úsečkou. V prípade správneho vstupu vznikla po prejdení všetkých bodov uzavretá lomená čiara tvoriaca obvod mnohouholníka. Na záver stačilo tejto kontúre vytvoriť výplň vo forme objektu sgSurfaces::Face.

Metóda addMesh() exportuje sieť mnohouholníkov vo formáte pcl::PolygonMesh, pričom tento formát knižnice PCL pozostáva z vektora bodov (body mnohouholníkov) a vektora spojníc týchto bodov, ktorý reprezentuje hrany mnohouholníkov. Každý objekt tohto typu je teda tvorený veľkým množstvom napr. malých trojuholníkov.

Knižnica sgCore našťastie poskytuje na vytvorenie siete trojuholníkov metódu ObjectFromTriangles. Pre jej aplikovanie bolo nutné prekonvertovať všetky body do formátu SG_POINT a všetky ich spojnice do formátu SG_INDEX_TRIANGLE.

Zápis takto prekonvertovaných objektov sme zrealizovali pomocou funkcie ExportDXF().

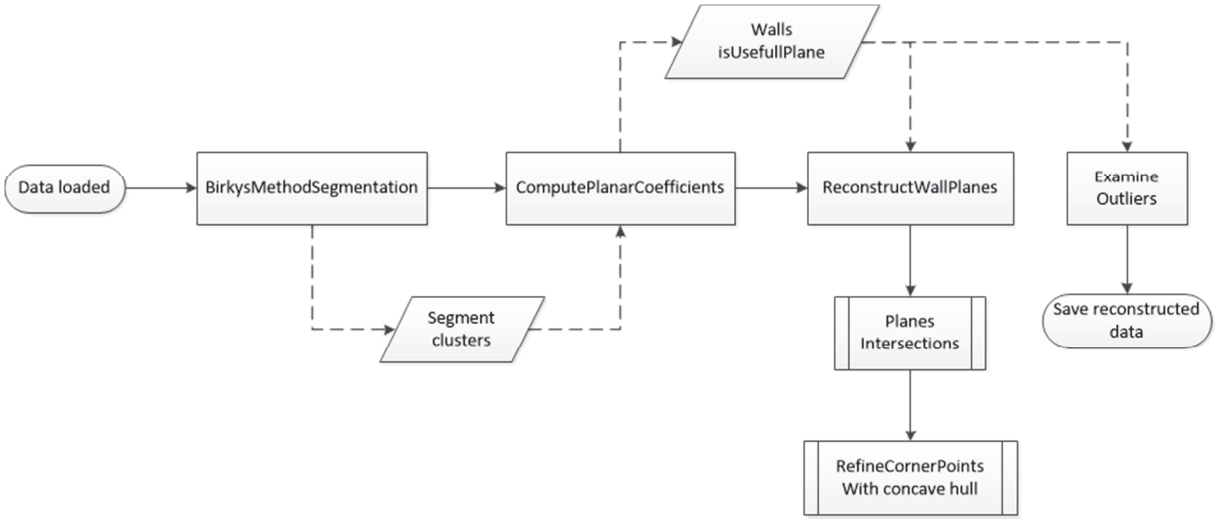
4.6. Konfigurácia a vstupné parametre z XML

Pre jednoduchšiu použiteľnosť bol naprogramovaný konfigurátor, pomocou ktorého je jednoducho možné nastaviť parametre metód rýchlo a bez nutnosti prekompilovania celého projektu. Parametre sú uložené v súbore formátu XML a pre spracovanie konfiguračných súborov využívame knižnicu TinyXML, ktorá je určená pre spracovávanie XML súborov, získavanie a rozdeľovanie hodnôt v nich zapísaných.

Knižnica pozostáva celkovo štyroch zdrojových súborov a dvoch hlavičkových, ktoré sú umiestnené v DataHandlingu. Otvorenie konfiguračného súboru je realizované pomocou open file dialogu v GUI. Ak sa nevyužije manuálne zvolený XML konfiguračný súbor, v tom prípade sa použije pevne stanovený so štandardnými hodnotami. Trieda Reconstruction implementuje metódu getParameter, ktorá dostáva ako parameterTiXmlNode, ukazovateľ na spracovaný a otvorený konfiguračný súbor a identifikátor typu const char* name, pre identifikovanie hodnoty s konfiguračného súboru. Spomenutá metóda getParameter využíva metódy samotnej triedy getFirstChildElemen, ktorá na základe parametra const char* identifikuje samotnú hodnotu v XML súbore a metódou getText vráti jej textovú podobu vo formáte const char*, ktorá je konvertovaná do int alebo float premennej pre nastavenie samotných parametrov metód. Samotné celkové nastavenie parametrov metód je zapuzdrené v metóde setConfiguration definovanej v triede Reconstruction. Metóda je prekonaná vo všetkých dcérskych triedach a parametrom je ukazovateľ TiXmlNode, ktorý ukazuje na miesto v konfiguračnom súbore, kde sa začína časť dát pre konkrétnu metódu.

4.7. Zmeny v spracovaní po letnom semestri

Hlavné zmeny, ku ktorým vo vývoji došlo je celková zmena vývoja v oblasti segmentačných a rekonštrukčných metód. Pozastavil sa vývoj troch osobitných metód. Pokračovalo sa v metóde 4 zo zimného semestra, ktorej podstata v knižnici sa z rekonštrukčnej zmenila na segmentačnú. Jej implementácia ostala v triede BirkisMethodReconstruction. Ďalšia implementačná rekonštrukčná logika sa implementovala do triedy GrowingRegionReconstrucion. Je pochopiteľné, že tieto názvy už úplne neodpovedajú podstate logiky toho čo obsahujú. Preto sa na celý projekt chystá masívny refaktoring. Momentálny tok dát a implementácia rekonštrukcie od nadviazania na metódu 4 je opísaná v tejto časti dokumentu (obrázok 10).



Obrázok 10 Vývojový diagram

4.7.1. Predspracovanie dátovej sady

Predspracovanie dát bolo riešené odstránením šumu s point cloudu. Pre tento účel bola vytvorená trieda NoiseRemover. Trieda obsahuje dôležitú metódu removeNoise(point cloud), ktorá dostáva vstupný parameter dáta point cloud a na výstupe vracia odfiltrovaný point cloud dát. Metóda odstránenia šumu využíva triedy knižnice PCL, StatisticalOutlierRemoval. Pre vykonanie odstránenia šumu point cloudu, sú dôležité dva parametre, počet najbližších susedov vzhľadom na skúmaný bod a odchýlka vzdialenosti všetkých susedov, ktorých berieme do úvahy, definovanej počtom susedov, vzhľadom na skúmaný bod.

Tie susedné body, ktoré majú väčšiu vzdialenosť, ako stanovená maximálna odchýlka, sú odstránené. Výsledkom je point cloud s redukovaným šumom. Výsledný dataset sa dá uložiť na disk, pre opätovné testovania a použitia bez nutnosti opakovania procesu odstraňovania šumu.

4.7.2. 3D rekonštrukcia

Prechodom k novému vývoju sa definovali nové elementárne kroky, ktoré sú zaznamenané vo vývojovom diagrame. Prvým krokom ostalo načítanie dát. Tieto dáta sa uložia vo formáte PointXYZRGBNormal, či už obsahujú všetky informácie alebo nie. Pri vyvolaní rekonštrukčnej metódy, sa dáta prekopírujú do jednotlivých oblakov, podľa typu dát, ktorý udržiavajú.

4.7.3. Segmentácia

Prvým krokom je segmentácia súvislých rovinných oblastí. Tento krok je vykonaný spomínanou metódou č 2. s volaním triedy BirkysMethodReconstruction. Výstupom tejto metódy je ofarbený oblak bodov a vektor klastrov. Farba bodov v oblaku oddeľuje body jednotlivých segmentov. Vektor klastrov obsahuje vektory indexov bodov z pôvodného oblaku bodov, ktoré spadajú do jednotlivých klastrov.

4.7.4. Výpočet koeficientov roviny

V ďalšom kroku sa pomocou metódy RANSAC s modelom pre rovinu určia koeficienty rovín každého segmentu. Táto funkcionalita je implementovaná v metóde computePlanarCoefficients. Vstupné parametre pre RANSAC boli určené experimentálne a sú načítavané z externého konfiguračného súboru. Po určení koeficientov všeobecnej rovnice roviny sa vytvorí objekt steny. Ako prvé sa priradí nový objekt polygón (Polygon), ktorý reprezentuje abstraktné ohraničenie segmentu. Do polygónu sa tiež uložia vypočítané koeficienty, ktoré patria spracovávanému segmentu. Ďalšie čo sa priradí inštancií objektu Wall je oblak bodov segmentu. Posledné dva kroky sú výpočet štatistiky - smerodajnej odchýlky presnosti merania senzora a overenie segmentu, či reálne predstavuje stenu. Druhý krok je potrebný, pretože RANSAC nesleduje či je segment reálne rovina. On len vypočíta analytické vyjadrenie a napasuje rovinu cez 4 body v rovine a štatisticky s pravdepodobnosťou určuje, či rovina pokrýva určené percento bodov. Určenie či je segment rovina je následne potrebné pre ďalšie kroky rekonštrukcie a hľadanie ohraničenia segmentov.

4.7.5. Verifikácia validnosti roviny

Pre rekonštrukciu rovín zo segmentu oblaku bodov je potrebné najprv tieto segmenty verifikovať, či ide o rovinný objekt alebo o nejaký iný. Verifikáciu validnosti rovín robíme na základe pomeru outliers a inliers, určením ideálneho pomeru by sa dalo ešte pohrať, ale vcelku dobré výsledky dáva 2:1 (inliers:outliers). To znamená, že ak počet inliers je dvojnásobne väčší ako počet outliers, tak sa jedná o rovina, v opačnom prípade sa o rovinu nejedná. Jednotlivé inliers a outliers daného segmentu vypočítavame pomocou vzdialenosti od preloženej roviny RANSAC-om cez daný segment. Vzdialenosť berieme ako štandardnú odchýlku zo segmentu, o ktorom vieme, že je validná rovina. Predpokladáme, že segment s najväčším počtom bodov je rovina (stena).

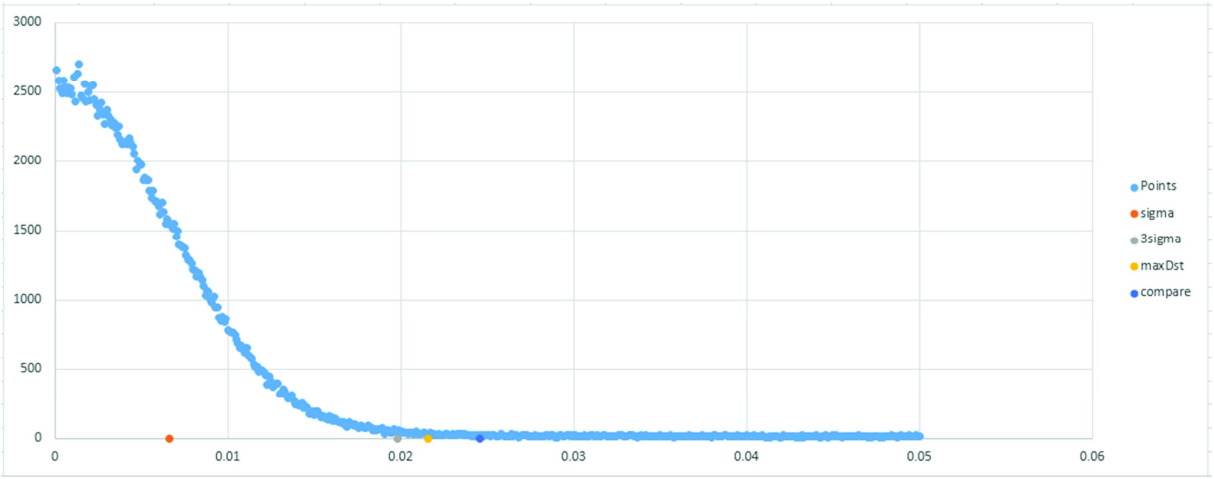
4.7.6. Výpočet štatistík pre segmenty

Pre úspešnú a najmä automatickú segmentáciu je potrebné poznať charakteristické vlastnosti segmentu. Pre tento účel nám slúžili dve hodnoty. Prvá bola štandardná odchýlka, ktorá určuje chybu merania senzora. Vďaka tejto hodnote vieme určiť hrúbku segmentu - maximálnu vzdialenosť bodov, ktoré patria stene/rovine a body, ktoré predstavujú objekty priľahlé tejto stene. Tiež je pomocou nej možné určiť či je segment vôbec rovinou alebo nie. Druhá hodnota je rozptyl bodov v segmente. Rozptyl predstavuje rovinné rozloženie bodov - vzájomnú priemernú vzdialenosť medzi jednotlivými bodmi segmentu.

Štandardná odchýlka segmentu

Výpočet štandardnej odchýlky sa určuje pomocou koeficientov roviny, ktorú určí RANSAC. Pomocou nim pomocou metódy getDistancesToModel, ktorá vráti vzdialenosti všetkých bodov k RANSAC modelu roviny, sa vypočíta histogram vzdialeností. Keďže body sú roztrúsené v celom priestore a rozdiely medzi vzdialenosťami sa pohybuje okolo štyroch stotisícin vzdialenosti mriežky, histogram sa ráta pre konštantne zadané rezy vzdialenosťami. Pre určenie smerodajnej odchýlky, ale potrebujeme vedieť počet bodov, ktoré reálne patria rovine. Logické zváženie - ak je rovina, najviac bodov bude ležať okolo vzdialenosti 0 a čím ďalej pôjdeme, tým týchto bodov bude menej. Ak začnú narastať, toto miesto už predstavuje

oblasť priľahlého objektu. Miesto ohraničenia sa predpokladá oblasť, kde počet bodov klesá tak, že začne byť konštantný - začne konvergovať. V histograme sa teda hľadá konvergencia počtu bodov od vzdialenosti. Ak medzi dvoma vzdialenosťami, ktorých vzdialenosť je zadaná hodnota m_lineCheckoutStep je uhol menší ako zadaná hodnota m_AngularCoefficientMax (3°) a táto udalosť sa opakuje za sebou m_linearRepetitions krát, tak sa prvá z posledného opakovania určí ako ohraničenie. Následne sa už len štatisticky určí smerodajná odchýlka. Na histograme je smerodajná odchýlka označená ako sigma a vzdialenosť ohraničenia konvergencie je maxDst a hodnota s ktorou sa porovnával compare.



Obrázok 11 Histogram vzdialeností bodov od roviny (os X- vzdialenosť, os Y- počet bodov v danej vzdialenosti)

Rozptyl bodov segmentu

Pre výpočet rozptylu sa využíva metóda KNN - K-najbližších bodov. Pre tento zámer bola využitá pcl implementácia z KdTreeFLANN. Výpočet prebieha nasledovne.

- Vyber N (100) počiatočných bodov
- Každému z vybraných bodov nájdi K (100) najbližších bodov
- Vypočítaj priemernú vzdialenosť k týmto bodom
- Vypočítaj priemernú hodnotu z vypočítaných priemerných vzdialeností
- Pre odstránenie chyby a náhodného slabého výberu počiatočných bodov → rozptyl * 1,03

4.7.7. Rekonštrukcia oblastí plôch - Hľadanie ohraničenia

Pre abstrakciu je potrebné nájsť ohraničenie bodov segmentu. V našom prípade sa riešilo iba ohraničenie rovinného segmentu. Všetky ostatné prípady sa ponechali ako Future work. Ohraničením segmentu dostaneme polygonálne geometrické vyjadrenie primitívy predstavujúcej časť objektu. V jednoduchšom prípade sa jedná len o stenu, kde jedna rovina predstavuje celý objekt reálneho sveta. Iné komplexné priestorové objekty je potrebné ešte spojiť podľa vzdialeností a iných charakteristických vlastností - tento problém už ale naša práca nerieši.

Vďaka určeniu validných rovín akákoľvek ďalšia operácia s rovinami zahŕňa už len tie, ktoré sú označené ako “isUsefulPlane = True”.

Priesečníky rovín a verifikácia priesečníka

V komplexných scénach dochádza štandardne k pretínaniu rovín. Preto sme tento jav využili pre náš prospech a body, kde sa roviny pretínajú označili ako body ohraničení polygónov segmentov, ktorých roviny sa v danom bode pretínajú.

Proces hľadania priesečníkov spočíva v prehliadaní kombinácií verifikovaných rovín. Každej trojici rovín sa určí priesečník (threePlanesIntersection).

Lenže v priestore, v nejakom bode sa pretínajú všetky trojice rovín okrem rovnobežných. Preto je potrebné overiť pozíciu priesečníka voči bodom segmentov. Toto sa vykoná pre každý segment pomocou hľadania najbližších bodov vo vzdialenosti m_CornerRadiusMultiplicator * disperzia bodov segmentu (KdTreeFLANN). Pokiaľ sa nájde aspoň jeden bod v tejto vzdialenosti, priesečník sa považuje za validný a je priradený každej stene. Pokiaľ sa nenájde bod aspoň jedného segmentu, tak sa bod za validný priesečník nepovažuje.

Takýmto spôsobom je jednoduché a veľmi úspešné nájsť priesečníky segmentov/stien, ktoré sú ohraničené zo všetkých strán. V prípade, že tak nie je, je potrebné využiť ďalšie metodiky hľadania bodov ohraničenia.

Verifikácia ohraničenia

Pre efektívne hľadania ohraničenia len segmentom, ktoré ho ešte nemajú úplné sme implementovali funkciu verifikácie polygonálneho ohraničenia.

Ohraničenie segmentu zistené v predchádzajúcom kroku je potrebné ešte dodatočne verifikovať. Za týmto účelom sme naimplementovali metódu, ktorá na základe parametrov určí správnosť ohraničenia segmentu.

Jej parametrami sú:

- segment steny v podobe oblaku bodov,
- vektor bodov predstavujúci mnohouholník ohraničujúci segment steny
- a maximálna odchýlka bodov segmentu od ohraničenia

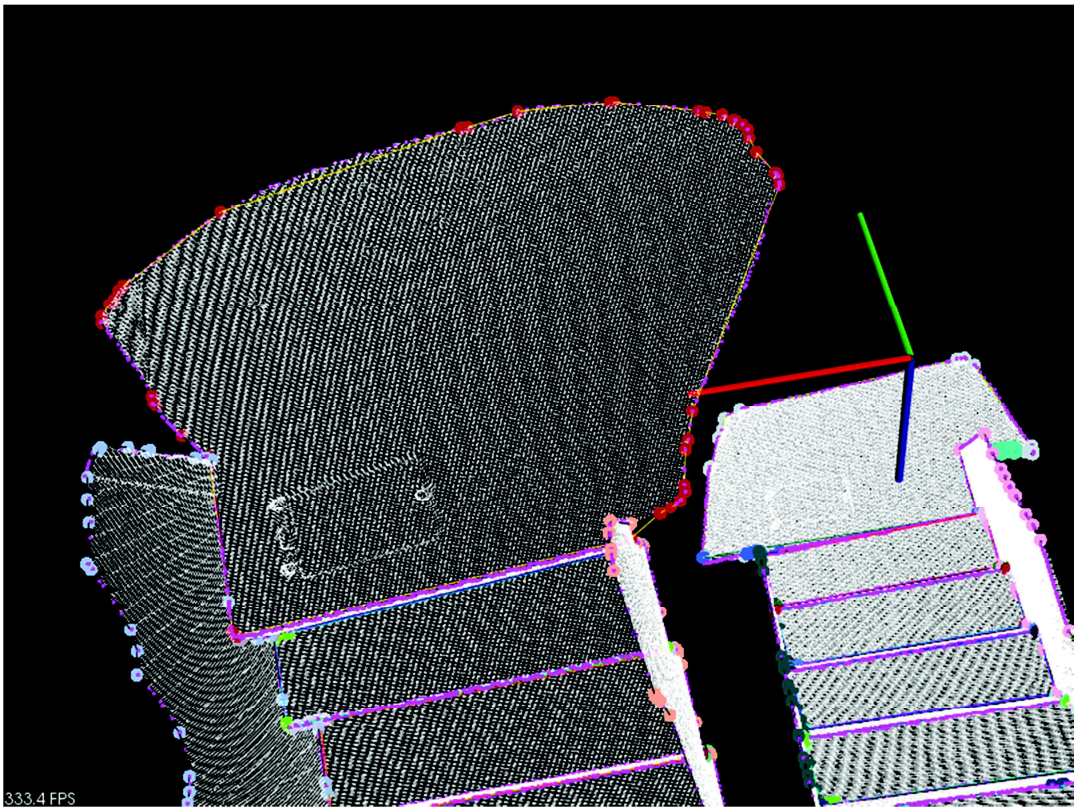
Funkcia vykonáva tento algoritmus:

- vypočíta sa konkávna obálka segmentu bodov.
- vypočíta sa vzdialenosť každého bodu obálky od najbližšej úsečky tvoriacej ohraničenie. Ak táto vzdialenosť presahuje medznú hodnotu

Ak je počet bodov presahujúcich medznú vzdialenosť väčší než 5%, je ohraničenie označené za nesprávne, inak je správne.

Zlepšenie-dokončenie ohraničenia

Je implementované v metóde refineCorners. Najjednoduchší spôsob ako nájsť ohraničenie segmentu je určiť jeho konkávnu obálku. Problém je, že ak by sme chceli brať polygón z konávnej obálky bez ďalšieho spracovania, aj keď pravidelný obdĺžnik by mohol mať viac ako 800bodov ohraničenia. Preto je potrebné tieto body najprv prefiltrovať. Aby sme sa k tomu dostali je potrebné najprv PCL konkávnu obálku (ConcaveHull) usporiadať tak, aby body ako idú za sebou vo vektore, išli za sebou aj po obvode obálky. Pre tento účel sme implementovali funkciu sortSegmentHull.



Obrázok 12 Odhad hraníc plochy

SortSegmentHull

V skratke funguje ako vlak, ktorý ide od bodu k bodu a prejde celú obálku.

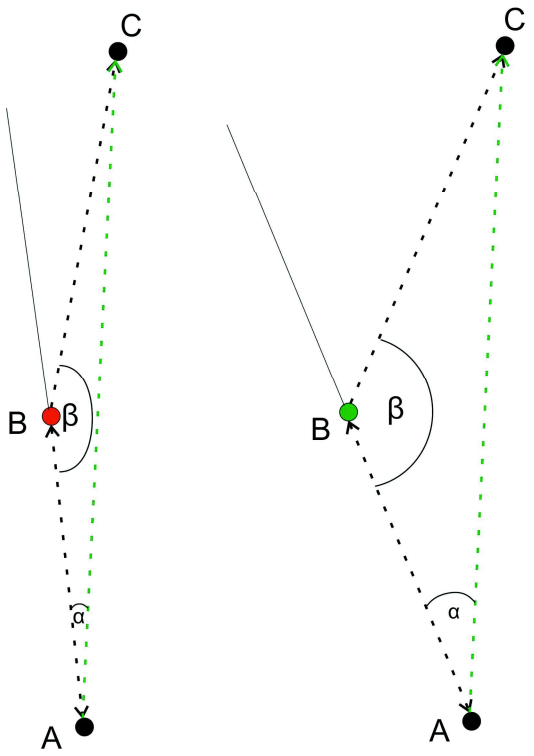
- Vyber ako počiatočný bod a odstráň prvý bod z oblaku bodov
- Pomocou KNN nájdi jeho najbližšieho suseda
- Odstráň suseda z oblaku bodov a sprav z neho ďalší počiatočný bod
- Kroky 2 a 3 opakuj pre všetky body, kým pôvodný oblak nie je prázdny
- Body v poradí ako boli nájdené priraď do nového oblaku bodov.

Ďalším krokom bolo samotné filtrovanie. Ak už segment mal nájdený aspoň jeden hraničný bod, našiel sa k nemu najbližší bod obálky a určil sa ako počiatočný, inak sa vybral prvý bod. Implementácia je vo funkcií filterHullPoints.

filterHullPoints

Proces filtrovania spočíval v predpoklade, že nie je potrebné mať v ohraničení body, ktoré ležia na jednej úsečke, ale sú postačujúce jej počiatočné body. Toto sa dosiahlo jednoduchou aproximáciou úsečky bodov podľa vzájomného uhla vektorov, ktoré tri za sebou idúce body zvierali. Ukážka uhlov alfa a beta je na obrázku. červenou je zvýraznený bod, ktorý ležal v intervale uhla, ktorý určoval smer priamky, uhol v zelenom ho už presahuje. Intervaly uhlov - alfa $|BAC| < 10°$, beta $|ABC| > 185°$.

Takýmto postupom sa podarilo úspešne filtrovať nadbytočné body ohraničujúcej obálky ako je znázornené na obrázku 13.



Obrázok 13 Filtrovanie bodov ohraničujúcej obálky

Záverečné úpravy

Posledné úpravy pre spresnenie ohraničenia pasujú body polygónu späť na segment a tie potom kolmo premietajú na vypočítanú rovinu. Takto sa určí presné ohraničenie a nájde sa aj hľadaný rovinný polygón, ktorý predstavuje ohraničenie segmentu. Pasovanie je implementované vo funkcii fitHullToSegmentPoints.

fitHullToSegmentPoints

Telo a kroky funkcie sú jednoduché. Pre každý bod určeného obvodu polygónu sa hľadá pomocou triedy KdTreeFLANN a jej metódy nearestKSearch najbližší bod segmentu, ktorý sa určí ako bod reálny ohraničenia.

Druhá funkcia kolmého priemetu nie je implementovaná. Jej podstatou je nájsť kolmé priemety bodov reálneho ohraničenia na analyticky určenú rovinu prechádzajúcu segmentom.

4.7.8. Identifikácia outliers

Identifikácia outliers sa robí pre každý segment oblaku bodov zvlášť. Pri výpočte sa využívajú vypočítané štatistiky jednotlivých segmentov. Konkrétne sa jedná o koeficienty roviny získané RANSAC-om a štandardnú odchýlku segmentu. Outliers sa následne získavajú na základe vzdialenosti bodov od roviny vypočítanej RANSAC-om. Každý bod, ktorý je od vypočítanej roviny vzdialenejšie ako 4.5 násobok štandardnej odchýlky segmentu sa berie ako outlier.

Inštalačná príručka

Dependencies

- Visual Studio 2013
- PCL 1.7.2 and external
- TinyXML 2.6.2

Summary of set up

- Install all required dependencies.
- Create enviroment variables:
    - OPENCV_ROOT
    - SGCORE_ROOT
    - PCL_ROOT
- Set path in Configuration
    - properties->Debugging->Enviroment
    - PATH=$(OPENCV_ROOT)\x64\vc12\bin;$(PCL_ROOT)\3rdParty\VTK\bin;$(PCL_ROOT)\3rdParty\Qhull\bin;$(PCL_ROOT)\3rdParty\FLANN\bin;$(PCL_ROOT)\3rdParty\Boost\bin;$(PCL_ROOT)\bin;$(SGCORE_ROOT);
- Set 'gui' as a StartUp project

Literatúra

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

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Tímový projekt

Rekonštrukcia 3D scény

Projektová dokumentácia – riadenie

Vedúci projektu: Ing. Vanda Benešová, CSc.

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. máj 2016

Obsah

1. Úvod

V tomto dokumente sú opísané postupy, plány, a metódy riadenia tímového projektu stanovené naším tímom pre vývoj prototypu desktopovej aplikácie rekonštrukcie 3D scény z 3D dát. Charakter projektu je výskumne orientovaný, preto aj jeho hlavným cieľom je vytvoriť aplikáciu, ktorá bude spĺňať predpoklady testovacieho prototypu rôznych metód rekonštrukcie a segmentácie objektov. Projekt má predpoklad byť vývojovo a zdrojovo náročný, rovnako ako jeho prototyp náročný výpočtovo a pamäťovo. Preto je potrebné vytvoriť a dodržiavať predpísané pravidlá, aby sa možné komplikácie eliminovali skôr akoby mali skomplikovať ďalší vývoj. Vzhľadom na charakter projektu bolo potrebné upraviť aj metodiky riadenia, plánovanie vývoja a rozdelenie úloh na funkcionalite, aby na ich základe vznikol požadovaný produkt.

Projekt 3DRecon je vyvíjaný pre výskumné centrum Joanneum v Grazi, aj preto nemá charakter klasického produktu pre koncového používateľa, ale skôr výskumne orientovaného prototypu, ktorý môže slúžiť ako základná knižnica pre ďalší vývoj v oblasti rekonštrukcie či segmentácie 3D dát.

1.1. Prehľad dokumentu

V časti 2 sú opísané manažérske činnosti pridelené jednotlivým členom tímu aj s opisom príslušnej zodpovednosti. V časti 3 sú opísané realizované manažmenty v rámci nášho tímového projektu. V kapitole 4 je prehľad používaných metodík s krátkym opisom, celé znenie jednotlivých metodík sa nachádza v prílohe. Retrospektívne zhodnotenie, jednotlivo pre zimný a letný semester, obsahuje kapitola 5.

Prílohy:

- Kompetencie členov tímu
- Metodiky
- Exporty úloh z nástroja TFS na jednotlivé šprinty

2. Role členov tímu

2.1. Manažérske činnosti

meno	rola	zodpovednosť
Lukáš Hudec	team leader, manažér architektúry	návrh a implementácia architektúry, správa tímu a rozdelenie úloh, evi dencia úloh v TFS, segmentácia
Róbert Birkus	manažér vývoja	vývoj hlavnej aplikačnej logiky, správa implementovaných metód, starostlivosť nad dodržiavaním konvencií
Martin Jurík	projektový integrátor	prepojenie modulov, kontrola funkčnosti, implementácia modulov
Michal Löffler	biznis manažér	analýza požiadaviek, návrh a rozdelenie úloh, komunikácia so zadávateľom, analýza rizík
Róbert Karásek	test manažér	tvorba testov, vykonanie testov, dohľad nad “definition of done”
Michal Korbeľ	manažér propagácie	správa webovej stránky
Katarína Janečková	manažér dokumentácie	udržiavanie dokumentácie a zápisníc zo stretnutí

Tabuľka 1 Rozdelenie úloh v tíme

3. Aplikácie manažmentov

3.1. Manažment komunikácie a ľudských zdrojov

Oficiálne stretnutia tímu prebiehali každý týždeň vo štvrtok o 10:00. Ich témou bolo zhodnotenie vykonanej práce za predchádzajúci týždeň a naplánovanie ďalšej. Po stretnutí prebiehala diskusia o spoločnej implementácií, o dokumentácií, čo možno zlepšiť a ďalšie problémy, ktoré sa vyskytli počas týždňa. Prebiehali aj neformálne stretnutia, ktoré sa konali v škole alebo pomocou skype. Ich témou bolo hlavne riešenie menších problémov, ktoré vznikli mimo stretnutí a bolo ich potrebné vyriešiť čo najskôr.

3.1.1. Komunikačné nástroje

Pri výbere nástrojov sme vychádzali zo skúseností a preferencií jednotlivých členov tímu. Analyzovali sme najznámejšie nástroje a dohodli sme sa na nasledujúcich nástrojoch:

Slack

Je to aplikácia, ktorá slúži na komunikáciu a veľa používateľov si ju pochvaľuje aj vďaka zvýšenej produktivite. Dá sa používať aj ako samostatná aplikácia alebo vo webovom prehliadači V aplikácii sme si vytvorili komunikačné kanály pre potrebné témy. Ako príklad jeden kanál je pre GUI ďalší pre datasety a podobne. Umožňuje posielanie súborov, zdieľanie kódov ale aj notifikácie, čo je užitočné hlavne pri naliehavých úlohách. V prvých fázach sa využíval Facebook, nakoľko tam bol registrovaný každý člen tímu.

Nástroj pre zdieľanie obsahu Google Drive

Google Drive používame pre zdieľanie a zálohovanie dokumentov, nie len k dokumentácií ale aj k implementácii. Nachádzajú sa tu aj datasety, logá k našej webovej stránke a pod. Prístup k tomuto priečinku má každý aj vedúci tímu.

3.2. Manažment rozvrhu a rozsahu projektu

Pre správne rozvrhnutie funkčných úloh pre členov tímu je dôkladná príprava a podrobná analýza problému, ktorým sa náš tím zaoberá. Bližšie informácie k tomuto problému poskytla diskusia so zadávateľom, ktorého zastupoval vedúci nášho tímu. Po tomto stretnutí a analýze požiadaviek sa vytvoril zoznam cieľov, ktoré si náš tím rozdelil medzi prioritné a voliteľné, podľa času kedy je ich žiadúce dosiahnuť. Následne bolo potrebné vytvoriť metodiky pre kooperatívne riešenie požiadaviek zadávateľa. Výsledkom tejto analýzy sa stali postupy práce a vývoja. Na základe dodržiavania metodík a postupovaní podľa nami zvolených princípov sme vytvorili sadu štruktúrou elementárnych úloh, na ktorých pracovali vybraní členovia tímu počas dvojtýždňových šprintov. Pre transparentnosť práce na projekte a vnútornú organizáciu jednotlivých úloh náš tím používal TFS – systém na správu projektu.

Práca a rozdelenie úloh medzi členov tímu prebiehalo počas scrum stretnutí na začiatku každého šprintu. Na začiatku sa skontrolovali úlohy v backlogu, podľa progresu na poslednom šprinte a ďalších požiadaviek od zadávateľa sa zhodnotil význam a potreba niektorých úloh. Následne sa podľa analýzy zo stretnutia vytvorili ďalšie „používateľské

príbehy“ a priradili sa k nim implementačné úlohy. Tieto príbehy boli následne ohodnocované všetkými členmi tímu, čím sa určila ich odhadovaná obťažnosť. Podľa tejto hodnoty bola jednotlivým úlohám priradená časová rezerva na ich riešenie. Vzhľadom na štruktúru problému a zameranie členov, každý člen dostal priradenú úlohu podľa oblasti, ktorú si vybral.

Po stretnutí sú nové a priradené úlohy a príbehy evidované v nástroji TFS podľa vopred spísanej metodiky vytvárania „taskov“ v TFS. Evidovanie progresu na jednotlivých úlohách prebiehalo tiež prostredníctvom TFS.

3.3. Manažment plánovania projektu

O manažment plánovania projektu sa v našom tíme staral team líder, manažér dokumentácie pričom ich hlavou úlohou v tejto oblasti bolo udržiavanie konvencií a dodržiavanie metodík tvorby úloh.

Keďže pre implementáciu riešenia bolo zvolené vývojové prostredie Visual Studio 2013 (pôvodne 2015, z dôvodu problémom s knižnicami sme prešli na stabilnejšiu verziu) s potrebnými nástrojmi a knižnicami, bolo pochopiteľné že pre správu projektu a jeho verzií, bol zvolený systém TFS. TFS poskytuje pripravenú štruktúru pre tvorbu úloh z hľadiska agilného vývoja, a tak sa po viacerých diskusiách vytvorili konvencie pre tvorbu a udržiavanie úloh v TFS. Vzhľadom na charakter projektu a výskumného zamerania zadávateľa úlohy bolo potrebné pristúpiť k určitým úpravám z hľadiska názvoslovia „Používateľských príbehov“. Každej úlohe na ktorej sa začalo robiť bolo možné prideliť 3 stavy – „New“ nová, „Active“ aktívna, „Closed“ ukončená. Pre zistené chyby je to o stav „Resolved“ preriešený viac.

Konvencia tvorby, udržiavania a kategorizácie úloh bola dodržiavaná podľa metodiky vytvorenej pre tento účel. Takto vznikli vyššie štruktúry Epics a Features, ktoré predstavovali najväčšie funkčné prvky, ktoré požadoval zadávateľ vo výslednom produkte. Na týchto prvkoch sa postupne pracuje a vytvárajú sa nové podúlohy – „Používateľské príbehy“ (User Stories - UT), ktoré bližšie špecifikujú vlastnosti požadovaných Features. Do každého šprintu sa vybral set UT podľa počtu „story pointov“ a kapacity tímu. Rozdelenie bolo takmer vždy korektné až na ojedinelé prípady keď sa stalo, že sa museli preniesť nedokončené User stories do ďalšieho šprintu, prípadne že člen tímu skončil svoju prácu skôr a do konca šprintu nemal pridelený žiadnu ďalšiu úlohu.

Vedenie agile zasadnutí: Na začiatku každého stand-upu každý člen tímu prezentoval svoj progres za posledný týždeň a predpokladané smerovanie do ďalšieho týždňa. Táto časť bola vždy porovnávaná s agile boardom na TFS. Následne sa v otvorenej diskusií prebrali problémy s jednotlivými úlohami a prichádzalo sa ku konštruktívnym riešeniam. Podľa zistení a nových poznatkov sa podľa potreby vytvorili nové úlohy. Po konzultácií s vedúcim tímu sa navrhli nové „user stories“. Tým sa po ukončení analýzy priradila zložitosť pomocou agile pokrových kariet. Na koniec sa vybrali UT pre jednotlivých členov tímu podľa ich kapacity. Podľa určenej zložitosti sa UT rozdelili na ďalšie implementačné úlohy pre priradeného člena. Bližšia špecifikácia úloh bola vždy obsiahnutá v opise úlohy, takže sa o jej postate dozvedel každý člen tímu aj keď na nej momentálne nepracoval.

Súvisiaca metodika: Metodika vytvárania úloh

3.4. Manažment kvality

Pod manažmentom kvality v našom tíme rozumieme všetky činnosti ktoré vedú k optimalizácii pracovných činností a tým ku zvýšeniu kvality výsledného produktu. Patrí sem: uplatňovanie dohodnutých pracovných a komunikačných metodík, rozdelenie kompetencií jednotlivým členom tímu ako aj používanie kvalitných vývojových nástrojov a knižníc. Súčasťou manažmentu kvality je aj pravidelné neformálne vyhodnocovanie plnenia stanovených úloh a výkonnosti jednotlivých členov tímu, na základe ktorého prispôsobujeme úlohy členov za cieľom zefektívnenia práce. Asi najdôležitejšou činnosťou je však získavanie spätnej väzby od zadávateľa produktu a vyhodnocovanie jeho spokojnosti. Vzhľadom na špecifickú povahu nášho projektu kedy zadávateľom sú výskumníci z ústavu Joanneum Research v Rakúsku, pravidelné a priame stretnutia so zákazníkom nie sú možné. Napriek tomu pre lepšie vyhodnotenie priebežnej spokojnosti a získanie ďalších požiadaviek sme sa dohodli na stretnutí so zástupcom z Joanneum Research na január roku 2016. Do tej doby sa musíme v ohľade požiadaviek vo veľkej miere spoliehať na vlastnú intuíciu a formálneho vlastníka produktu v osobe nášho pedagogického vedúceho p. Dr. Benešovej.

3.5. Manažment rizík

Jednou z hlavných činností ktoré sme vykonávali v počiatkoch projektu bola identifikácia rizík ktoré môžu ohroziť projekt a hľadanie vhodných protiopatrení. Každý z členov sa snažil identifikovať najmä riziká týkajúce sa jeho pridelených úloh a následne sme ich spoločne vyhodnotili a navrhli riešenia. Vybrané identifikované riziká:

Riziko: odchod člena tímu
Pravdepodobnosť: stredná
Preventívne opatrenie: rozdelenie kompetencií v tíme tak, aby boli do riešenia každého problému zapojení aspoň dvaja členovia tímu
Akcia na zmiernenie dopadu: pridelenie kompetencií odišlého člena jeho zástupcovi

Riziko: zadávateľ projektu nedodá testovacie dáta
Pravdepodobnosť: vysoká
Preventívne opatrenie: nájdenie dát na webe, vytvorenie syntetických dát
Akcia na zmiernenie dopadu: použitie záložných dát vytvorených v rámci prevencie

Riziko: nezvládneme dokončiť niektorý z dôležitých medzičlánkov, čo znemožní ďalšiu prácu na projekte
Pravdepodobnosť: nízka až stredná
Preventívne opatrenie: plánovanie úloh a dôkladná analýza možností uplatnenia existujúcich komponentov od tretích strán
Akcia na zmiernenie dopadu: použitie existujúcich komponentov nájdených v rámci prevencie

Riziko: nedostupnosť niektorých kľúčových funkcií pre v knižnici PCL

Pravdepodobnosť: nízka
Preventívne opatrenie: prieskum možností iných knižníc pre spracovanie oblakov bodov, napríklad CGAL, MeshLab

Riziko: nedostatočná kvalita implementácie metód kvôli ich veľkému počtu
Pravdepodobnosť: stredná až vysoká
Preventívne opatrenie: sústredenie sa na 2-3 metódy, pričom na každej pracujú aspoň dvaja členovia tímu

3.6. Manažment integrácie a podpory vývoja

Pre zabezpečenie konzistentnosti počas riadenia tímového projektu s pohľadu jednotlivých častí a zaistenie požadovanej celistvosti modulov systému máme stanovený postup integrácie a podpory vývoja nasledovne:

Jednotlivý členovia tímu sú zodpovedný za ucelené časti tímového projektu ktoré spracúvajú. Majú na starosti ich kvalitu vyhotovenia a tým pádom aj výstup daného modulu a celkové spracovanie. Zodpovedný členovia komunikujú spoločne a aj s integrátorom projektu a podpory vývoja. Tým je dosiahnutá potrebná transparentnosť a lepšia spolupráca. Ďalej integrátor zozbiera od jednotlivých členov ich vytvorené moduly a vytvorí konečnú podobu procesu.

Pre prehľadnú spoluprácu tvorby softvéru, manažovanie práce, úloh a zdieľanie zdrojového kódu sme sa rozhodli použiť TFS - Visual Studio Team Foundation Server 2015, ktorý je jednoducho prepojiteľný a synchronizovateľný s Microsoft Visual Studio. Vďaka tomu dokážeme dodržiavať dohodnuté postupy a konvencie, udržiavať kód prehľadný a zároveň všetkým dostupný. Poskytuje jednoduché aktualizovanie zmien v zdrojovom kóde od všetkých členov tímu.

Kvôli potrebe vytvoriť prehľadnú komunikáciu medzi všetkými členmi tímu sme si zvolili aplikáciu Slack, ktorá Umožňuje triediť si jednotlivé problémy do oddelených konverzácii aby sa medzi sebou nemiešali a taktiež ponúka rôzne možnosti citovania, vytvárania útržkov zdrojového kódu či možnosť nainštalovania na smart zariadenia.

4. Používané metodiky

názov	opis	autor
Metodika komunikácie	Pravidlá pre používanie aplikácie Slack na komunikáciu v tímoch. Vytváranie kanálov, odpovedanie na správy a pod.	Róber Karásek
Metodika evidencie úloh v TFS	Pravidlá pre vytváranie nových úloh v nástroji TFS. Ktoré polia treba vyplniť a ako, ako správne nastaviť všetky parametre.	Lukáš Hudec
Metodika písania zdrojového kódu	Konvencie pre písanie zdrojového kódu v C++ v rámci projektu.	Róbert Karásek
Metodika pre technickú dokumentáciu	Pravidlá písania technickej doku mentácie pomocou Doxygen.	Róbert Karásek
Metodika pre commiting a branching	Ako správne odosielať (commitovať) zdrojový kód a organizovať ho do vetiev (branching)	Lukáš Hudec
Metodika testovania	Návod na vykonávanie testov	Róbert Karásek

Tabuľka 2 Zoznam používaných metodík s ich opisom

5. Globálna retrospektíva

5.1. Zimný semester

Ako tím sme sa posunuli počas riešenia projektu v zimnom semestri dopredu vo viacerých oblastiach. Ako prvý problém sme riešili komunikáciu v tíme. Na komunikovanie v rámci tímu sme prestali používať komunikačný nástroj Facebook a začali sme komunikovať prostredníctvom nástroja Slack, čím sme oddelili komunikačný pracovný kanál od iných komunikačných kanálov (zábava, priatelia, ...), ktoré výrazne ovplyvňovali úspešnosť dorozumievania v tíme pri riešení pracovných záležitostí.

Ďalším výrazným krokom vpred bola zmena manažmentu evidencie úloh v TFS (Team Foundation Server). Zmena nastala po 2. šprinte, dovtedy každý člen evidoval úlohy v rámci svojej user-story, ktorú si dopredu na stretnutí vytvoril. Časová náročnosť úloh bola rozložená na približne 16 hodín. Avšak po odpracovaní svojich úloh člen môže robiť na ďalších úlohách, v prípade, že má definované user-story a malé úlohy. Týmto spôsobom sa to nedalo. Aj ako začiatočníci sme nevedeli dobre ohodnocovať úlohy (ako začiatočníci všeobecne. v oblasti rekonštrukcie objektov) a mohlo sa stať, že sme ohodnotili úlohu vysokým číslom a úloha bola ľahko riešiteľná (nikto neoponoval). Nový manažment predstavoval definovanie množiny veľkých úloh v podobe “epics”, “features”, ktoré budú otvorené počas celej fázy riešenia projektu a nebudú sa meniť. Tieto ďalej členíme do “user- stories”, čo predstavujú väčšie úlohy riešené v rámci šprintu. Snažili sme sa zvoliť čo najväčšiu množinu týchto úloh, aby každý člen si potom mohol v šprintoch vyberať riešiť danú user-story a definovať k nej malé úlohy. Rovnako sme zaviedli vytváranie úloh v rámci každej user-story, ktoré budeme vedieť presne ohodnotiť, či sú vykonané alebo nie, teda tzv. „definition of done“ úlohy.

V ďalšej fáze projektu sme narazili na problém pri kompilovaní knižníc OpenCV verzie 3.0.0 vo vývojovom prostredí VS2015 a navyše niektoré z knižníc OpenCV verzie 3.0.0 si vyžadovali podporu platformy CUDA a zároveň táto ešte nepodporuje VS2015, preto sme sa rozhodli jednohlasne pre migráciu z VS2015 do VS2013, kde knižnica OpenCV verzie 3.0.0 má plnú podporu.

Pre potrebu ľahkého dokumentovania kódu projektu sme sa rozhodli zaviesť používanie nástroja Doxygen. Dokumentácia sa generuje na základe komentárov pri jednotlivých úsekoch kódu. Doxygen prepája časti kódu a referencie na dokumentáciu k týmto častiam kódu, čím sa stáva projekt prehľadnejší z dlhodobého hľadiska pre všetkých záujemcov o prácu s projektom.

Pre správnosť kódu v poslednom období plánujeme zaviesť testovanie. Testovanie kódu by sa malo uskutočňovať prostredníctvom Google testov. Testovanie pravdepodobne vykonáme až budúci semester nakoľko doteraz neboli naimplementované žiadne väčšie funkčné celky.

5.2. Letný semester

Zaužívaný spôsob zadávania a evidencie úloh sme použili aj v letnom semestri.

Úlohy v tíme zostali rozdelené do týchto troch oblastí: práca na rekonštrukčných metódach, adaptácia grafického rozhrania podľa pribúdajúcej funkcionality, export rekonštruovaných dát, propagácia projektu a dokumentácia.

Po tom ako sme zovšeobecnili prvý funkčný prototyp rekonštrukcie nastala fáza intenzívnejšieho testovania a ladenia parametrov rekonštrukčných metód. Vtedy sme identifikovali potrebu konfiguračného mechanizmu. /*Jeho implementáciou a údržbou bol poverený Martin Jurík. */

Keďže sme narazili sme na problém, kedy rekonštrukcia vyžadovala veľké množstvo výpočtových prostriedkov a jej vykonanie vyžadovalo dlhší čas, sprevádzkovali sme náš projekt na výkonnom školskom počítači. Tento sa nachádza na fakulte v laboratóriu Siemens Healthcare, do ktorého mali prístup všetci členovia tímu a v ktorom sme sa od letného semestra stretávali.

K tomuto počítaču sme pre pohodlnejšiu prácu zriadili vzdialený prístup pomocou nástroja Teamviewer. Keďže na počítači mohol súčasne pracovať vždy len jeden človek, vytvorili sme si v komunikačnom nástroji Slack kanál prostredníctvom ktorého sme si prístup k tomuto stroju plánovali a rezervovali.

Počas semestra sme upustili od pôvodného plánu implementovať do nášho nástroja interakciu pomocou 3D myši z dôvodu, že sme objavili možnosť jej využitia v existujúcom modelovacom nástroji AutoCAD.

Počas semestra vyplynuli nové úlohy spojené s propagáciou nášho produktu

- prezentácia produktu na študentskej vedeckej konferencii IIT.SRC
- príprava článku pre portál robime.it
- prezentácia produktu na TP-cupe
- konzultácia s výskumným pracovníkom z ústavu Janneum Research

Na základe komentárov zdrojových kódov sme pomocou nástroja Doxygen vygenerovali technickú dokumentáciu.

Zoznam kompetencií tímu

Každý v tíme zohráva dôležitú rolu, hlavné oblasti zodpovednosti sme si rozdelili nasledovne:

- manažér architektúry - Lukáš Hudec
- manažér vývoja - Róbert Birkus
- projektový integrátor - Martin Jurík
- biznis manažér - Michal Löffler
- test manažér - Róbert Karásek
- manažér propagácie - Michal Korbeľ
- manažér dokumentácie - Katarína Janečková

Metodiky

1. Metodika písania zdrojového kódu

Header Files

- každý .cpp súbor musí mať "pridružený" .h súbor, okrem unit testov a malých .cc, ktoré obsahujú iba main()
- prípona .h
- súbory, ktoré začleňujú text, ale nie sú hlavičkovými majú príponu .inc (napr. ak sa používajú na viacerých miestach v kóde, alebo sú určené pre špecifickú platformu)
- každý by mal obsahovať „header guards"(nižšie), a mal by byť prepojený s ďalšími headermi
- ak je v headeri deklarovaná template alebo inline funkcia, tak definície týchto konštruktorov musia byť v každom .cpp súbore, ktorý ich používa
- (As an exception, a function template that is explicitly instantiated for all relevant sets of template arguments, or that is a private member of a class, may be defined in the only .cpp file that instantiates the template.)

The #define Guard

- každý header musí obsahovať #define guard
- formát <*PROJECT*>_<*PATH*>_<*FILE*>_H_
    - napr. súbor v foo/src/bar/baz.h by mal vyzerať nasledovne

```
#ifndef FOO_BAR_BAZ_H_

#define FOO_BAR_BAZ_H_

#endif  // FOO_BAR_BAZ_H_
```

Forward Declarations

- ako predísť zbytočným #includes
- ak používam funkciu zadeklarovanú v headeri, vždy #include konkrétny header
- pri použití class template, pre istotu #include jeho header
- ak používam bežnú triedu, spoliehať sa na poprednú dekleráciu je OK, ale treba vždy skontrolovať
- nenahradzovať datové členy s ukazovateľmi len preto aby sa zabránilo #include

Inline Functions

- iba funkcie, ktoré majú malý počet riadkov (napr. 10 a menej)
- kompilátor zavolá funkcie inline, teda neprejdu zvyčajným volacím mechanizmom
- neodporúča sa aby inline funkcia obsahovala cykly alebo switch (može ak sa nikdy nebude vykonávať), nemali by byť ani rekurzívne

Function Parameter Ordering

- najprv vstupy, potom výstupy (aj keď pridáme nový vstup, tak na začiatok s ním)

Names and Order of Includes

- všetko od source zložky teda
    - google-awesome-project/src/base/logging.h → #include "base/logging.h"
- ak *dir/foo.cc* alebo *dir/foo_test.cc* ma za úlohu implementovať alebo testovať *dir2/foo2.h,* poradie includov bude nasledovné
    - dir2/foo2.h.
    - C system files.
    - C++ system files.
    - Other libraries' .h files.
    - Your project's .h files.
        - každá sekcia zoradená podľa abecedy
- **Príklad**
    - google-awesome-project/src/foo/internal/fooserver.cpp
    - *#include "foo/server/fooserver.h"*

      #include <sys/types.h>
      #include <unistd.h> #include <hash_map>
      #include <vector>

      #include "base/basictypes.h"
      #include "base/commandlineflags.h"
      #include "foo/server/bar.h"
- ak obsahuje podmienky, tak až nakoniec
  #include "foo/public/fooserver.h"

  #include "base/port.h" // For LANG_CXX11.
  #ifdef LANG_CXX11
  #include <initializer_list>
  #endif // LANG_CXX11

Scoping
Namespaces

- nepomenované namespace je v .cpp súboroch podporované
- ak chcem pomenovať tak v kombinácii s menom projektu a jeho path
- nepoužívať inline namespace
- (Namespaces subdivide the global scope into distinct, named scopes, and so are useful for preventing name collisions in the global scope.)
- *namespace { // This is in a .cc file.*
  // The content of a namespace is not indented.
  //
  // This function is guaranteed not to generate a colliding symbol
  // with other symbols at link time, and is only visible to
  // callers in this .cc file
  bool UpdateInternals(Frobber f, int newval) {*
  ...
  }
  } // namespace
- ***// In the .h file***
  namespace mynamespace {

  // All declarations are within the namespace scope
  // Notice the lack of indentation
  class MyClass {
  public:
  ...
  void Foo();
  };

  } // namespace mynamespace
- ***// In the .cpp file***
  namespace mynamespace {
  // Definition of functions is within scope of the namespace.
  void MyClass::Foo() {
  ...
  }
  } // namespace mynamespace
- *#include "a.h"*
  DEFINE_bool(someflag, false, "dummy flag");

```
class C;  // Forward declaration of class C in the global namespace.
namespace a { class A; }  // Forward declaration of a::A.
namespace b {
        .code for b...        // Code goes against the left margin.
}  // namespace b
```

- **nič nedeklarovať v namespace std ani popredne deklarované classy zo standard lib**

Nested Classes

- nerobiť vnorené triedy ak nie su súčasťou interface
- používať nonmember funkcie s namespace alebo static member funkcie namiesto globalnych funkcii

Local Variables

- int i;
  i = f();
- int j = g();

- vector<int> v;
  v.push_back(1);
  v.push_back(2);
- vector<int> v = {1, 2};

Static and Global Variables

- statické alebo globálne premenné typov tried sú zakázané

Classes

- nikdy nevolať konštruktorom virtuálne funkcie
- C++ keyword explicit for constructors callable with one argument.
- používať delegovanie a dedičnosť konštruktorov, keď znižujú duplicitu kódu
- poradie public: pred protected: pred private: , metódy pred dátovými členmi (premenné)
- Typedefs and Enums
- Constants (static const data members)
- Constructors
- Destructor
- Methods, including static methods
- Data Members (except static const data members)

Ak ma funkcia viac ako 40 riadkov = rozmýšlať či sa nedá rozdeliť.

General Naming Rules

- Premenné pomenovávať zrozumiteľne, slová oddeľovať podčiarovníkom
  int price_count_reader; // No abbreviation.
  int num_errors; // "num" is a widespread convention.
  int num_dns_connections; // Most people know what "DNS"
- **mená súborov taktiež s _ alebo -**
  my_useful_class.cc
  my-useful-class.cc
  myusefulclass.cc
  myusefulclass_test.cc
- **mená metód, tried** – *CamelCase*
  - globálne s prefixom g_ napríklad
  - *static Pool<UrlTableProperties>* pool;*
- funkcie = CamelCase
- **enum** CamelCase
- **makrá** VELKYM_PISMOM
- **komentáre**
  - /* nad metódami */
  - //v riadku
- max dĺžka riadka 80 písmen
- nepoužívať taby :(iba space a 2x
- PODMIENKY

```
if (condition) {  // no spaces inside parentheses
  ...  // 2 space indent.
} else if (...) {  // The else goes on the same line as the closing brace.
  ...
} else {
  ...
}

if (x == kFoo) return new Foo();
```

  - **jednoriadkový for**

```
for (int i = 0; i < kSomeNumber; ++i)
  printf("I love you\n");
```

alebo

```
for (int i = 0; i < kSomeNumber; ++i) {
  printf("I take it back\n");
}
```

- viac podmienok

```
if (this_one_thing > this_other_thing &&
    a_third_thing == a_fourth_thing &&
    yet_another && last_one) {
  ...
}
```

- POINTRE

```
// These are fine, space preceding.
char *c;
const string &str;

// These are fine, space following.
char* c;        // but remember to do "char* c, *d, *e, ...;"!
const string& str;
```

- RETURN

```
return result;                  // No parentheses in the simple case.
// Parentheses OK to make a complex expression more readable.
return (some_long_condition &&
        another_condition);
```

- MINIMALIZOVAT vertikálny space
- **HLAVNE BYŤ KONZISTENTNÝ**

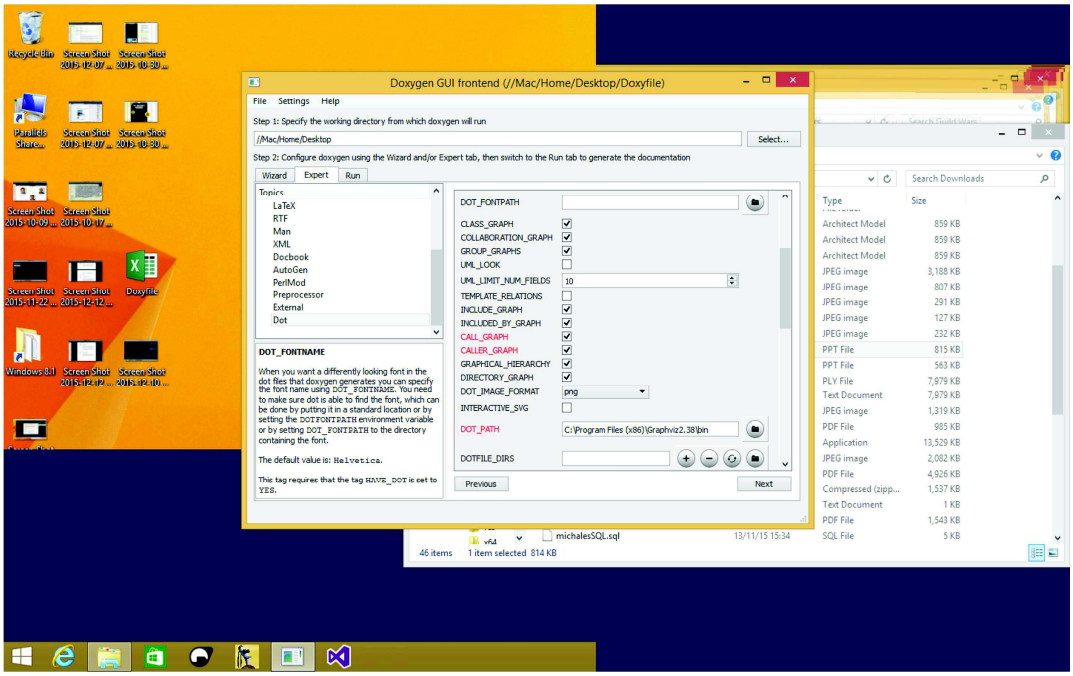
Tipy

- cpplint nájde chyby vo formátovaní a pod.
- nepoužívať exceptrions RTTI
- static_cast<>(). Do not use other cast formats like int y = (int)x; or int y = int(x);
- Use prefix form (++i) of the increment and decrement operators with iterators and other template objects.

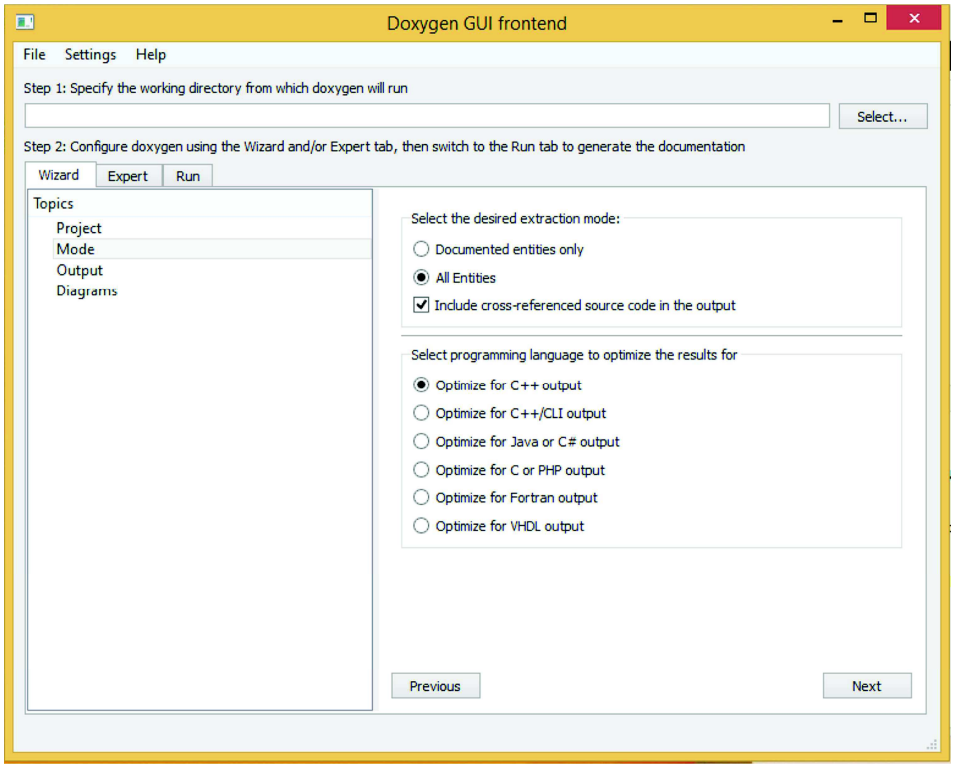
2. Metodika pre technickú dokumentáciu

Na vygenerovanie technickej dokumentácie sa používa Doxygen(verzia 1.8.10).
Inštalácia sa nachádza na http://www.stack.nl/~dimitri/doxygen/download.html.

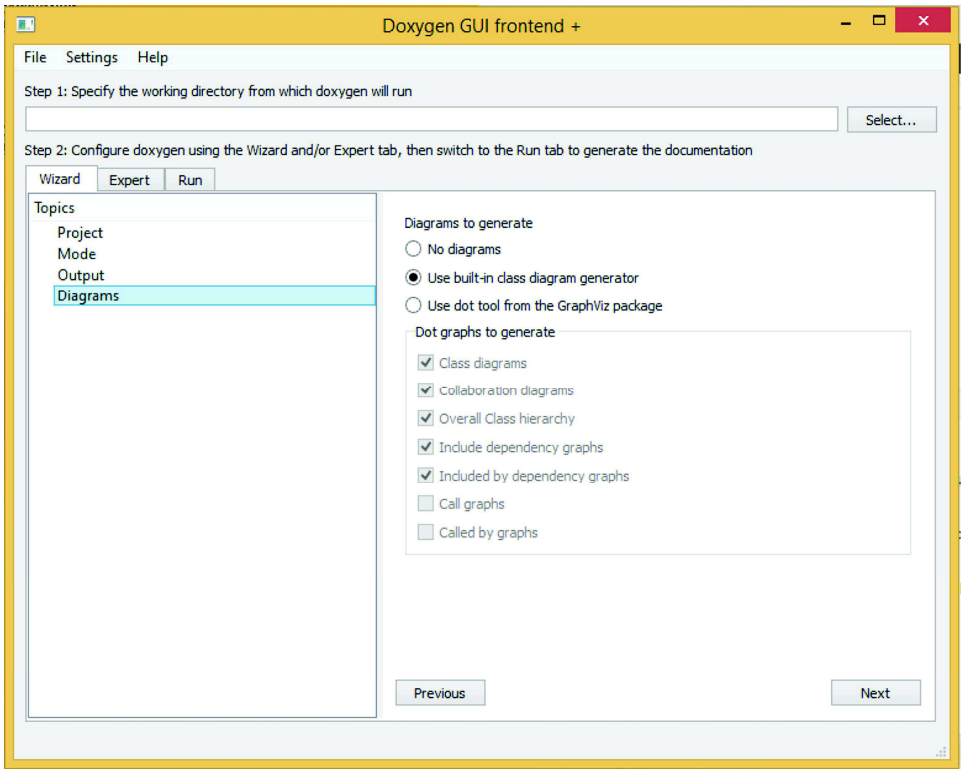
- Každý autor zdrojového kódu je nútený komentovať a napísať hlavičku podľa (http://www.stack.nl/~dimitri/doxygen/manual/docblocks.html)
- pre zistenie špeciálnych príkazov (http://www.stack.nl/~dimitri/doxygen/manual/commands.html)
- Komentáre musia byť stručné a výstižné
- Treba komentovať len potrebné časti kódu
- Netreba spomínať znova zdrojový kód v komentári
- Nezakomentovávať zdrojový kód!!!
- Konfigurácia je vyobrazená na obrázkoch
- Už vygenerovanú dokumentáciu je potrebné ukladať na Drive, rozdelenú do priečinkov podľa formátov
- vždy keď sú otázky, treba použiť kanál doxygen v slacku
- Bolo pridané generovanie diagramov pre ktorý je potrebný program Graphviz s *dot*
- Cestu k programu treba nastaviť v záložke expert viď obr. 1 alebo použiť konfiguračný súbor, ktorý sa nachádza na google drive
- Pri generovaní treba označiť všetky typy diagramov
- Vždy nahrať novú vygenerovanú dokumentáciu aj na google drive



Obrázok 1 Nastavenie dot



Obrázok 2 Nastavenie Doxygen



Obrázok 3 Nastavenie Doxygen

3. Metodika evidencie úloh v nástroji TFS

Prístup k nástroju je možný cez portál tfs.fiit.stuba.sk, alebo s použitím pluginu vo Visual Studio 2015/2013 po presune do tohto vývojového prostredia. Postup vo vypĺňaní jednotlivých polí je rovnaký a výstup musí spĺňať metodikou zavedenú normu. Pre polia, ktoré obsahujú všetky udalosti platia rovnaké pravidlá.

Momentálne existujú 3 najvyššie celky - Epic, Feature a User Story. Epic a Feature sa vytvoria na naplánujú s vlastníkom produktu a predstavujú hlavné celky požiadaviek. Ďalej sa už len pridávajú User Story do Feature podľa zamerania požiadavky a úlohy s ňou spojenej.

Pri vytváraní úlohy sa riadíme najvyšším celkom - User story, jednotlivé úlohy pre členov tímu sú označené ako Task a Bug.

- *User story:*
  - definovaný používateľský príbeh, abstrahovaná menšia časť požiadavky vlastníka produktu - implementačná oblasť problému
  - má svojho zodpovedného vedúceho - ten je pridelený na stretnutí tímu, podľa osobnej žiadosti alebo určený podľa skúseností
  - ak nie je určený, user story vytvorí team leader a úlohu nastaví podľa dohodnutých kritérií, zodpovedný vedúci bude následne určený hlasovaním v tíme
- *Task:*
  - jednotlivé úlohy používateľského príbehu
  - ak sú jasné hneď na stretnutí, vytvorí ich zodpovedný vedúci a pridelení členovia tímu si ich rozdelia podľa vlastného záujmu tak, aby ich stihli za šprint vypracovať
- *Bug:*
  - označenie pre úlohu zaoberajúcu sa chybou v programe alebo dokumentácií
  - jej vytvorenie má na zodpovednosť ten člen, ktorý chybu odhalil, ale riešenie môže po konzultácií so zodpovedným vedúcim user story ponechať na iného člena - zodpovedného za oblasť, v ktorej bola chyba odhalená

Vypĺňanie polí úloh

Celá dokumentácia a správa projektu je písaná výhradne v slovenčine s použitím diakritiky okrem explicitných výnimiek vyznačených v metodike(pokiaľ nie sú zistené žiadne problémy).

- Meno:
  - začiatočné písmeno veľkým, ostatné podľa pravidiel slovenského jazyka
  - stručný a výstižný
- Tagy:
  - využitie všeobecne známych a výstižných skratiek
  - bez diakritiky

    - môže byť nastavených viac, podľa potreby
- Detail, Description:
    - detailnejší popis úlohy, z ktorého bude každému jasné o čo v danej úlohe ide
    - spomenúť možné zmeny v úlohe
    - v prípade zmeny opísať všetky zmeny, ktoré nastali
- Stav:
    - keď je úloha vytvorená, je označená ako "New" (prednastavené, nemožno zmeniť)
- Priradenie (Assigned to):
    - vyplnené podľa typu úlohy a stavu, v ktorom sa nachádza
- Priorita:
    - defaultne nastavená na hodnotu 2
    - upravená podľa dohody na stretnutí a momentálnych požiadaviek
- Iterácia:
    - nastavuje sa podľa toho do ktorého šprintu daná úloha patri - táto hodnota musí byť vyplnená korektne, inak úloha nie je správne zobrazená v tabuľke
- Effort:
    - vypĺňajú sa 3 hodnoty - "Original estimate", "Remaining" a "Completed"
    - Original estimate:
        - odhadovaný čas, ktorý riešenie zaberie
        - nastavená hodnota od ktorej sa počíta burndown chart
    - Remaining
        - original estimate - completed (odpracovaný čas) = zostávajúci čas
        - podľa tejto hodnoty sa počíta pokles burndown chart-u
    - Completed
        - riešiteľ zadáva postupne koľko času na riešení tejto úlohy strávil

Stav úlohy

Stav závisí od toho či je to "User story" alebo "Task".
Hlavné stavy:

- New
    - nastavená pokiaľ je priamo vytvorená
- Active
    - keď riešiteľ začne pracovať na úlohe zmení jej stav na aktívnu
    - "User stories", na ktorých sa pracuje majú tento stav od začiatku riešenia aj keď sa práve nerieši žiadna z ich taskov
- Resolved
    - "User story", ktorej riešenie je hotové a prešlo aj testami
    - (Task Item tento stav nemá)
- Closed
    - ukončená úloha, na ktorej sa predpokladá už viac nebude pracovať

Šprinty

Šprinty sa vytvárajú s dvojtýždňovým trvaním, najčastejšie od agile stretnutia po najbližšie stretnutie. V prípade technických alebo problémov v manažmente je možné tieto termíny posunúť - najviac však o 2 dni. Ak šprint obsahuje úlohy k doplneniu dokumentácie, tieto úlohy sa evidujú v jednej špeciálnej User Story, vzhľadom na to, že dokumentáciu potrebuje aj vlastník produktu, jej štandardné ohodnotenie je 5 SP.

Šprinty vytvára a spravuje biznis manažér, ktorý zastupuje scrum mastera, prípadne tím líder, ktorý má prehľad o dianí v tíme a progrese práce na jednotlivých úlohách.

Vytvorenie úloh

Úlohy sa plánujú na agile zasadnutí tímu, na ktorom sa plánuje nový šprint. Na začiatku každého stretnutia sa vykoná stand-up, na ktorom každý člen tímu opíše svoj progres za posledný týždeň, prípadné problémy. Ak sa vyskytnú problémy, ostatní členovia tímu produktívne prispejú s pomocou k riešeniu daného problému.

V prípade stretnutia s plánovaním šprintu, po stand-up scrum master vedie plánovanie nových úloh na nasledujúci šprint. Ak vznikli nové požiadavky vlastníka produktu, najprv sa určí ich priorita a podľa jej výšky sa vyberajú požiadavky s backlogu v kombinácií s novými požiadavkami. Vybrané požiadavky sa vložia do backlogu šprintu a ostatné sa uložia do backlogu a evidujú sa k Epicom a Featurám podľa tematiky. Ak majú vzniknúť nové tím sa riadi vyššie spomenutým postupom vyplnenia úloh. Vybrané úlohy sa ohodnotia pokrom a prebehne analýza, ktorá má pomôcť zodpovednému riešiteľovi vytvoriť postup práce na úlohe - taktiež evidovať svoju prácu v TFS ako Task Items. Bližšie informácie k správe úloh podľa agilných metód sú v dokumente “Manažment plánovania projektu”.

Pokiaľ sú tieto úlohy/Task Items jasné počas stretnutia - zadáva ich do TFS team leader, zodpovední členovia tímu len vyplnia Description. V opačnom prípade si ich vytvárajú členovia postupne ako podľa vlastnej analýzy zistia, aké úlohy ich práca bude obnášať.

Zápisnice

Písanie zápisov zo stretnutia sa eviduje rovnako ako ostatné úlohy a ich zaradenie spadá do každého šprintu avšak bez priradenia k User Story. Zápisy majú svoju formu a predpokladaný obsah, čo je určené šablónou, ktorá sa nachádza v priečinku na Google Drive tímu.

- Názov úlohy je zadaný v tvare:
    - “Zápisnica č.X (‘RRRRMMDD’)” (napr. Zápisnica č.5 (20151001))
- Description
    - obsahuje text zápisnice (stačí plain text)
- Attachements
    - formátovaná pdf verzia zápisnice

4. Metodika pre komunikáciu

Komunikácia

Nami zvolený systém pre komunikáciu medzi členmi tímu je Slack.

- Možnosť vytvoriť kanál ma každý člen tímu
- Do komunikačného kanála treba písať vždy k danej téme, ak neviem kde patrí konkrétna otázka, treba sa opýtať v general chate alebo konkrétnej osoby, na ktorú je smerovaná.
- používať slack anotácie (napr. vloženie kódu do `code`)
- Slack kontrolovať denne

Zdieľané dokumenty

Nami zvolený systém pre zdieľanie dokumentov je Google Drive.

- Štruktúru treba zachovávať ako je, prípadne požiadať o zmenu vedúceho tímu.
- Je dovolené meniť dokumenty ak sa nájdu v nich nezrovnalosti alebo presunúť súbor do správneho priečinka
- Nahrávať súbory vždy načas

5. Metodika pre odosielanie (committing) a používanie vetiev (branching)

Všeobecné:

Používa sa GIT integrovaný vo Visual Studiu.

- Správa commitu obsahuje
    - iniciály autora alebo spoluautorov v tvare [MP,MP,MP] (Meno Priezvisko)
    - stručný opis zmien a doplnkov
    - poznámka ak je časť kódu neoverená/netestovaná
- Povolenie k odoslaniu kódu závisí od vetvy, do ktorej je odosielaný
- Implicitne sa odosiela LEN skompilovateľná verzia (aby sa zbytočne neobmedzovali ostatní členovia tímu)
- Ďalšie pravidlá závisia od vetvy, do ktorej chce vývojár odoslať kód

Vetvenie:

Pre správu kódu sú založené 3 vetvy. *"Master, Development, Debugging"*. Každá má osobitné pravidlá používania.

Všeobecné informácie k vetvám:

- medzi vetvami je možné prepínať
    - len ak sú odoslané všetky zmeny
    - ak zmeny nie sú odoslané, ale obe vetvy majú rovnakú verziu (rovnaký commit)
- medzi vetvami nie je možné prepínať
    - medzi rozdielnymi verziami vetiev, keď nie sú odoslané zmeny

Kedy a akú vetvu použiť:

"Master":

O túto vetvu sa stará len poverený integrátor, ktorý má na starosti spojenie nového otestovaného a skontrolovaného kódu do hlavnej vetvy, z ktorej sa vytváraná aplikácia nasadzuje.

Do vetvy sa môže odoslať kód, len pokiaľ došlo k chybe v spájaní vetiev, alebo pri oprave ľudskej chyby po odosielaní kódu a jeho nedostatočnej kontroly.

"Development":

Podľa názvu je jasné, že je to hlavná vetva pre udržiavanie a verziovanie kódu.

Do tejto vetvy sa môže odosielať len funkčný kód, ktorý nevracia výnimky a nepadá počas spoločnej línie vykonania pre všetky metódy. Čo znamená, že pre ďalších, ktorí stiahnu odoslanú verziu budú môcť pracovať a testovať kód bez problémov a obmedzení.

Čo sa odosiela do tejto vetvy musí dodržiavať konvencie.

Kto odošle niečo do developmentu a je si istý, že tá verzia má byť presunutá do Mastra, odosielaný kód musí obsahovať dokumentáciu a mať priložené Unit Testy. Vývojár je ďalej povinný oznámiť to človeku zodpovednému za kontrolu a revíziu kódu, ktorý je povinný tento

kód skontrolovať a pokiaľ je v poriadku dať vedieť integrátorovi o možnom presune verzie do Mastra. Pokiaľ ale odhalí chyby v kóde, odosielateľ kódu je povinný opraviť chyby, ktorých sa dopustil a odoslať/appendnúť zmeny do vetvy.

"Debugging":

Táto vetva môže obsahovať aj chybný, nekompilovateľný kód - táto vetva je vytvorená pre spoločné riešenie problémov vznikajúcich pri implementácií.

1. Pravidlo: odosiela sa v zásade fungujúci kód
2. Pravidlo: pokiaľ hľadáme pomoc u kolegov, je potrebné oznámiť (určeným spôsobom), že posledný commit je neskompilovateľný čo teda pre tých, ktorí na ňom nemajú v pláne pracovať znamená, že ho nebudú sťahovať
    a. oznámenie o nefunkčnosti - oznámiť v správe commitu
    b. oznámenie o chybe - vytvoriť Bug v TFS a opísať chybu, priložiť prípadné screeny, stack trace a pod.
    c. žiadosť o pomoc - oznámiť prostredníctvom Slack-u
3. Pravidlo: keď sa problém vyrieši, riešiteľ je do vetvy povinný odoslať fungujúcu verziu

Poznámka:

Ak sú nezrovnalosti ohľadne commitov, treba kontaktovať integrátora alebo autora metodiky.

6. Metodika testovania

Prvá fáza

Každý, kto vytvára samostatnú metódu je zodpovedný za prvé testovanie a za integrovanie do systému je zodpovedný integrátor. Jeho úlohou je aj otestovanie, či metóda je funkčná v systéme a negatívne neovplyvňuje iné.

Druhá fáza

Pri druhej fáze sa už úlohy menia a testovanie ostáva na testeroch. Hlavným bodom testera je otestovanie novej implementácie. Funkčnosť samostatnej novej implementácie a vplyv na ostatné moduly.

Výsledok testovania je správa po testovaní a vyhodnotenie miery úspešnosti. Najprv ide o vyhodnotenie unit testov (viď nižšie) a neskôr akceptačné testovanie. Pri akceptačnom testovaní je potreba vývojára, ktorý danú implementáciu vykonal a projektového manažéra. Ich hlavnou úlohou je na základe požiadaviek overiť, či sú naozaj splnené požiadavky, vykonáva sa to čo má a dostávame správne výsledky. Po ich schválení môžeme testovanie a aj implementáciu považovať za úspešnú. Tieto výsledky a výsledky z unit testov sa zaznamenajú do google drive.

Unit testy

Pomocou nich sa zisťuje správna funkčnosť modulu v systéme aj po novej implementácii alebo po zmene. V prípade, že niektorý z unit testov dá nesprávne výsledky, je potrebné vrátiť zmenu do pôvodnej a odhaliť chybu prípadne doimplementovať.

Pre unit testy sa používa framework *Google test.* Vstupu a výstup je možné vidieť na ukážkovom prípade nižšie. Tento framework obsahuje širokú škálu funkcií, ktoré treba postupom projektu využiť, taktiež sledovať pokrytie kódu testami.

```
#include "gtest/gtest.h"

TEST(SquareRootTest, PositiveNos) {
    EXPECT_EQ (18.0, square-root (324.0));
    EXPECT_EQ (25.4, square-root (645.16));
    EXPECT_EQ (50.3321, square-root (2533.310224));
}

TEST (SquareRootTest, ZeroAndNegativeNos) {
    ASSERT_EQ (0.0, square-root (0.0));
    ASSERT_EQ (-1, square-root (-22.0));
}

int main(int argc, char **argv) {
  ::testing::InitGoogleTest(&argc. argv);
  return RUN_ALL_TESTS();
}
```

Obr.1 Pokrytie kódu testami

```
Running main() from user_main.cpp
[==========] Running 2 tests from 1 test case.
[----------] Global test environment set-up.
[----------] 2 tests from SquareRootTest
[ RUN      ] SquareRootTest.PositiveNos
..\user_sqrt.cpp(6862): error: Value of: sqrt (2533.310224)
  Actual: 50.332
Expected: 50.3321
[  FAILED  ] SquareRootTest.PositiveNos (9 ms)
[ RUN      ] SquareRootTest.ZeroAndNegativeNos
[       OK ] SquareRootTest.ZeroAndNegativeNos (0 ms)
[----------] 2 tests from SquareRootTest (0 ms total)

[----------] Global test environment tear-down
[==========] 2 tests from 1 test case ran. (10 ms total)
[  PASSED  ] 1 test.
[  FAILED  ] 1 test, listed below:
[  FAILED  ] SquareRootTest.PositiveNos

 1 FAILED TEST
```

Obr.2 Výstup testov z Obr.1

Exporty z TFS

1. šprint

ID	Title	Assigned To	Iteration Path	Description	Tags
2622	Analýza metódy horizontálnych rezov	Bc. Katarina Janeckova	3DRecon\Sprint 1	Analyzovať: - spôsob určenia a extrakcie rezu - projekcia bodov s určitou normálou na z-os	analyza
2698	Analýza metódy mean shift	Bc. Katarina Janeckova	3DRecon\Sprint 1	Analyzovať metódu mean shift, spôsob jej využitia a implementáciu v knižniciach.	analyza
2682	Načítanie súboru formátu PLY	Bc. Lukas Hudec	3DRecon\Sprint 1	PointCloud data je možné ukladať v rôznych typoch súborov a od toho sa odvíja aj formát akým sú uložené. Jedným možným spôsobom je načítavať dáta z formátu PLY (polygón)	
2683	Načítane súboru formátu OBJ	Bc. Lukas Hudec	3DRecon\Sprint 1	Ďalším možným formátom pre uloženie dát je formát Obj (wawefront) knižnica pcl obsahuje funkcionalitu pre načítanie aj tohto formátu.	
2603	Analýza dát a konfiguračných súborov	Bc. Lukas Hudec	3DRecon\Sprint 1	Analýza a získanie vedomostí o možnostiach implementácie a spracovania dát z viacerých snímok s určením pozície kamery v konfiguračnom súbore alebo hlavičke jednotlivých načítavaných súborov.	analyza
2604	Implementácia načítania súborov	Bc. Lukas Hudec	3DRecon\Sprint 1	Implementácia načítavacieho rozhrania s možnosťou načítať "multiple files" a automatické vyhľadanie konfiguračného súboru.	dev
2605	Implementácia kombinácie snímok scény	Bc. Lukas Hudec	3DRecon\Sprint 1	Kombinácia snímok do jednej karteziánskej sústavy podľa určenia pozície kamery z konfiguračného súboru, alebo metadát - hlavičky - súborov obsahujúce jednotlivé snímky	dev

2606	Testovanie a doimplementácia pre rôzne vstupy	Bc. Lukas Hudec	3DRecon\Sprint 1	Ako bolo spomenuté existujú rôzne typy dát a rôzne vstupy - všetky majú ale spoločnú podstatu - pozícia kamery a súradnice snímky - je potrebné preto docieliť aby sa problém s načítavaním týchto dát zovšeobecnil a výstup zjednotil.	dev; test
2608	Zápisnica č. 3 (20151008)	Bc. Martin Jurik	3DRecon\Sprint 1		doku
2567	Nacitanie dat formatu PCD	Bc. Martin Jurik	3DRecon\Sprint 1	Vyhladanie informacii a nacitanie dat formatu PCD, pokus o jednoduche zobrazenie a nastavenie zobrazenych bodov do kamery	dev
2578	Vyhladanie informacii o formate LAS	Bc. Martin Jurik	3DRecon\Sprint 1	Vyhladanie informacii o formate LAS, analyza struktury ulozenych dat, moznosti jeho nacitania a konverzie do pouzitelneho formatu	dev
2579	Nacitanie dat vo formate LAS	Bc. Martin Jurik	3DRecon\Sprint 1	Nacitanie point could-ov vo formate LAS a testovanie	dev
2651	štúdium PCL funkcií vhodných pre GR, výpočet normál bodov pomocou PCL	Bc. Michal Korbel	3DRecon\Sprint 1		
2652	Vytvorenie hrubého prototypu modulu GR	Bc. Michal Korbel	3DRecon\Sprint 1		
2653	Štúdium PCLVisualizer	Bc. Michal Loffler	3DRecon\Sprint 1	Štúdium triedy PCLVisualizer obsiahnutej v knižnici PCL.	
2648	Vizualizácia vstupných dát	Bc. Michal Loffler	3DRecon\Sprint 1	Štúdium možností vizualizácie vstupných - point cloud dát	
2649	Vizualizácia výstupných dát	Bc. Michal Loffler	3DRecon\Sprint 1	Štúdium možností vizualizácie výstupných - point cloud/mesh dát	
2687	Štúdium formátu Autodesk DXF	Bc. Michal Loffler	3DRecon\Sprint 1	Štúdium v praxi rozšíreného parametrického formátu Autodesk DXF (Drawing eXchange Format), ktorý sa používa na zakresľovanie plánov budov, príp. strojárskych a elektrotechnických výkresov.	

2688	Implementácia intuitívnejšieho spôsobu ovládania pohybu kamery	Bc. Michal Loffler	3DRecon\Sprint 1		
2677	Analýza a hrubý návrh modulu histogram normál	Bc. Robert Birkus	3DRecon\Sprint 1	Pre implementáciu histogramu je potrebné najprv zanalyzvať možné spôsoby výpočtu normál a následného návrhu štruktúry histogramu normál.	analyza; design; histogram; normal
2678	Výpočet normál point cloud-u pomocou PCL	Bc. Robert Birkus	3DRecon\Sprint 1	Cieľom tejto úlohy je naimplementovať výpočet normál point cloudu pomocou PCL funkcií.	dev; normal; PCL
2679	Vytvorenie histogramu normál	Bc. Robert Birkus	3DRecon\Sprint 1	Cieľom je implementácia histogramu normál podľa vypočítaných normál z point cloudu	dev; histogram; normal
2680	Zistenie smeru stien z point cloud-u pomocou histogramu	Bc. Robert Birkus	3DRecon\Sprint 1	Cieľom tejto úlohy je pomocou dominantných oblastí v histograme normál zistiť smer stien v naskenovanej miestnosti (point cloud).	analyza; design; dev; histogram; normal
2681	Zápisnica č. 4 (20151015)	Bc. Robert Birkus	3DRecon\Sprint 1	Vytvorenie zápisnice zo 4. stretnutia.	doku
2655	Analýza Data Collection and Preprocessing	Bc. Robert Karasek	3DRecon\Sprint 1		
2656	Analýza Inverse CSG	Bc. Robert Karasek	3DRecon\Sprint 1		
2657	Analýza Reconstructing 2D CSG Models	Bc. Robert Karasek	3DRecon\Sprint 1		
2658	Analýza Hough transformation	Bc. Robert Karasek	3DRecon\Sprint 1		
2659	Implementácia Hough transformation	Bc. Robert Karasek	3DRecon\Sprint 1		
2660	Zoznámenie sa s knižnicou a nájdenie vhodných funkcií pre	Bc. Robert Karasek	3DRecon\Sprint 1		

	Furukawu				

2. šprint

ID	Title	Assigned To	Iteration Path	Description	Tags
2732	Práca s normálami	Bc. Katarina Janeckova	3DRecon\Sprint 2	- prehľad metód v knižniciach - analýza a pochopenie - spolupráca s RB	analyza
2735	Ortogonálna projekcia	Bc. Katarina Janeckova	3DRecon\Sprint 2	- analýza a pochopenie - prehľad implementácie v knižniciach - edit: projekcia bodov na os z	analyza
2738	Mean shift	Bc. Katarina Janeckova	3DRecon\Sprint 2	- analýza a pochopenie metódy - prehľad o implementácii v knižniciach	analyza
2740	Implementácia využitia normál	Bc. Katarina Janeckova	3DRecon\Sprint 2	- implementácia metód pracujúcich s normálami	implementacia
2742	Implementácia projekcie	Bc. Katarina Janeckova	3DRecon\Sprint 2	Premietnutie bodov na os z.	implementacia
2743	Implementácia metódy mean shift	Bc. Katarina Janeckova	3DRecon\Sprint 2	Použiť metódy z PCL alebo OpenCV aplikujúce mean shift v našom projekte.	implementacia
2834	Implementácia architektúry projektu	Bc. Lukas Hudec	3DRecon\Sprint 2	Podľa architektonického návrhu projektu z posledného zasadnutia timu je potrebné vytvoriť nový projekt a implementovať navrhnuté triedy.	dev; implementacia
2725	Návrh architektúry	Bc. Lukas Hudec	3DRecon\Sprint 2	Esenciálnym krokom vo vývoji aplikácie je návrh jej architektúry, štruktúry tried a tried samotných, ktoré budú potrebné pre oo prístup k riešeniu a implementácií aplikácie.	analyza; design; doku
2726	Vytvorenie diagramov	Bc. Lukas Hudec	3DRecon\Sprint 2	Pre čitateľnosť a vyjadrenie návrhu architektúry sú potrebné UML diagramy. Najvhodnejšie pre naše použitie budú pravdepodobne	design; doku

				"class diagramy".	
2727	Implementácia Base class	Bc. Lukas Hudec	3DRecon\Sprint 2	Hlavná trieda, resp. "abstraktná trieda", ktorá určí štruktúru tried obsahujúcich metódy a bude slúžiť ako "interface" medzi načítaním dát a vizualizáciou spracovaných dát. Ďalšie info o tejto triede bude priložené a spracované v diagramoch.	dev; implementacia
2728	Vytvorenie testovacieho datasetu	Bc. Lukas Hudec	3DRecon\Sprint 2	Existujúci a 100% funkčný a použiteľný dataset je príliš zložitý a rozsiahly, čo je pre potreby vývoja a testovania aplikácie zbytočne priveľa. Z tohto dôvodu je potrebné z tohto datasetu (apartmán) vybrať len jednu miestnosť, na ktorej sa budú testovať funkčnosť metód rekonštrukcie.	dev; implementacia; test
2730	Úprava triedy načítania a správy dát	Bc. Lukas Hudec	3DRecon\Sprint 2	Upravenie triedy načítavania dát zo súborov. Momentálny stav je viac-menej procedurálneho rázu a pre priblíženie a zdokonalenie objektovo orientovaného prístupu je potrebné zmeniť ráz metód a štruktúry triedy. Táto trieda a jej inštancie budú obsahovať objekty metód 3D rekonštrukcie a pracovať nad dátami, ktoré načíta. Preto je potrebné ju upraviť aj na toto použitie a podľa implementácie "Base class".	dev; implementacia
2723	Doriesenie LAS	Bc. Martin Jurik	3DRecon\Sprint 2		
2724	Konverzia LAS->PCD	Bc. Martin Jurik	3DRecon\Sprint 2		
2866	Debugovanie pouzivanych libraries ku kniznici LAS	Bc. Martin Jurik	3DRecon\Sprint 2		
2830	Vytvorenie prototypu Grafického rozhrania	Bc. Martin Jurik	3DRecon\Sprint 2	Vytvorenie jednoduchého grafického rozhrania pre testovacie účely aplikácie. Načítanie súborov - multiselect Spustenie jednotlivých metód Prepínanie možností	

				výstupu	
2750	Analýza indexov	Bc. Michal Korbel	3DRecon\Sprint 2		
2751	Test example rozne PCD subory	Bc. Michal Korbel	3DRecon\Sprint 2		
2752	Test example rozne thresholds	Bc. Michal Korbel	3DRecon\Sprint 2		
2753	Komplet uprava webu	Bc. Michal Korbel	3DRecon\Sprint 2		
2831	Zápisnica č. 6 (20151029)	Bc. Michal Loffler	3DRecon\Sprint 2	Vytvorenie zápisnice zo 6. stretnutia.	doku
2741	Analýza možností knižnice CGAL	Bc. Michal Loffler	3DRecon\Sprint 2	Zistiť aké sú výhody knižnice CGAL oproti PCL, nájsť komponenty využiteľné pre náš projekt	
2739	Analýza možností manipulácie s kamerou	Bc. Michal Loffler	3DRecon\Sprint 2	Problém: vo východzom stave sa v PCLVisualizeri kamera otáča okolo nejakého bodu pred kamerou (začiatok súradnicovej sústavy?), čo je trochu neprirodzené. Riešenie: os otáčania kamery presunúť do miesta "stredu" kamery (t.j. ako sa napríklad ovládajú bežné FPS hry)	
2736	Uzamknutie osy kamery	Bc. Michal Loffler	3DRecon\Sprint 2	Problém: PCLVisualizer defaultne otáča kamerou v smere všetkých osí X,Y,Z Riešenie: uzamknutie rotovania kamery okolo osi Z	
2737	Analýza princípov interakcie s triedou PCLVisualizer pomocou myši a klávesnice	Bc. Michal Loffler	3DRecon\Sprint 2	Štúdium tried InteractorStyle, ktoré definujú spôsob interpretácie stláčania kláves a hýbania myšou.	
2733	Implementácia histogramu normál	Bc. Robert Birkus	3DRecon\Sprint 2	Po analýze a návrhu je potrebné naimplementovať histogram normál.	dev; histogram; normal
2731	Analýza štruktúry	Bc. Robert	3DRecon\Sprint	Cieľom je zanalyzovať štruktúru pcl::Normal, ako reprezentuje	analyza;

	pcl::Normal	Birkus	2	normály, čo všetko iné obsahuje.	normal
2729	Analýza možností PCL pre histogram normál	Bc. Robert Birkus	3DRecon\Sprint 2	Pred samotnou implementáciou histogramu normál je potrebné analyzovať všetky možnosti knižnice PCL, ktoré by nám uľahčili prácu pri vytváraní histogramu normál.	analyza; histogram; normal
2716	Návrh histrogramu normál	Bc. Robert Birkus	3DRecon\Sprint 2	Je potrebné navrhnúť štruktúru histogramu normál.	design; histogram; normal
2794	Analýza výpočtu normál v PCL	Bc. Robert Birkus	3DRecon\Sprint 2	Je potrebné analyzovať možnosti knižnice PCL pre výpočet normál.	analyza; normal; PCL
2793	Zápisnica č.5 (20151022)	Bc. Robert Karasek	3DRecon\Sprint 2	Vytvorenie zápisnice z 5. stretnutia.	doku
2746	Analýza rezov pre metódu Furukawa	Bc. Robert Karasek	3DRecon\Sprint 2		
2747	Porovnanie implementácií Hougha pre Furukawu	Bc. Robert Karasek	3DRecon\Sprint 2		
2749	Testovanie Hougha na datasetoch	Bc. Robert Karasek	3DRecon\Sprint 2		

3. šprint

ID	Title	Assigned To	Iteration Path	Description	Tags
2973	Nájdenie bodov s normálou podobnou osi z	Bc. Katarina Janeckova	3DRecon\Sprint 3	Po nájdení normál pre všetky body vybrať len tie body, ktoré majú podobnú normálu s osou z.	implementacia
2975	Dokončenie implementácie výpočtu normál	Bc. Katarina Janeckova	3DRecon\Sprint 3		implementacia
2976	Implementácia projekcie bodov na os z	Bc. Katarina Janeckova	3DRecon\Sprint 3	Projekcia bodov, ktoré majú normálu podobnú s osou z, na os z.	implementacia

2982	Vytvorenie kostry celkovej dokumentácie	Bc. Katarina Janeckova	3DRecon\Sprint 3	Vytvoriť základný dokument na dopĺňanie pre všetkých členov.	doku
2983	Vytvorenie kapitoly Úvod (inžinierske dielo)	Bc. Katarina Janeckova	3DRecon\Sprint 3	Napísať úvodnú kapitolu dokumentácie k inžinierskemu dielu.	doku
2986	Vytvorenie podkapitoly Dátový model (inžinierske dielo)	Bc. Katarina Janeckova	3DRecon\Sprint 3	Do dokumentácie k inžinierskemu dielu vytvoriť dátový model aj s opisom.	doku
2988	Zdokumentovanie metódy 1 (inžinierske dielo)	Bc. Katarina Janeckova	3DRecon\Sprint 3	Do dokumentácie k inžinierskemu dielu napísať kapitolu o metóde 1, ktorá bude obsahovať nasledovné podkapitoly: analýza, návrh, implementácia a testovanie.	doku
2993	Vytvorenie podkapitoly s odkazmi na priložené e-dokumenty (inžinierske dielo)	Bc. Katarina Janeckova	3DRecon\Sprint 3	Do dokumentácie k inžinierskemu dielu napísať časť, ktorá bude obsahovať zoznam priložených e-dokumentov a ich opis.	doku
2995	Vytvorenie kapitoly Role členov tímu a podiel práce (riadenie)	Bc. Katarina Janeckova	3DRecon\Sprint 3	Do dokumentácie k riadeniu napísať kapitolu s vysvetlením jednotlivých zodpovedností členov tímu a tiež zapísať ich podiel práce na častiach dokumentácie.	doku
2998	Vytvorenie kapitoly Používané metodiky (riadenie)	Bc. Katarina Janeckova	3DRecon\Sprint 3	Spísať zoznam používaných metodík s krátkym opisom ku každej do dokumentácie k riadeniu.	doku
2999	Vytvorenie kapitoly Zoznam kompetencií tímu (riadenie)	Bc. Katarina Janeckova	3DRecon\Sprint 3	Do dokumentácie k riadeniu opísať kompetencie tímu.	doku
3000	Vytvorenie kapitoly Export evidencie úloh + urobiť exporty z TFS (riadenie)	Bc. Katarina Janeckova	3DRecon\Sprint 3	Vytvoriť v dokumentácii k riadeniu kapitolu exportov z TFS.	doku
3058	Zápisnica č. 8 (20151112)	Bc. Katarina Janeckova	3DRecon\Sprint 3	Vytvorenie zápisnice zo stretnutia č. 8	doku

3042	Vytvorenie syntetických datasetov	Bc. Lukas Hudec	3DRecon\Sprint 3	Vytvorenie testovacích datasetov pre účely vývoja.	
2994	Vytvorenie kapitoly Úvod (riadenie)	Bc. Lukas Hudec	3DRecon\Sprint 3	Napísanie úvodnej kapitoly do dokumentácie k riadeniu.	doku
2987	Vytvorenie podkapitoly Diagram tried a moduly (inžinierske dielo)	Bc. Lukas Hudec	3DRecon\Sprint 3	Do dokumentácie k inžinierskemu dielu vytvoriť diagram tried aj s opisom.	doku
2985	Vytvorenie podkapitoly Architektúra (inžinierske dielo)	Bc. Lukas Hudec	3DRecon\Sprint 3	Návrh architektúry, pridanie diagramov, rozdelenie a opis modulov	
2939	Analýza segmentácie objektov z 3D pointCloudu	Bc. Lukas Hudec	3DRecon\Sprint 3		
2984	Vytvorenie kapitoly Globálne ciele pre ZS (inžinierske dielo)	Bc. Martin Jurik	3DRecon\Sprint 3	Do dokumentácie k inžinierskemu dielu spísať ciele, ktoré sme si určili na ZS. Premyslieť ciele, ktoré je možné stihnúť za ZS	doku
2962	Vyriesenie exception s boost library	Bc. Martin Jurik	3DRecon\Sprint 3	Odstranenie exception odchytenu programom pri return 0;	
3011	implementacia multiselect dialogu a otestovanie	Bc. Martin Jurik	3DRecon\Sprint 3	Implementacia multiselect dialogu pre otvorenie viacerch suborov a navrat vektora stringov absolutnych ciest pre metody nacitania point cloudov.	
3012	Implementacia eventov a handlerov pre vyber a spustenie metody	Bc. Martin Jurik	3DRecon\Sprint 3	Implementovanie vyberu metody v gui a naprogramovanie funkcii pre priestor implementacie jednotlivych metod.	
3059	Zápisnica č. 7 (20151116)	Bc. Michal Korbel	3DRecon\Sprint 3		
2991	Zdokumentovanie metódy 2 (inžinierske dielo)	Bc. Michal Korbel	3DRecon\Sprint 3	Do dokumentácie k inžinierskemu dielu napísať kapitolu o metóde 4, ktorá bude obsahovať nasledovné podkapitoly: analýza, návrh, implementácia a testovanie.	doku

2997	Vytvorenie kapitoly Sumarizácie šprintov (riadenie)	Bc. Michal Korbel	3DRecon\Sprint 3	Do dokumentácie k riadeniu napísať kapitolu, v ktorej bude zhrnutie všetkých šprintov. Podľa úloh v TFS spraviť spätný opis/retrospektíva jednotlivých šprintov. Úspechy, neúspechy = "stop doing, start doing, keep doing" z TFS z časti Reporty vytiahnuť dáta o šprintoch (všetko možné čo sa dá ;))	doku
2992	Nastavenie stredu rotácie	Bc. Michal Loffler	3DRecon\Sprint 3	Problém: Knižnice PCL, resp. VTL uplatňujú vo svojich východzích InteractorStyle-triedach správanie, pri ktorom sa kamera rotuje okolo bodu umiestneného v určitej vzdialenosti pred kamerou (focalPoint). Prirodzeným spôsobom je však otáčanie okolo samotného "ťažiska kamery" - podobne ako človek otáča svojou hlavou. Riešenie: Zadefinovať bod otáčania kamery do jej stredu resp. do zanedbateľne malej vzdialenosti pred ňu.	
2979	Posun doľava a doprava pomocou kláves A a D	Bc. Michal Loffler	3DRecon\Sprint 3	Problém: Vo východzom nastavení PCL(VTK) je potrebné pre posun (pan) kamery držať stlačené (prostredné/pravé?) tlačidlo myši a súčasne hýbať myšou. Riešenie: Namapovať pohyby doprava a doľava na klávesy A a D.	
2980	Posun dopredu/dozadu pomocou klaves W a S	Bc. Michal Loffler	3DRecon\Sprint 3	Problém: Vo východzom nastavení PCL(VTK) je potrebné pre pohyb kamery dopredu a dozadu točiť kolečkom myši alebo držať stlačené pravé tlačidlo myši a hýbať ňou dopredu/dozadu. Riešenie: Namapovať pohyby dopredu a dozadu na klávesy W a S.	
2981	Zrušenie nutnosti mať stále stlačené tlačidlo	Bc. Michal Loffler	3DRecon\Sprint 3	Problém: Ak chceme natáčať pohľad kamery, musíme pritom držať stlačené ľavé tlačidlo myši,	

	myši pre rotovanie			čo je trocha nepohodlné. Riešenie: prvým kliknutím sa aktivuje natáčanie kamery pohybom myšou druhým kliknutím sa deaktivuje	
3057	Zdokumentovanie vizualizácie (inžinierske dielo)	Bc. Michal Loffler	3DRecon\Sprint 3	Do dokumentácie k inžinierskemu dielu napísať kapitolu o vizualizácie, ktorá bude obsahovať nasledovné podkapitoly: analýza, návrh, implementácia a testovanie.	doku
2996	Vytvorenie kapitoly Aplikácie manažmentov (riadenie)	Bc. Michal Loffler	3DRecon\Sprint 3	Kapitola do dokumentácie k riadeniu s opisom realizácie jednotlivých činností potrebných pre riadenie projektu, procesu a produktu. Manažment komunikácie a ľudských zdrojov (Róbert Karásek) - stretnutia tímu, komunikačné nástroje, nástroje na zdieľanie obsahu Manažment rozvrhu a rozsahu projektu (Lukáš Hudec) Manažment plánovania projektu (Lukáš Hudec) Manažment kvality (Mišo Löffler) - (zaistenie že projekt jeho výsledky uspokojuje potreby pre ktoré sa vytvoril/inicioval) Manažment rizík (Mišo Löffler) Manažment integrácie a podpory vývoja (Martin Jurík) - (zaistenie že rôzne elementy projektu sú správne koordinované)	doku
3004	Analýza možností vizualizácie 2D histogramu	Bc. Robert Birkus	3DRecon\Sprint 3	Cieľom je zistiť možnosti vizualizácie 2D histogramu a zvoliť si ten najvhodnejší pre náš problém.	analyza
3005	Implementácia vizualizácie histogramu	Bc. Robert Birkus	3DRecon\Sprint 3	Cieľom je implementácia vizualizácie histogramu.	dev
2989	Zdokumentovanie metódy 4 (inžinierske dielo)	Bc. Robert Birkus	3DRecon\Sprint 3	Do dokumentácie k inžinierskemu dielu napísať kapitolu o metóde 2, ktorá bude obsahovať nasledovné podkapitoly: analýza, návrh,	doku

				implementácia a testovanie.	
2990	Zdokumentovanie metódy 3 (inžinierske dielo)	Bc. Robert Karasek	3DRecon\Sprint 3	Do dokumentácie k inžinierskemu dielu napísať kapitolu o metóde 3, ktorá bude obsahovať nasledovné podkapitoly: analýza, návrh, implementácia a testovanie.	doku
2941	Priprava Testovacich scenarov	Bc. Robert Karasek	3DRecon\Sprint 3		
2942	Príprava konvencií pre generovanie technickej dokumentácie	Bc. Robert Karasek	3DRecon\Sprint 3	Príprava základných konvencií a návodu ako vygenerovať technickú dokumentáciu	
3001	Vytvorenie metodiky pre komunikáciu a pre commiting/branching	Bc. Robert Karasek	3DRecon\Sprint 3	Spísanie metodiky pre komunikáciu členov tímu v aplikácii Slack Metodika pre správny commiting + merging do projektu, formát správ commitu - premyslieť aby to dobre vyzeralo a malo informačnú hodnotu pre ostatných členov tímu.	doku

4. šprint

ID	Title	Assigned To	Iteration Path	Description	Tags
2744	Implementácia extrakcie rezu	Bc. Katarina Janeckova	3DRecon\Sprint 4		implementacia
3130	Implementácia metódy mean shift	Bc. Katarina Janeckova	3DRecon\Sprint 4		
3133	Funkčná integrácia podporných modulov	Bc. Lukas Hudec	3DRecon\Sprint 4	Prepojiť existujúce a vytvorené moduly s hlavným GUI	
3125	Zápisnica č.9 (20151120)	Bc. Lukas Hudec	3DRecon\Sprint 4	Spísanie zápisnice z 9. stretnutia a jej sprístupnenie pre celý tím.	
3120	Reštrukturalizácia	Bc. Lukas	3DRecon\Sprint	Z dôvodu vyššej efektivity a lepšieho rozvrhnutia dizajnu	implementacia

	architektúry	Hudec	4	architektúry projektu je potrebné zmeniť implementáciu modulov, rozbiť zložité štruktúry podľa tematického zamerania a znížiť závislosť na includoch.	
3140	Projekcia 3D primitív na 2D zobrazenie pôdorysu	Bc. Lukas Hudec	3DRecon\Sprint 4	Pohľad zhora - zmena 3D na 2D zobrazenie a príprava na spracovanie pôdorysu	
3175	Integrácia metód do gui	Bc. Lukas Hudec	3DRecon\Sprint 4	Integrácia spúšťania metód cez gui.	
3246	Zápisnica č.10 (20151126)	Bc. Martin Jurik	3DRecon\Sprint 4		
3013	Implementacia spustenia vizualicazie v gui projekte	Bc. Martin Jurik	3DRecon\Sprint 4	Riesenie vtk erroru a zatvaranie okna vizualizacie z gui	
3134	Ošetrenie situácie keď je kurzor mimo okna vizualizácie	Bc. Michal Loffler	3DRecon\Sprint 4		
3126	Implementácia posunu nahor a nadol (elevation)	Bc. Michal Loffler	3DRecon\Sprint 4		
3127	Implementácia rotácie kamery	Bc. Michal Loffler	3DRecon\Sprint 4		
3128	Implementácia interakcie pomocou 3D myši (knižnica VTK)	Bc. Michal Loffler	3DRecon\Sprint 4	1. prekompilovanie knižnice VTK 2. nastavenie vtkRenderWindowInteractor 3. implementácia event observerov	
3129	Analýza možností knižnice VTK pre ovládanie 3D myšou	Bc. Michal Loffler	3DRecon\Sprint 4		
3184	Oboznámenie sa s 3Dconexxion SDK	Bc. Michal Loffler	3DRecon\Sprint 4		
3121	Navrhnúť spôsob označovania bodov	Bc. Robert Birkus	3DRecon\Sprint 4		design

3122	Implementácia označovania bodov	Bc. Robert Birkus	3DRecon\Sprint 4	Implementovanie označovania všetkých bodov v point cloude pomocou dominantných normál. Podľa normály jednotlivých bodov sa priradí bod k najpodobnejšej dominantnej normály	dev
3123	Otestovanie presnosti a správnosti označovania bodov	Bc. Robert Birkus	3DRecon\Sprint 4	Je potrebné otestovať efektívnosť označovania bodov v pointcloude vizuálne z hľadiska segmentácie bodov reprezentujúcich stenu miestnosti.	test
3006	Analýza možností zisťovania dominantných oblastí v 2D histograme	Bc. Robert Birkus	3DRecon\Sprint 4	Cieľom je zanalyzovať rôzne algoritmy vhodné pre hľadanie dominantných oblastí v 2D histograme. Po analýze sme sa rozhodli, že najvhodnejší algoritmus pre daný problém bude mean-shift. A keďže knižnice, ktoré používame obsahujú implementáciu mean-shiftu len pre špecifické úlohy je potrebné si navrhnúť a naimplementovať vlastný mean-shift pre náš problém.	analyza
3007	Návrh algoritmu pre nájdenie dominantných oblastí	Bc. Robert Birkus	3DRecon\Sprint 4	Cieľom je navrhnúť algoritmus pre nájdenie dominantných oblastí v histogram. Predpokladá sa použitie mean-shift algoritmu.	design; histogram
3008	Implementácia algoritmu pre nájdenie dominantných oblastí	Bc. Robert Birkus	3DRecon\Sprint 4	Implementácia navrhnutého algoritmu pre nájdenie dominantných oblastí v histograme.	dev; histogram
3009	Otestovanie správnosti algoritmu pre nájdenie dominantných oblastí	Bc. Robert Birkus	3DRecon\Sprint 4	Cieľom je otestovať správnosť a vhodnosť algoritmu pre nájdenie dominantných oblastí v 2D histograme.	histogram; test
3010	Nasadenie implementácie výpočtu a vizualizácie histogramu normál	Bc. Robert Birkus	3DRecon\Sprint 4	Cieľom je nasadenie implementácie histogramu na náš hlavný projekt.	deploy; dev

	na hlavný projekt				
3110	Vytvorenie stručných pravidiel pre komentovanie	Bc. Robert Karasek	3DRecon\Sprint 4		
3111	Vytvorenie prvých správne okomentovaných súborov	Bc. Robert Karasek	3DRecon\Sprint 4	Jedná sa o správne anotácie v komentároch, podľa ktorých sa môžete držať.	

5. šprint

ID	Title	Assigned To	Iteration Path	Description	Tags
3199	Práca na dokumentácii	Bc. Katarina Janeckova	3DRecon\Sprint 5	Spísanie nových vecí v rámci projektu do dokumentácie v častiach, v ktorých najviac prispievam.	doku
3206	Finálne spracovanie dokumentácie k ZS	Bc. Katarina Janeckova	3DRecon\Sprint 5	Doplnenie potrebných vecí, konečné úpravy, kompletizácia.	doku
3207	Zjednodušenie pridania nového objektu steny	Bc. Lukas Hudec	3DRecon\Sprint 5	Vzhľadom na implementačnú zložitosť vytvorenia nového objektu steny, je potrebná jeho reštrukturalizácia a zjednodušenie po maximálnu abstrakciu - overloading konštruktora.	
3200	Práca na dokumentácii	Bc. Lukas Hudec	3DRecon\Sprint 5	Spísanie nových vecí v rámci projektu do dokumentácie v častiach, v ktorých najviac prispievam. Doplnenie a úprava metodiky evidencie úloh Doplnenie a úpravy dokumentácie inžinierskeho diela v časti architektúra podľa posledných zmien Dokumentácia Doxygen komentáre do kódu	doku
3139	Primitívne 3D zobrazenie primitív	Bc. Lukas Hudec	3DRecon\Sprint 5	prvý pokus pre zobrazenie primitív bude ich jednoduché 3D premietnutie pcl vizualizérom	
3216	Implementovanie funkcionality check boxov	Bc. Martin Jurik	3DRecon\Sprint 5	Implementovanie vyberu point cloudov na zaklade vybranych check boxov	
3217	Rozsirenie GUI o	Bc. Martin	3DRecon\Sprint		

	featuru pre Birkyho	Jurik	5		
3201	Práca na dokumentácii	Bc. Martin Jurik	3DRecon\Sprint 5	Spísanie nových vecí v rámci projektu do dokumentácie v častiach, v ktorých najviac prispievam.	doku
3202	Práca na dokumentácii	Bc. Michal Korbel	3DRecon\Sprint 5	Spísanie nových vecí v rámci projektu do dokumentácie v častiach, v ktorých najviac prispievam. Globalna retrospektiva, sumarizacia sprintov a opis GR.	doku
3203	Práca na dokumentácii	Bc. Michal Loffler	3DRecon\Sprint 5	Spísanie nových vecí v rámci projektu do dokumentácie v častiach, v ktorých najviac prispievam.	doku
3212	Zmenšenie veľkosti kroku pri pohybe do strán	Bc. Michal Loffler	3DRecon\Sprint 5		
3213	Zjemnenie plynulosti pohybu do strán	Bc. Michal Loffler	3DRecon\Sprint 5	Zatiaľ sa neporadilo naimplementovať dôvod: pri použití vtkRepeatingTimer je príliš dlhá odozva na stláčanie kláves. (vtkRepeatingTimer je automatický časovač, ktorý po stlačení klávesy pre pohyb začne v pravidelných krátkych intervaloch generovať pohyb o malý kúsok dopredu/ do strany)	
3204	Práca na dokumentácii	Bc. Robert Birkus	3DRecon\Sprint 5	Spísanie nových vecí v rámci projektu do dokumentácie v častiach, v ktorých najviac prispievam.	doku
3243	Zápisnica č.11 (20151203)	Bc. Robert Birkus	3DRecon\Sprint 5	Spísanie zápisnice z agile stretnutia tímu	doku
3193	Návrh a implementácia metódy pre nájdenie dominantných oblastí v histograme pomocou maxím	Bc. Robert Birkus	3DRecon\Sprint 5	Je potrebné navrhnúť a implementovať metódu pre nájdenie dominantných oblastí v histogram normál pomocou hľadania maxím.	design; dev
3194	Kontrola správneho fungovania metódy	Bc. Robert Birkus	3DRecon\Sprint 5	Je potrebné overiť, či naimplementovaná metóda vykonáva požadovanú funkcionalitu.	test
3205	Práca na dokumentácii	Bc. Robert Karasek	3DRecon\Sprint 5	Spísanie nových vecí v rámci projektu do dokumentácie v častiach, v ktorých najviac prispievam. Metodika technickej dokumentácie Metodika testovania	doku

				Príprava a vygenerovanie technickej dokumentácie	

„novoročný“ šprint

ID	Title	Assigned To	Iteration Path	Description	Tags
2623	Analýza binary inside/outside segmentation	Bc. Katarina Janeckova	3DRecon\Novoročný šprint	-pochopenie princípu binárnej inside/outside segmentácie - prehľad o implementácii v knižniciach - urobiť prehľad o danej metóde pre zvyšok tímu	analyza
3132	Analýza outline simplification	Bc. Katarina Janeckova	3DRecon\Novoročný šprint		
3208	Hľadanie priesečníkov rovín - body ohraničenia steny	Bc. Lukas Hudec	3DRecon\Novoročný šprint		
3124	Zistenie roviny z bodov	Bc. Lukas Hudec	3DRecon\Novoročný šprint	Pomocou štatistiky a metód analytickej geometrie získať funkčný predpis aproximovanej roviny, ktorá predstavuje na abstraktnej úrovni priestor v ktorom sa nachádzajú body rovnakého príznaku predstavujúce rovinu, ktorú hľadáme.	implementacia
3137	Analýza binary inside/outside segmentation	Bc. Martin Jurik	3DRecon\Novoročný šprint		
3138	Analýza outline simplification	Bc. Martin Jurik	3DRecon\Novoročný šprint		
3014	Nájdenie hraničných čiar objektu pomocou Houghovej metódy	Bc. Michal Korbel	3DRecon\Novoročný šprint		
3015	Segmentovanie jednej plochy z objektu podľa rezov roviny	Bc. Michal Korbel	3DRecon\Novoročný šprint		
3118	Zistenie roviny z bodov	Bc. Michal Korbel	3DRecon\Novoročný šprint	- Ransac model plane - výpočet koeficientov roviny (ax +by + cz +d = 0) - preloženie roviny v smere plochy	
3214	Výpomoc pri hľadaní priesečníkov rovín	Bc. Michal Loffler	3DRecon\Novoročný šprint		
3259	Návrh growing region algoritmu podľa označení bodov oblaku	Bc. Robert Birkus	3DRecon\Novoročný šprint		design
3270	Implementácia	Bc. Robert	3DRecon\Novoročný		dev

	navrhnutého growing region algoritmu	Birkus	šprint		
3271	Zmeny v architektúre	Bc. Robert Birkus	3DRecon\Novoročný šprint	V triede closedSpace sa jednotlivé dáta XYZ, RGB a Normal ukladávajú v jednotlivých premenných. Gui automaticky rozpozná, že aké dáta obsahuje načítaný súbor a podľa toho naplní vhodné premenné. Návrh zmien som konzultoval s Lukášom Hudecom	design; dev
3195	Nasadenie fungujúcich častí rekonštrukčnej metódy č. 4 do hlavného projektu	Bc. Robert Birkus	3DRecon\Novoročný šprint	Hotové časti metódy č.4 je potrebné nasadiť do hlavného projektu.	deploy
3196	Nastavenie knižnice OpenCV 3.0.0 v debug móde v hlavnom projekte	Bc. Robert Birkus	3DRecon\Novoročný šprint	V hlavnom projekte sa knižnica OpenCV 3.0.0 bije s knižnicou boost a každá z knižníc si vyžaduje iné nastavenia. Je potrebné vyriešiť, aby v projekte fungovali obi dve knižnice naraz. Táto úloha je vykonávaná v spolupráci s Martinom Juríkom	configuration
3112	Štúdium Unit Testov	Bc. Robert Karasek	3DRecon\Novoročný šprint		

6. šprint

ID	Title	Assigned To	Iteration Path	Description	Tags
3314	Dokument k práci s branchami	Bc. Lukas Hudec	3DRecon\Sprint 6	Napísať dokument ako a čo commitovať do jednotlivých novo vytvorených branchov.	
3317	Doplnenie funkcionality do triedy Wall	Bc. Lukas Hudec	3DRecon\Sprint 6	Doplnenie obsahu Cloudu do triedy Wall a primitívy	
3302	Vytvorenie tried	Bc. Martin Jurik	3DRecon\Sprint 6		
3303	Default xml config	Bc. Martin Jurik	3DRecon\Sprint 6	Vytvorenie defaultného cfg.xml	
3324	Zistovanie hran	Bc. Michal Korbel	3DRecon\Sprint 6	Zistenie hran v pointcloude.	
3309	Prehľad formátov CAD	Bc. Michal Loffler	3DRecon\Sprint 6		
3326	Návrh algoritmu	Bc. Robert Birkus	3DRecon\Sprint 6		design
3327	Implementácia RANSAC algoritmu pre zistenie koeficientov plôch	Bc. Robert Birkus	3DRecon\Sprint 6		dev
3143	Build Google Test pre	Bc. Robert	3DRecon\Sprint	Príprava buildu google testov pre	

	projekt	Karasek	6	náš projekt v prípade viacerých chýb => ostane sa pri MS assertoch	

7. šprint

ID	Title	Assigned To	Iteration Path	Description	Tags
3384	Zrotovať dataset schody	Bc. Katarina Janeckova	3DRecon\Sprint 7	Vycentrovať dataset, aby bol kolmý k osiam	implementacia
3318	Hľadanie outliers	Bc. Lukas Hudec	3DRecon\Sprint 7	hľadanie outliers, ktoré môžu byť objektom na segmentovanej stene	
3359	Zápisnica č.13 20160224	Bc. Lukas Hudec	3DRecon\Sprint 7	Zápisnica č.2 LS	
3361	Doplniť atribúty koeficientov roviny do triedy Wall	Bc. Lukas Hudec	3DRecon\Sprint 7	Vzhľadom na pokrok vo vývoji a udržiavanie súdržnosti segmentovaných objektov, je potrebné pridať do objektu steny aj koeficienty všeobecnej rovnice roviny, ktorá prechádza touto stenou.	
3382	Zakladne konfiguracie	Bc. Martin Jurik	3DRecon\Sprint 7	Zozbieranie zakladnych info od kolegov co chce mat kto ako nakonfigurovane, zapisat to do xml suboru	
3383	Nastavenie tried a metod	Bc. Martin Jurik	3DRecon\Sprint 7	Pridanie konfiguracnych informacii do tried a metod	
3364	Vymedzenie priestoru hľadaných priesečníkov	Bc. Michal Korbel	3DRecon\Sprint 7	Pointcloud sa ohraničí, aby sa nehľadali priesečníky mimo tohto priestoru.	
3365	Hľadanie priesečníkov pomocou 3 rovín	Bc. Michal Korbel	3DRecon\Sprint 7	Nájdenie priesečníka z 3 rovín a vytvorenie ohraničenia okolo priesečníka. V ohraničení sa získajú body z každej zo stien a pomocou nich sa priradí priesečník k jednotlivým stenám.	
3362	Kontaktovať tvorcov sgCore SDK	Bc. Michal Loffler	3DRecon\Sprint 7	- opýtať sa, či nám sprístupnia ich SDK pre náš tímákhttp://www.geometros.com/ -- 1. tak nakoniec som sa v dokumentácii dočítal, že free verzia toho SDKčka je obmedzená právae tým, že neumožňuje žiaden výstup do CAD formátov 2. napísal som im e-mail že či nám poskutnú licenciu pre naše potreby - odpovedali mi iba niečo v zmysle: free verzia umožňuje uloženie poľa "trianglov" do OBJ (non-CAD) súboru . No ale my predsa nemáme žiadne triangle-mesh dáta.. 3. tak ma napadá že na vytovrenie modelu miestnosti použijem CSG operácie ktoré to SDK poskytuje, následne	

				model "ztriangulizujem" a potom uložím do OBJ. !! 4. zistil som ze funkcia na ktoru ma tvorcovia odkazali nam je na nic :D , pretoze ona nevytvara OBJ subor, ale len nejaku instanciu triedy sgObject alebo take nieco... no ale nakoniec som zistil ze export do DXF funguje napriek tomu ze by nemal..	
3310	Začlenenie knižnice pre výstup do 3D modelu	Bc. Michal Loffler	3DRecon\Sprint 7		
3355	Optimalizácia segmentovania oblaku bodov (vyriešenie chybnej segmentácie na datasete schodov)	Bc. Robert Birkus	3DRecon\Sprint 7		design; dev
3391	Doxygen komentáre a upráva kódu metódy 4	Bc. Robert Birkus	3DRecon\Sprint 7		dev
3312	Úprava branchov	Bc. Robert Karasek	3DRecon\Sprint 7	odstránenie chýb a úprava branchov podľa metodiky	
3313	Doxygen komentáre	Bc. Robert Karasek	3DRecon\Sprint 7	Úprava komentárov a vygenerovanie doku	
3316	Administrácia commitov	Bc. Robert Karasek	3DRecon\Sprint 7	Spravovanie nových commitov medzi branchami a kontrola nového kódu * správna funkčnosť * nenarúšanie iných modulov	

8. šprint

ID	Title	Assigned To	Iteration Path	Description	Tags
3300	Zistiť funkčnosť meanshift vzhľadom na vstup so schodmi	Bc. Katarina Janeckova	3DRecon\Sprint 8	Zistiť, či metóda číslo 1 správne určuje miesta rezov na vstupe so schodmi.	dev; implementacia
3436	Vizualizácia histogramu rozmiestnenia outliers	Bc. Lukas Hudec	3DRecon\Sprint 8	Vytvorený histogram rozloženia outliers vzhľadom na rovinu steny je potrebné vizualizovať, aby bola možná ďalšia analýza outliers.	
3469	Umožniť vizualizáciu orginálnych dát bez potreby spustenia rekonštrukcie	Bc. Lukas Hudec	3DRecon\Sprint 8	V prípade, že si chceme zobraziť iba orginálny dataset je potrebné pustiť najprv rekšontrukciu, čo je zbytočné.	dev
3470	Vizualizácia originál dát hneď po načítaní	Bc. Lukas Hudec	3DRecon\Sprint 8	Možnosť vizualizovať načítané dáta hneď po načítaní, nie až po spustení výpočtu metódy.	

3438	Analyza moznosti upravy datasetu	Bc. Martin Jurik	3DRecon\Sprint 8	Zistenie moznosti ako odstranit z datasetu zasumenie aby prebehla lepsia segmentacia	
3439	Implementácia upravy datasetu	Bc. Martin Jurik	3DRecon\Sprint 8	Implementacia a otestovanie odstranenia sumu v datasete vybranou metodou	
3409	Zápisnica č.14 20160203	Bc. Robert Birkus	3DRecon\Sprint 8	Zápisnica č.3 LS	doku
3418	Vytvorenie datasetu pre skúšanie MOG	Bc. Robert Birkus	3DRecon\Sprint 8		dev
3421	Vytvoriť možnosť dynamického počtu dominantných smerov	Bc. Robert Birkus	3DRecon\Sprint 8		dev
3423	Optimalizácia rekonštrukcie pomocou vstupných parametrov	Bc. Robert Birkus	3DRecon\Sprint 8		dev
3424	Čítanie vstupných parametrov z konfiguračného súboru	Bc. Robert Birkus	3DRecon\Sprint 8		dev
3426	Vizualizácia dominantných smerov	Bc. Robert Birkus	3DRecon\Sprint 8		dev
3411	Spracovať code review	Bc. Robert Karasek	3DRecon\Sprint 8	+ Opravovať aj kód, komentáre atď. Review, ktorý ste spravili vo dvojici mimo mňa poslať so stručným opisom na slack do utorka večera najneskôr	
3412	Testy	Bc. Robert Karasek	3DRecon\Sprint 8	Skúsiť vypracovať testy pre metódu a oboznámiť s tým kolegov	

9. šprint

ID	Title	Assigned To	Iteration Path	Description	Tags
3420	Vypočítanie otočenia datasetu z dominantných smerov	Bc. Katarina Janeckova	3DRecon\Sprint 9	Spolupráca s Robert B.	dev
3491	Segmentácia vstupných dát - získanie najväčšej steny	Bc. Lukas Hudec	3DRecon\Sprint 9	Potreba nájsť vhodné parametre, ktoré by dokázali segmentovať najväčšie plochy = steny	
3492	Histogram outliers nájdených bodov okolo odhaleného segmentu	Bc. Lukas Hudec	3DRecon\Sprint 9	Z okolia získaného segmentu, je potrebné vytvoriť histogram a nájsť smerodajnú odchýlku, podľa ktorej sa určí "roughness" datasetu a teda kvalita senzora.	
3493	Extrakcia metódy pre získanie koeficientov roviny spolu s outliers a inliers	Bc. Lukas Hudec	3DRecon\Sprint 9	Vzhľadom na refaktoring kódu, je potrebné extrahovať momentálne nesamostatnú časť kódu pre hľadanie koeficientov,	

				inliers a outliers tak, aby mohla byť volaná ako samostatná funkcia	
3496	Uprava v open file dialogu pre XML	Bc. Martin Jurik	3DRecon\Sprint 9	Doplnenie akceptovatelnych pripon pre otvaranie konfiguracnych suborov	
3497	Doplnenie do gui moznost odstranit sum z datasetu	Bc. Martin Jurik	3DRecon\Sprint 9	implementacia funkcie pre odstranenie sumu z datasetu, predpocitanie datasetu a jeho ulozenie pre dalsie spracovavanie	
3502	Implementácia metód pre pridávanie geom. útvarov	Bc. Michal Loffler	3DRecon\Sprint 9		
3435	Validácia priesečníkov rovín ako rohov stien	Bc. Michal Loffler	3DRecon\Sprint 9		
3485	Vytvorenie 3D histogramu normál	Bc. Robert Birkus	3DRecon\Sprint 9		dev
3487	Rozbehanie CUDA na školskom počítači	Bc. Robert Birkus	3DRecon\Sprint 9		configuration
3488	Rozbehanie svojej gSLIC implementácie na školskom PC	Bc. Robert Birkus	3DRecon\Sprint 9		dev
3425	Nájdenie dominantných smerov v 3D histograme normál	Bc. Robert Birkus	3DRecon\Sprint 9		dev
3354	Získať vysegmentované oblaky bodov pre všetky steny	Bc. Robert Birkus	3DRecon\Sprint 9		design; dev
3419	Rozdelenie metódy do funkcií	Bc. Robert Birkus	3DRecon\Sprint 9		dev
3499	Code review	Bc. Robert Karasek	3DRecon\Sprint 9		
3501	Doxygen	Bc. Robert Karasek	3DRecon\Sprint 9	Ďalšia úprava pre doxygen	

10. šprint

ID	Title	Assigned To	Iteration Path	Description	Tags
3505	Adaptívne okno pre meanshift	Bc. Katarina Janeckova	3DRecon\Sprint 10	Zmena pevného okno v meanshift na adaptívne.	implementacia
3298	Zmena zobrazenia lokácie rezov	Bc. Katarina Janeckova	3DRecon\Sprint 10	Namiesto jedného bodu určiť miesto rezu celou rovinou.	dev; implementacia
3304	Ďalej pracovať na hľadaní rezov	Bc. Katarina Janeckova	3DRecon\Sprint 10	Opraviť metódu tak, aby správne našla miesta rezov aj pre iné dáta ako	dev; implementacia

				je kocka. Prioritne sa zamerať na schody.	
3568	Vizualizácia bodov premietnutých na os z	Bc. Katarina Janeckova	3DRecon\Sprint 10		implementacia
3569	Určenie správnych parametrov pre meanshift	Bc. Katarina Janeckova	3DRecon\Sprint 10		implementacia
3570	Vizualizácia priebehu a výsledkov meanshift	Bc. Katarina Janeckova	3DRecon\Sprint 10		implementacia
3353	Vykresliť roviny stien správnej veľkosti a správnou pozíciou pomocou koeficientov	Bc. Lukas Hudec	3DRecon\Sprint 10	vpodstate iná úloha, ale splnená vrámci druhého zadania	design; dev
3541	Nájdenie vhodných parametrov	Bc. Lukas Hudec	3DRecon\Sprint 10	Pri rôznych parametroch je výstup metódy regiongrowing vždy iný a vždy inak dlho trvá. Ide o pokus nastaviť benevolentné parametre, ktoré zoberú do roviny takmer všetky outliere ale nepokryjú iné nerovinné objekty... toto všetko je tiež potrebné stihnúť v optimálnom čase, ktorý je používateľ ochotný čakať na výsledok.	
3542	Segmentácia bodov v oblastí priesečníkov 3 rovín	Bc. Lukas Hudec	3DRecon\Sprint 10	Vytvorenie algoritmu, ktorý by podľa počtu/hustoty/normál bodov v oblasti priesečníku zistil či je validný, alebo fiktívny.	
3543	Experiment s priesečníkmi 2 rovín a hľadanie ohraničenia priamkou	Bc. Lukas Hudec	3DRecon\Sprint 10	Pokus o využitie poznatkov z predchádzajúcich implementácie algoritmu na vytvorenie priamok z vysegmentovaných úsekov region growingu pre hľadanie ohraničenia stien.	
3473	Presunutie XML Parsera do knižnice DataHandling	Bc. Martin Jurik	3DRecon\Sprint 10	Dôvod je jednoduchý - príliš dlhá kompilácia GUI aj jednotlivých metód - vždy keď sa kompilujú, je potrebné skompilovať aj XML parser... problém by sa odstránil presunutím do osobitnej knižnice na správu súborov - DataHandling a možno	

				ju zapuzdriť do našich tried... alebo inak... Každopádne musí preč.	
3564	Zapisnica č. 5	Bc. Martin Jurik	3DRecon\Sprint 10		
3566	Premiestnenie funkcie odstranenia sumu	Bc. Martin Jurik	3DRecon\Sprint 10	Premiestnenie funkcie odstranenia sumu do libky	
3567	Nahradenie datasetu	Bc. Martin Jurik	3DRecon\Sprint 10	Nahradenie aktivneho datasetu spracovanym datasetom po odstraneni sumu	
3555	Aplikácia RANSAC-u na malé plochy	Bc. Michal Korbel	3DRecon\Sprint 10		
3440	Vytvorenie polygonov (priradenie rohových bodov)	Bc. Michal Korbel	3DRecon\Sprint 10		
3443	Nastavenie konfigurovateľných parametrov pre RANSAC	Bc. Michal Korbel	3DRecon\Sprint 10		
3553	Aplikácia RANSAC-u na malé plochy	Bc. Michal Loffler	3DRecon\Sprint 10		
3544	Prepísanie celej BirkysMethodReconstruction metódy do funkcií	Bc. Robert Birkus	3DRecon\Sprint 10		dev
3545	Dokončiť reimplementáciu nájdenia dominantných smerov (3D histogram)	Bc. Robert Birkus	3DRecon\Sprint 10		dev
3547	Implementácia kostry triedy	Bc. Robert Birkus	3DRecon\Sprint 10		dev
3548	Vytvorenie potrebných premenných a metód danej triedy	Bc. Robert Birkus	3DRecon\Sprint 10		
3551	Rozloženie semienok rovnomerne po celom oblaku bodov	Bc. Robert Birkus	3DRecon\Sprint 10		design; dev
3410	Návrh architektúry pre prepojenie metód	Bc. Robert Karasek	3DRecon\Sprint 10	CommonUtil Ešte ak bude potrebné nechávam task ale v podstate Lukáš má už zaradenú metódu na výpočet histogramu vzdialeností	
3628	Vytvorenie Unit Testu pre CommonUtil	Bc. Robert Karasek	3DRecon\Sprint 10		
3639	Testu pre GR	Bc. Robert Karasek	3DRecon\Sprint 10		

11. šprint

ID	Title	Assigned To	Iteration Path	Description	Tags
3674	Nájdenie bodov	Bc.	3DRecon\Sprint	Nie všetky segmentované	

	ohraničenia pre roviny s neúplným ohraničením	Lukas Hudec	11	plochy/roviny sa pretínajú s ďalšími 2 stenami, na to, aby bolo možné určiť ich ohraničenie z len z týchto priesečníkov. Preto je potrebné získať ich body ohraničenia ďalšími sofistikovanými prístupmi.	
3679	3D "Mean shift" na outliers segmentu	Bc. Martin Jurik	3DRecon\Sprint 11	Pomocou 3D kocky/3D mriežky hľadať oblasti, v ktorých sa nachádzajú oddelené zhluky bodov - outliers, ktoré môžu predstavovať objekt na stene. Spolupracuj s Katkou, ak - tak mi volajte, dohodneme sa cez skype	
3666	Doplnenie tlačidla Export	Bc. Martin Jurik	3DRecon\Sprint 11	Pridanie tlačidla na export rekonštruovaných stien a objektov do DXF pre vizualizáciu v CAD programoch.	
3669	Import SDK	Bc. Michal Korbel	3DRecon\Sprint 11	Import libiek a správne nastavenie Vizualizéra, prípadne zistiť či sa vôbec dá a ak, tak prečo sa nedá použiť s VTK vizualizérom.	
3670	Možnosť využitia "trackmouse" v blenderi	Bc. Michal Korbel	3DRecon\Sprint 11	Zistiť čo to dokáže a či v ňom vieme vizualizovať CAD výstup.	
3672	Verifikovanie úplnosti ohraničenia segmentovanej roviny	Bc. Michal Korbel	3DRecon\Sprint 11	Vzhľadom na nejednoznačnosť hľadania hraničných bodov, je potrebné verifikovať, či je už daný segment úplne ohraničený. komentár a poznámky k splneniu úlohy sú v kóde	
3680	Možnost využitia "trackmouse" v blenderi	Bc. Michal Loffler	3DRecon\Sprint 11	Spolupráca s Mišom, skúste na niečo prísť, viete o čo sa jedná.	
3308	Konverzia PCD -> CAD	Bc. Michal Loffler	3DRecon\Sprint 11		
3552	Priradenie každého bodu k najbližšiemu semienku	Bc. Robert Birkus	3DRecon\Sprint 11		design; dev
3554	Výpočet stredu (priemeru) všetkých zhlukov	Bc. Robert Birkus	3DRecon\Sprint 11		design; dev
3682	Definovanie spojistosti bodov v 3D priestore	Bc. Robert Birkus	3DRecon\Sprint 11		design
3683	Návrh enfoceConnectivity funkcie	Bc. Robert Birkus	3DRecon\Sprint 11		design
3684	Implementácia enforceConnectivity funkcie	Bc. Robert Birkus	3DRecon\Sprint 11		dev
3673	Verifikovanie bodov ohraničenia segmentovanej roviny	Bc. Robert Karasek	3DRecon\Sprint 11	Vzhľadom na nejednoznačnosť hľadania hraničných bodov, je potrebné verifikovať, či je už daný segment úplne ohraničený. komentár a poznámky k splneniu úlohy sú v	

				kóde	

12. šprint

ID	Title	Assigned To	Iteration Path	Description	Tags
3741	Vizualizácia a overenie	Bc. Katarina Janeckova	3DRecon\Sprint 12	Ide o správne priradenie farby pre segmenty outliers... každý segment nech je osobitnou farbou, nech sa dá identifikovať... Robove plane segmenty radšej zruš ofarbenie, nech vidieť len tie tvoje segmenty outlierov	
3759	Nainštalovať a spojazdniť TeamViewer a napojenie na školský PC!	Bc. Katarina Janeckova	3DRecon\Sprint 12	A pronto! nech sa nevyhováraš, že ti tie testy idú pomaly :P	
3762	Výber "Hull" alebo "Priesečníky + kde nič tu nič, tak tam Hull"	Bc. Lukas Hudec	3DRecon\Sprint 12	Ohraničenie segmentov, ktoré nemajú priesečník s 2 ďalšími segmentami, že by sa našiel bod ohraničenia je potrebné nájsť iným spôsobom... 2 spôsoby prichádzajú na um 1. jednoduchší - použiť len Hull so správnym nastavením parametra, prípadne osekať miesta, kde sa nachádza priveľa bodov a menšiť ich počet v prehustených oblastiach 2. zložitejší - hľadať priesečníky RANSAC lines a v ich okolí + spojnici vyhodiť body obálky, kde nie sú, tak tam nechať preriedený hull ako v 1.	
3740	Nájdenie vhodného parametra	Bc. Lukas Hudec	3DRecon\Sprint 12	Zbehnúť pár testov a zistiť najvhodnejšiu hodnotu pre oddelenie outliers od inliers Celé sa to odvíja od správne vypočítanej hodnoty smerodajnej odchýlky - tu je ten problém, že nebol správne nafitovaný RANSAC - túto úlohu má niekto iný, ty zatiaľ môžeš testovať s malými konštatnými hodnotami. Je dosť možné, že keď Mišo alebo ja vyriešime problém so smerodajnou odchýlkou, tak to bude hodnota 2xSD dovtedy skús nejakú malú hodnotu... reimplementovať nemusíš nič, len spúšťať testy s parametrami :) na školskom PC vieš pustiť aj viac testov naraz ;)	
3757	Vzájomná vzdialenosť bodov - rozptyl v rovine	Bc. Lukas Hudec	3DRecon\Sprint 12	Už to malo byť zistené dávno... ale teraz už viem ako na to KDTreeFLANN a testujem random body a vzd k najbližším bodom v ich okolí, následne vypočítam priemer alebo iným spôsobom strednú hodnotu pre merania. Hodnota sa použije pre hľadanie bodov v okolí priesečníkov stien...	
3748	Export verifikovaných stien a objektov do DXF	Bc. Martin Jurik	3DRecon\Sprint 12	To isté ako pri vizualizácií - myslím že Mišo Löffler má stále na sebe task na doplnenie funkcionality Export button - konzultuj s ním.	
3750	Labeling verifikovaných stien	Bc. Michal Korbel	3DRecon\Sprint 12	Označenie verifikovanej steny True/False pre ďalšie použitie v hľadaní ohraničení a pri exporte vytvoriť member metódu do triedy Wall "bool isValidWall()"	

3753	Bounding box pre ohraničenia stien	Bc. Michal Korbel	3DRecon\Sprint 12	Zistiť či sa dá nejakým jednoduchým spôsobom vykresliť spolu s bodmi ohraničenia aj "bounding box" pointcloudu - niečo na ten spôsob keď zblbne vizualizácia a spolu s bodmi sa zobrazjú kvádre okolo jednotlivých pointcloudov toto je len optional - pre lepšie zobrazenie, ak bude dostatočné odlíšenie farbami, toto nebude potrebné.	
3763	Rozptyl bodov od roviny	Bc. Michal Korbel	3DRecon\Sprint 12	Nadviazať na pôvodnú metódu získavania štatistík z celého datasetu. Momentálne musíme zrušiť toto zverstvo, ukázalo sa, že RANSAC si to napasuje hocijako a potom berie aj to čo nechceme + trvá zbytočne dlho... computeRoughness môže ostať, ale je potrebné pridať 2 funkcie 1. computeRoughnessOfSegment - vstup bude pointcloud segmentu alebo dataset a indexy segmentu - výber nechám na teba, nepamätám si s čím vie pracovať RANSAC, vpodsate to môže byť jedno, aj tak sa segmenty extrahujú do objektov 2. to isté ale s tým, že sa mu zadá ako parameter AJ koeficienty roviny, ktorá cez segment prechádza - najlepšie riešenie by bolo, keby si v 1. vypočítal cez RANSAC koeficienty a tejto ich potom len poslal, nech nemusíš 2x robiť ten istý kód RANSAC má 2 parametre distanceThreshold a Probability ... treba otestovať aké budú lepšie výsledky keď: distanceThreshold bude 0.0001 ... viem je to málo, ale neriešiš inliers a ani outliers, len presnosť RANSACU distanceThreshold bude 0.001 a 0.01 - myslím, že najlepšie bude pri 0.0001 - už som to ako tak testoval v škole pre Katkine outliers Probability: 0.5 = 50% (default je 99%, čo je blbosť)	
3756	Rozptyl bodov od roviny	Bc. Michal Loffler	3DRecon\Sprint 12	Nadviazať na pôvodnú metódu získavania štatistík z celého datasetu. Momentálne musíme zrušiť toto zverstvo, ukázalo sa, že RANSAC si to napasuje hocijako a potom berie aj to čo nechceme + trvá zbytočne dlho... computeRoughness môže ostať, ale je potrebné pridať 2 funkcie 1. computeRoughnessOfSegment - vstup bude pointcloud segmentu alebo dataset a indexy segmentu - výber nechám na teba, nepamätám si s čím vie pracovať RANSAC, vpodsate to môže byť jedno, aj tak sa segmenty extrahujú do objektov 2. to isté ale s tým, že sa mu zadá ako parameter AJ koeficienty roviny, ktorá cez segment prechádza - najlepšie riešenie by bolo, keby si v 1. vypočítal cez RANSAC koeficienty a tejto ich potom len poslal, nech nemusíš 2x robiť ten istý kód RANSAC má 2 parametre distanceThreshold a Probability ... treba otestovať aké budú lepšie výsledky keď: distanceThreshold bude 0.0001 ... viem je to	

				málo, ale neriešiš inliers a ani outliers, len presnosť RANSACU distanceThreshold bude 0.001 a 0.01 - myslím, že najlepšie bude pri 0.0001 - už som to ako tak testoval v škole pre Katkine outliers Probability: 0.5 = 50% (default je 99%, čo je blbosť)	
3441	Prepojenie GUI a exportéra	Bc. Michal Loffler	3DRecon\Sprint 12		
3558	Otestovanie exportu rekonštruovaných útvarov	Bc. Michal Loffler	3DRecon\Sprint 12		
3751	Testovanie a farebné odlíšenie validných stien pri testovacej vizualizácií	Bc. Robert Karasek	3DRecon\Sprint 12	Testovacie zobrazenie cez metódu - visualise, pre overenie validnosti metódy - verím, že je, ale takto sa overia aj či sú správne ohraničené oblasti. Každý cloud s koordinátmi - bodmi ohraničenia nech sa vykreslí inou farbou, nech je jednoznačne vidieť, kde sú tie body a či ohraničujú len daný segment OriginalData nech sa vykresľuje bielo, nech nemýlia farby segmentov False dajte červenou	
3752	Log pre výpis verifikovaných stien a bodov	Bc. Robert Karasek	3DRecon\Sprint 12	Keď sa overuje metódou stena, nech sa vypíšu aj súradnice bodov ohraničenia - možno bude celkom zaujímavé si ich potom prejsť a prezrieť rozloženie vzhľadom na segment Nech výpis obsahuje všetko užitočné (počet bodov, počet ktoré sú ohraničené, počet neohraničených...) a jednoznačne verdikt metódy	

13. šprint

ID	Title	Assigned To	Iteration Path	Description	Tags
3319	Hľadanie chýb v scéne	Bc. Katarina Janeckova	3DRecon\Sprint 13	Vizualizácia oddelených segmentov inliers a outliers roviny, analyzuj rozdelenie segmentov podľa tvojej metódy a pokús sa ich vizualizovať rozdielnymi farbami. pre rozdielne farby sa nehraj a použi nejaký random generátor - na stacku ich je plno ;)	
3360	3D "Mean shift" na outliers segmentu	Bc. Katarina Janeckova	3DRecon\Sprint 13	Pomocou 3D kocky/3D mriežky hľadať oblasti, v ktorých sa nachádzajú oddelené zhluky bodov - outliers, ktoré môžu predstavovať objekt na stene. Ak s tým budeš mať problémy, tak sa ozvi	
3821	robime.IT	Bc. Katarina Janeckova	3DRecon\Sprint 13	Napísať a poslať článok do 16.4.2016 ak nie tak prúšvih ;)	
3816	Vylepšenie ohraničenia rovín bez priesečníkov	Bc. Lukas Hudec	3DRecon\Sprint 13	oprav to čo nefunguje a modli sa aby konečne fungovalo...	
3747	Vizualizácia verifikovaných stien a objektov	Bc. Martin Jurik	3DRecon\Sprint 13	Keď sa nám podarí rekonštruovať a labelovať overené steny, tak pri vizualizácií z GUI - "Reconstructed data" radio, tak by	

				sa mali zobraziť už nie body, ale rekonštruované shapes. Podľa všetkého by to nemalo byť veľa roboty... Verifikované steny budú označené bool hodnotou resp. "isValidWall()" member metódou. momentálne je to trochu nachuja riešené tou architektúrou, ale na refactoring nie je čas... to asi potom keď to budem opravovať na DP a OOANS :D	
3823	Plagát - dizajn	Bc. Michal Korbel	3DRecon\Sprint 13	vytvoriť návrh dizajnu pre náš plagát, čo sme sa rozprávali že by tam malo byť nejaký flow a z toho vychádzať.	
3824	Video - scenár, príprava	Bc. Michal Korbel	3DRecon\Sprint 13	nejaké prestrihy postupného vývoja a krokov o čo sa jednalo a ako sme to vyriešili...	
3667	Doplnenie funkcionality tlačidla Export	Bc. Michal Loffler	3DRecon\Sprint 13	Keď Martin vytvorí button, je potrebné po úspešnom prepojení Exportera s GUI doplniť funkcionalitu, ktorá exportuje všetky získané steny/objekty do DXF file.	
3678	Export rekonštruovaných stien	Bc. Michal Loffler	3DRecon\Sprint 13	Síce som priradil teba, keďže si s tým robil a máš o tom najväčší prehľad, ale ak vie Robo o čo sa jedná, tak mu daj vedieť a prípadne to prehoď na neho.	
3742	Detekcia potencionálnych objekov z outliers	Bc. Robert Birkus	3DRecon\Sprint 13	Cieľom je nájsť z ouliers také oblaky bodov, ktoré sú časťou potencionálnych objektov	
3818	Skontrolovať komentáre	Bc. Robert Karasek	3DRecon\Sprint 13		
3822	Plagát - dizajn	Bc. Robert Karasek	3DRecon\Sprint 13	vytvoriť návrh dizajnu pre náš plagát, čo sme sa rozprávali že by tam malo byť nejaký flow a z toho vychádzať.	
3825	Video - scenár a príprava	Bc. Robert Karasek	3DRecon\Sprint 13		

Preberací protokol

k projektu

„Rekonštrukcia 3D scény“

Tím č. 5 (R3D) dodal dokumenty

- Dokumentácia k inžinierskemu dielu
- Dokumentácia k riadeniu
- Elektronické médium (zdrojové súbory riešenia zadania, elektronická forma dokumentácie)

ako výstup z predmetu Tímový projekt v akademickom roku 2015/2016.

V Bratislave, dňa 19.5.2016

Ing. Vanda Benešová, Phd	Bc. Katarína Janečková
vedúca projektu	zástupca tímu č. 5