Univerzita Pavla Jozefa Šafárika v Košiciach

Prírodovedecká fakulta

SEGMENTÁCIA RGBD OBRAZOV

DIPLOMOVÁ PRÁCA

Študijný program:	**Im - Informatika**
Študijný odbor:	**9.2.1. Informatika**
Školiace pracovisko:	**Ústav informatiky**
Vedúci záverečnej práce:	**RNDr. František Galčík, PhD.**
Konzultant:	**Ing. Radoslav Gargalík**

Košice 2014

Bc. Matej Nikorovič

Poďakovanie

Moja vďaka patrí RNDr. Františkovi Galčíkovi, PhD. a Ing. Radoslavovi Gargalíkovi za námet, vedenie, pomoci pri riešení problémov a strávený čas počas konzultácií.

Abstrakt

V tejto práci sa zaoberáme segmentáciou RGBD obrazov. Prvá časť pozostáva z analýzy a porovnania jednotlivých prístupov segmentácií RGBD obrazov. V druhej časti opisujeme implementáciu segmentačnej metódy hladkého obmedzenia, jej vylepšenia a prezentujeme dosiahnuté výsledky.

Abstract

The thesis focuses on segmentation of RGBD images. Analysis and comparison of segmentation methods is presented in the first part of the thesis. In the second part, we propose improvements of smoothness constraint segmentation method, describe implementation details, and present results achieved utilizing the improved method.

Obsah

Úvod

Rok čo rok vznikajú nové a nové technológie medzi ktoré patria aj 3D skenery a zariadenia pre prirodzenú interakciu. Ich úlohou je zosnímať priestor a vrátiť 3D reprezentáciu zosnímaného priestoru. Zariadenia pre prirodzenú interakciu sa snažia o niečo viac, a to rozpoznať objekty v tomto priestore. Samotné rozpoznávanie objektov v obraze je problém, ale ak sa scéna rozdelí na menšie časti, ktoré korelujú s objektami v realite, tak tento problém sa značne zjednoduší. Takto rozdelený obraz na malé časti môže byť použitý aj ako vstup pre ďalšie spracovanie určený pre 3D tlač.

Táto práca sa zaoberá práve delením priestorových obrazov na jednotlivé objekty záujmu. V práci analyzujeme existujúce prístupy segmentačných metód pre RGBD obraz. Implementujeme metódu hladkého obmedzenia, ktorú prispôsobíme na veľké a husté priestorové obrazy získané zo zariadenia pre prirodzenú interakciu. Popíšeme implementáciu tak, aby bežala v čo najkratšom čase a preskúmame možnosti vylepšenia tejto metódy.

Prvá kapitola vysvetľuje základné pojmy, poukazuje na nedostatky RGB obrazu, čo je dôvod, prečo sa zaoberáme RGBD obrazmi a ako k ním môžeme pristupovať. Kapitoly 2 a 3 analyzujú existujúce riešenia. V štvrtej kapitole analyzujeme segmentačnú metódu hladkého obmedzenia, kde dávame dôraz na efektivitu čiastkových problémov. Piata kapitola popisuje jednotlivé prispôsobenia a vylepšenia segmentačnej metódy hladkého obmedzenia. V šiestej kapitole poukazujeme na implementačnú časť práce. Posledná kapitola prezentuje dosiahnuté výsledky na RGBD obrazoch získaných zo zariadení pre prirodzenú interakciu.

Kapitola 1

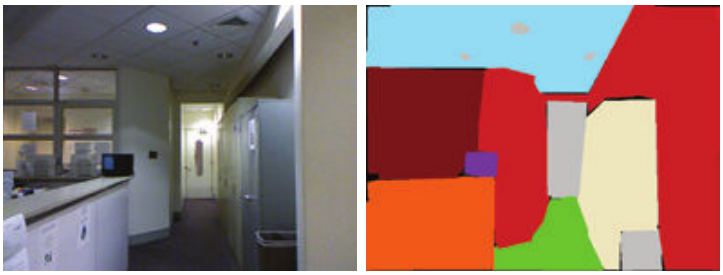
Úvod do problematiky

1.1 Základné pojmy

Segmentácia obrazu je proces delenia obrazu na menšie časti (segmenty), ktoré korelujú s objektmi reálneho sveta (ukážka obr. 1.1). Cieľ segmentácie chápeme subjektívne (napr. objektom môže byť skriňa alebo aj jej časti ako dvierka, kľučka, bočné steny, ...). I keď segmentáciu chápeme subjektívne, definovať ju môžeme presne. Nech I je obraz, ktorý chceme rozdeliť na segmenty $S_1, S_2, \ldots, S_n$, potom platí:

$$\bigcup_{i=1}^{n} S_i = I$$

$$S_i \cap S_j = \emptyset \quad \forall i, j \in \{1, 2, \ldots, n\}, i \neq j$$



Obr. 1.1: Príklad segmentácie.

Jej hlavným cieľom je redukovať objem dát, prípadne zjednodušiť reprezentáciu dát určených pre ďalšie spracovanie.

Mračno bodov (point cloud) je množina priestorových bodov popisujúca tvar objektov.

RGBD obraz je rozšírenie RGB obrazu o hĺbkovú informáciu. Pre každý pixel z RGB obrazu máme k dispozícií informáciu o vzdialenosti tohto bodu od senzora.

Normálový vektor povrchu v bode a je vektor kolmý na dotyčnicu v bode a každej krivky ležiacej na povrchu prechádzajúcej bodom a.

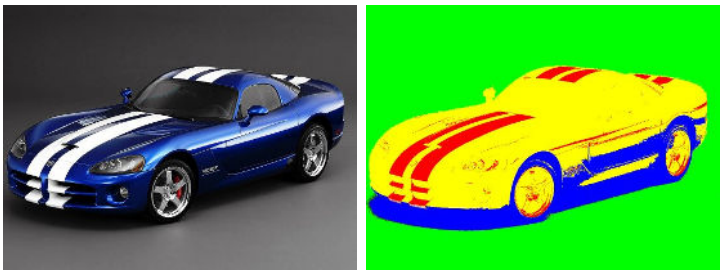
Stred zakrivenia je prienik dvoch nekonečne blízkych normál krivky. Polomerom zakrivenia R budeme rozumieť vzdialenosť medzi bodom krivky a stredom zakrivenia.

Zakrivenie v bode a je $\kappa = \frac{1}{R}$, kde R je polomer zakrivenia v bode a.

1.2 Nedostatky RGB segmentácie

RGB segmentácia je založená buď na danej charakteristike alebo na predpočítanej vlastnosti ako farba, jasová intenzita, textúra. Žiaľ RGB obraz nedrží v sebe priestorovú informáciu, a preto tieto techniky zlyhávajú na týchto problémoch:

- ak jedno teleso má viac rôznych farieb alebo veľké rozdiely jasových intenzít, potom je segmentáciou rozdelené na dva a viac segmentov (obr. 1.2),
- ak je pri sebe viac telies s podobnou farbou, potom sú segmentáciou spojené do jedného segmentu.



Obr. 1.2: Problém RGB segmentácie.

Tieto problémy sú riešiteľné napr. v prípade, že segmentovaný obraz je rozšírený o priestorovú informáciu (RGBD obraz alebo mračno bodov).

1.3 Prehľad existujúcich prístupov

Segmentácia obrazu je relatívne známa oblasť využívaná vedeckými disciplínami ako počítačové videnie alebo spracovávanie digitálneho obrazu. Zatiaľ čo segmentácia

RGB obrazu je relatívne preskúmaná, tak segmentácia RGBD obrazu (resp. mračna bodov) sa rok čo rok vyvíja spolu s vývojom 3D skenerov. Podľa prístupu dokážeme rozdeliť metódy segmentácie RGBD obrazov do troch rôznych skupín:

1. *Metódy založené na minimalizácii energetickej funkcie* – pozerajú na mračno bodov ako na množinu vrcholov grafu a body v lokálnom okolí spájajú hranami. V ňom sa potom formuluje energetická funkcia. Minimalizácia tejto funkcie rieši problémy ako hladkosť obrazu alebo segmentáciu. Viac informácii možno nájsť v kapitole 2.

2. *Clusteringové metódy* – sú založené na triedení bodov do zhlukov bodov. Využívajú data miningové alebo štatistické zhlukovacie (clusteringové) metódy (napr. k-means, k-medians, ...). Viac informácii možno nájsť v kapitole 3.

3. *Segmentácia pomocou podobných vlastností* – patria tu metódy spájania alebo delenia mračien bodov využívajúce podobné vlastnosti bodov (napr. vzdialenosť, farba, normálový vektor povrchu, zakrivenie povrchu, ...). Viac informácii možno nájsť v kapitole 4.

Kapitola 2

Metódy založené na minimalizácii energetickej funkcie

Tieto metódy zjednodušujú problém segmentácie obrazu na oddelenie objektu záujmu od pozadia. Pre objekt záujmu vieme definovať energetickú funkciu E:

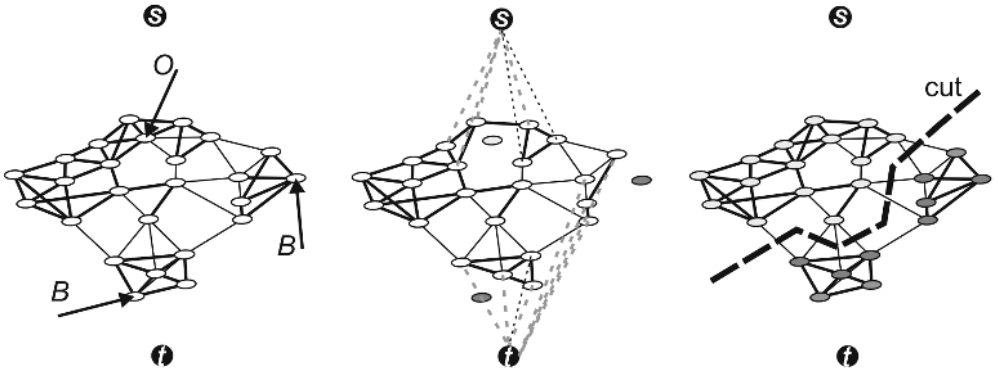
$$E = E_S + E_B. \tag{2.1}$$

Kde E_S je vnútorná energia segmentu, ktorá vyjadruje mieru homogenity segmentu. E_B je energia, ktorá zodpovedá za vlastnosti hranice segmentu.

Vstupom pre tieto metódy je označkovaný obraz. To znamená, že máme čiastočnú informáciu o tom, čo patrí objektu záujmu, čo patrí pozadiu. O zvyšku obrazu rozhodne algoritmus či patrí objektu záujmu alebo pozadiu.

Tento problém je najčastejšie riešený ako problém minimálneho rezu v grafe, ktorý pozostáva z 3 krokov vyobrazených na obr. 2.3.:

1. zostrojenie grafu,
2. zjednodušenie grafu aplikovaním duálneho problému o maximálnom toku, pokým sa graf nerozpadne na 2 segmenty,
3. nájdenie minimálneho rezu.



Obr. 2.3: Priebeh algoritmu minimálneho rezu. Prevzaté z [6].

2.1 Zostrojenie grafu

Máme orientovaný ohodnotený graf G s kapacitou na hranách, ktorého vrcholy sú priestorové body a 2 terminály s, t (s = source, t = target). Terminály s resp. t predstavujú objekt záujmu resp. pozadie. Hrany v tomto grafe delíme na N-linky a T-linky. N-linky sú obojsmerné hrany spájajúce vrcholy, ktoré predstavujú susedné priestorové body. Za susedné priestorové body sa považujú 4 najbližší susedia. Týmto hranám priradíme kapacitu hrany C_E podľa [6] takto:

$$C_E = \kappa * e^{-d_E^2/\sigma} \tag{2.2}$$

kde d_E je euklidovská vzdialenosť koncových vrcholov hrany E, σ je konštanta, ktorá riadi správanie exponenciálnej funkcie, t.j. čím je väčšia, tým vzdialenosť menej ovplyvňuje kapacitu. κ je škálovacia konštanta, ktorá mení rozsah hodnôt kapacity na $\langle 0, \kappa \rangle$.

T-linky sú hrany, ktoré spájajú terminály s vrcholmi, ktoré predstavujú priestorové body. Podľa označkovaného obrazu spojíme terminál s smerom k vrcholom patriacich objektu záujmu a vrcholy patriace pozadiu spojíme smerom k terminálu t. T-linkám priradíme kapacitu nekonečno.

Na takto zostrojený graf sa aplikuje metóda maximálneho toku. Práce [7] a [6] riešia tento problém pomocou Boykovho algoritmu popísaného v práci [8].

Kapitola 3

Zhlukovacie techniky

Zhlukovacie techniky sú založené na triedení bodov do zhlukov bodov. Využívajú data miningové alebo štatistické zhlukovacie (clusteringové) metódy (napr. k-means, k-medians, ...). Definícia týchto metód je veľmi blízka definícii segmentácie.

3.1 K–Means

K–Means je heuristický algoritmus vektorovej klasifikácie. Rieši problém rozdelenia n-dimenzionálnych vektorov do k homogénnych zhlukov reprezentovaných ich stredom (centroidom). I keď tento problém je NP–ťažký, K–Means rýchlo skonverguje k lokálnemu minimu. Vstupom algoritmu je k, ktoré predstavuje výsledný počet zhlukov, a $v_1, v_2, \ldots, v_m$ sú vstupné n-dimenzionálne vektory. Algoritmus pozostáva z niekoľkých krokov:

1. *Inicializácia* sa stará o úvodné rozmiestnenie centroidov $c_1, c_2, \ldots, c_k$. Tento krok popíšeme v ďalšej sekcii.

2. *Pridelenie vektorov k centroidom.* Každému vstupnému vektoru v_i nájdeme (euklidovský) najbližší centroid c_j a pridelíme tento vektor v_i k centroidu c_j.

3. *Nové vektory centroidov* získame aritmetickým priemerom vektorov, ktoré sme k ním pridelili v predchádzajúcom kroku. Ak sa nové vektory centroidov nelíšia od starých vektorov tak končíme, inak pokračujeme krokom 2.

Na výstupe nájdeme k centroidov, ku ktorým sú pridelené zhluky bodov, čiže segmenty.

3.1.1 Inicializačný krok K–Meansu

Ako sme už spomínali, tento krok slúži na úvodné rozmiestnenie centroidov. Štandardne sa používa *výber náhodných vektorov*, t.j. každý centroid inicializujeme náhodne rovnomerne jedným zo vstupných vektorov.

Autori algoritmu K–Means++ [5] popisujú metódu, ktorou je možné znížiť počet iterácií K–Meansu práve pomocou vhodného výberu centroidov. Ich hlavnou myšlienkou je vyberať centroidy čo najďalej od seba. Ich postup vyzerá nasledovne:

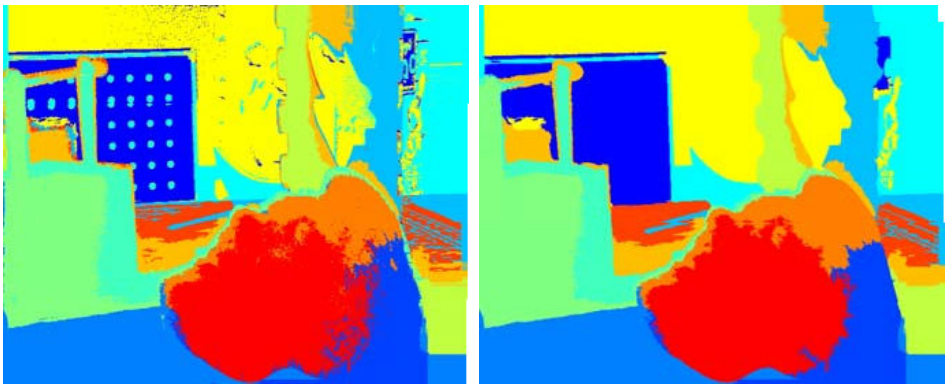
1. Prvý centroid c_1 inicializujeme náhodne rovnomerne zo vstupných vektorov.
2. Ďalší centroid c_i vyberieme $c_i = v_j$ zo vstupných vektorov s pravdepodobnosťou $\frac{D(v_j)}{\sum_{l=0}^{m} D(v_l)}$, kde $D(x)$ je vzdialenosť vektora x od najbližšieho centroida.
3. Krok 2 opakujeme, pokým inicializujeme každý centroid.

3.2 Segmentácia využívajúca zhlukovacie techniky

Práca [4] sa zaoberá segmentačným prístupom, ktorý využíva uniformný farebný priestor a K–Means algoritmus na nájdenie segmentov obrazu. Uniformný farebný priestor je taký, v ktorom zmenou hodnoty farby vykonáme podobnú zmenu ľudského vizuálneho vnemu. Medzi uniformné farebné priestory patria: CIELab, Hunter–Lab.

Pri tomto prístupe je vstupom algoritmu K-means 6-rozmerný vektor $[w_p * x, w_p * y, w_p * z, w_c * L, w_c * a, w_c * b]$, kde $[x, y, z]$ je súradnica priestorového bodu, $[L, a, b]$ je farba priestorového bodu v CIELab farebnom priestore a w_p, w_c sú priestorové a farebné váhy používané ako korekcia spojenia dvoch modelov. Algoritmus sa spúšťa s pomerne vysokým číslom k, aby sa predišlo problému, že algoritmus nájde menej segmentov, ako je počet vizuálnych segmentov na obraze. Na tento odsegmentovaný obraz sa aplikuje post-processing, ktorý pospája príbuzné segmenty do jedného. Vstup a výsledok algoritmu je vyobrazený na obrázkoch 3.4 a 3.5.

Obr. 3.4: Vstup K-Meansu - vľavo RGB obraz a vpravo hĺbková mapa. Prevzaté z [4].



Obr. 3.5: Výstup K–Meansu a výsledok post-processingu. Prevzaté z [4].

Kapitola 4

Segmentácia pomocou podobných vlastností

K metódam segmentácie pomocou podobných vlastností patria tu metódy spájania alebo delenia mračien bodov využívajúce podobné geometrické vlastnosti. Jednou z metód je segmentácia využívajúca hladké obmedzenie.

Segmentácia využívajúca hladké obmedzenie (podľa [1]) je segmentačný prístup založený na princípe hladkosti povrchov, ktoré obklopujú objekty záujmu. Hlavný princíp tejto metódy je nasledovný: Uvažujme 2 body, pričom jeden z nich leží v okolí toho druhého. Pre obidva body poznáme normálový vektor a zakrivenie povrchu, na ktorom ležia. Ak rozdiel zakrivenia plôch v daných bodoch je malý a súčasne uhol medzi normálovými vektormi v daných bodoch je malý, potom tieto body patria jednému objektu. Vznik metódy hladkého obmedzenia bol motivovaný viacerými problémami, ktoré sa vyskytovali pri známych segmentačných metódach:

- Segmentačné metódy mali veľký počet parametrov (napr. počet iterácií, veľkosť najmenšej chyby, veľkosť najmenšieho segmentu, maximálny počet segmentov, veľkosť lokálneho okolia, prah homogenity, maximálna okrúhlosť, ...), ktorých význam a efekt na výsledok nie je vždy jasný. Väčšina z nich mala ešte samostatnú metódu na odhad týchto parametrov.
- Väčšina metód mala problém so zakriveným povrchom.
- Modelovo-založené segmentačné metódy, ktoré rozpoznávali povrch objektov, napríklad pomocou Houghovej transformácie, sa nedali použiť na mračná bodov.

Algoritmus sa skladá z niekoľkých krokov, kde na vstupe máme mračno bodov (podľa práce [1]):

1. Ku každému bodu z mračna bodov nájdeme jeho najbližších susedov.
2. Pomocou najbližších susedov odhadneme normálový vektor a zakrivenie povrchu pre každý bod mračna.
3. Priradenie bodov objektom.
4. Vrátime zoznam objektov s ich zoznamom bodov.

Zložitejšie kroky algoritmu postupne rozoberieme v ďalších sekciách.

4.1 Hľadanie najbližších susedov

K hľadaniu najbližších susedov môžme pristupovať dvoma spôsobmi: fixovať počet najbližších susedov lokálneho okolia alebo fixovať polomer lokálneho okolia.

k najbližších susedov (kNN)

Tento prístup vyberá práve k najbližších susedov z okolia bodu (pokým nie je v mračne bodov menej bodov ako k). Veľkosť okolia, z ktorého sa body vyberajú závisí od konštanty k. Výhoda tohto prístupu je zložitosť, lebo vždy vyberie konštantný počet bodov. Bežne sa používa tento prístup na riedke mračná, lebo ľubovoľne si nastavuje veľkosť okolia. Veľká nevýhoda tohto prístupu je pri zlom výbere k. Vtedy pre nejaký bod nájde príliš vzdialené body, ktoré môžu pokaziť ďalšie výpočty.

Najbližší susedia v guľovom okolí (FDNN)

Tento prístup nemení veľkosť okolia, z ktorého sa vyberajú susedia. Ale nemáme garantovaný počet bodov, ktorý sa nachádza v tomto okolí. Keďže tento prístup garantuje veľkosť okolia, tak vo všeobecnosti dáva lepšie výsledky ako kNN. Používa sa na hustejšie mračná, ale pri zlom výbere polomeru okolia rastie rýchlo čas potrebný na nájdenie susedov.

Ak vezmeme do úvahy, že mračná bodov získané 3D skenermi sú husté a presnosť by mala byť na prvom mieste, potom vhodnejší prístup bude FDNN, ktorý ďalej rozanalyzujeme v 4 rôznych spôsoboch riešenia (podľa [2]):

1. brute force,
2. projekcia,

3. zhlukovanie,

4. k-d strom.

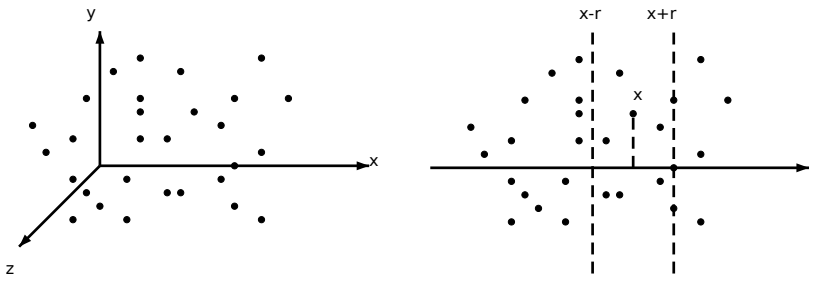
4.1.1 Brute force

Najjednoduchší prístup hľadania najbližších susedov. Na vstupe máme polomer gule, mračno bodov a bod, pre ktorý hľadáme suseda. Pre každý bod z mračna určíme vzdialenosť od hľadaného bodu. Ak vzdialenosť je menšia ako polomer gule, tak si ho uložíme do pomocnej štruktúry.

Zložitosť: ak n je počet bodov v mračne bodov, potom algoritmus má zložitosť $O(n)$. Tento krok nám treba vykonať pre každý bod z mračna bodov, t.j. celkovo $O(n^2)$.

4.1.2 Projekcia

Tento prístup orezáva vstup o body, ktoré sú určite ďalej ako polomer gule. Bez ujmy na všeobecnosti usporiadame mračno bodov podľa prvej dimenzie. K bodu, ku ktorému hľadáme okolie, umelo vytvoríme 2 body s rovnakými súradnicami okrem prvej dimenzie. V jednom odčítame a v druhom pričítame polomer v prvej dimenzii. S pomocou binárneho vyhľadávania nájdeme pozície týchto dvoch bodov (viď obr. 4.6). Ak sa nenachádzajú v mračne bodov, tak to nevadí, lebo binárne vyhľadávanie nám vráti pozíciu, kde by sa mal nachádzať hľadaný bod. Tým sme získali rozsah bodov, ktoré podľa prvej dimenzie patria do gule. Pre tieto body ešte overíme, či ich vzdialenosť je menšia ako polomer gule.

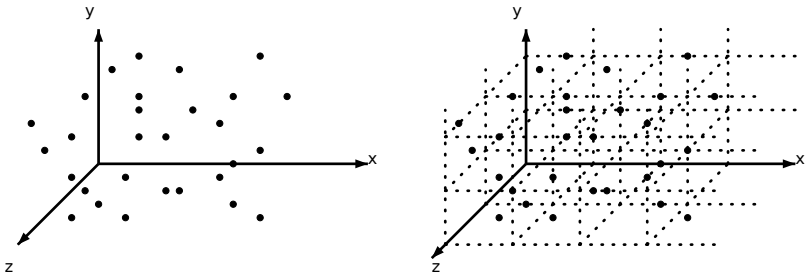


Obr. 4.6: Využitie projekcie na redukciu bodov pri hľadaní najbližších susedov.

4.1.3 Zhlukovanie

Prístup zhlukovania sa snaží vyriešiť problém projekcie. A to je testovanie bodov s veľkou vzdialenosťou získanou ostatnými dimenziami. Celý priestor rozdelíme na

zhluky bodov na d_i rovnomerných častí v i-tej dimenzii (viď obr. 4.7). Vieme zistiť, do ktorého zhluku patrí bod, pre ktorého hľadáme susedov. Podľa tejto pozície preiterujeme okolité zhluky, v ktorých sa môže nachádzať sused vzdialený menej ako polomer gule.



Obr. 4.7: Využitie zhlukovania na redukciu bodov pri hľadaní najbližších susedov.

K-d strom

K-d strom (k-dimensional tree) je viacrozmerný binárny vyhľadávací strom. V každom uzle si ukladá k-rozmerný bod. Vo všeobecnosti vnútorné uzly tohto stromu delia nadrovinu na 2 časti. Body vľavo od deliaceho uzla patria ľavému podstromu deliaceho uzla a rovnako body vpravo pravému podstromu. Uzly v jednej generácii stromu rozdeľujú nadrovinu v rovnakom rozmere. Každou generáciou sa použitý rozmer strieda.

Ak je tento strom vyvážený, tak nájsť najbližšieho suseda trvá $O(\log n)$. Zároveň ak hľadáme ďalšieho najbližšieho suseda, tak je treba aktuálne nájdeného odstrániť zo stromu, čo trvá $O(\log n)$. Toto opakujeme pokým nájdeme bod, ktorý má väčšiu vzdialenosť ako polomer gule.

Lenže ak chceme s týmto stromom manipulovať, tak ho potrebujeme vybudovať. Ak má byť strom vyvážený, tak koreň stromu je medián zoznamu bodov. Na to je hneď niekoľko spôsobov:

- $O(n \log^2 n)$ v každej úrovni stromu usporiadame zoznam bodov, ktoré sme ešte nepridali do stromu,
- $O(n \log n)$ v každej úrovni hľadáme medián v lineárnom čase.

Zložitosť: Ak k je maximálny počet bodov v guľovom okolí, tak zložitosť algoritmu je $O(k \log n)$ + čas na vybudovanie stromu. Pre každý bod mračna bodov má zložitosť $O(kn \log n)$ + čas na budovanie stromu.

4.2 Metódy odhadu normálových vektorov a zakrivenia povrchu

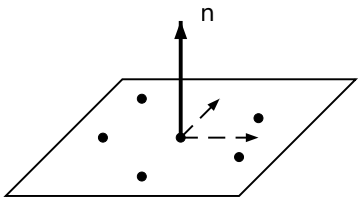
Z poznania okolia každého bodu sme schopní odhadnúť, akým smerom smeruje normálový vektor povrchu v danom bode a ako je v tomto bode povrch zakrivený. Existuje niekoľko prístupov, ktoré riešia túto problematiku (podľa [3]):

- PlaneMLR,
- PlanePCA,
- VectorMLR,
- VectorMean,
- SphereMLR.

4.2.1 PlaneMLR

PlaneMLR je priamočiara metóda počítania normálového vektora povrchu. Tento prístup sa snaží nájsť rovinu, ktorá má minimálnu vzdialenosť od každého bodu z okolia (obr. 4.8). Pomocou metódy najmenších štvorcov vieme formálne zapísať chybovú funkciu (4.1), ktorej minimalizáciou získame rovinu s najmenšou chybou ku každému bodu z okolia. Hľadaný normálový vektor v danom bode je normálový vektor nájdenej roviny.

$$f(\alpha) = \sum_{i=1}^{n} \left|\alpha X_i\right|^2 \tag{4.1}$$



Obr. 4.8: PlaneMLR.

Metóda najmenších štvorcov

Táto metóda sa používa na nájdenie parametrov matematického modelu. Jej cieľ je minimalizovať kvadrát rozdielu medzi hľadaným modelom a vstupnými dátami. Použitie tejto metódy na vyriešenie problému PlaneMLR:

Majme ľubovoľnú rovinu $Ax+By+Cz+E=0$ a vstupné dáta $a_1, a_2, \ldots, a_n \in R^3$, kde $x_i = a_i[0]$, $y_i = a_i[1]$, $z_i = a_i[2]$.

Bez ujmy na všeobecnosti upravíme rovnicu roviny na $Ax + By + Cz + D = K$, kde K je nenulová konštanta. Z nej si vyjadríme funkciu horizontálnej vzdialenosti od roviny $f(A, B, C, D) = Ax + By + Cz + D - K$. Dosadením ľubovoľného vstupu a_i do modelu získame jeho horizontálnu vzdialenosť. Potom podľa definície tejto metódy hľadaná funkcia, ktorú minimalizujeme, vyzerá takto:

$$f(A, B, C, D) = \sum_{i=1}^{n} \big[Ax_i + By_i + Cz_i + D - K\big]^2$$

Minimum nájdeme pomocou parciálnych derivácií podľa každej premennej, ktoré sa rovnajú 0:

$$\frac{\partial f}{\partial A} = \sum_{i=1}^{n} 2\big(Ax_i^2 + Bx_iy_i + Cx_iz_i + Dx_i - x_iK\big) = 0$$

$$\frac{\partial f}{\partial B} = \sum_{i=1}^{n} 2\big(Ax_iy_i + By_i^2 + Cy_iz_i + Dy_i - y_iK\big) = 0$$

$$\frac{\partial f}{\partial C} = \sum_{i=1}^{n} 2\big(Ax_iz_i + By_iz_i + Cz_i^2 + Dz_i - z_iK\big) = 0$$

$$\frac{\partial f}{\partial D} = \sum_{i=1}^{n} 2\big(Ax_i + By_i + Cz_i + D - K\big) = 0$$

Hore uvedené vzťahy prepíšeme do maticového tvaru.

$$\begin{bmatrix} \sum x_i^2 & \sum x_iy_i & \sum x_iz_i & \sum x_i \\ \sum x_iy_i & \sum y_i^2 & \sum y_iz_i & \sum y_i \\ \sum x_iz_i & \sum y_iz_i & \sum z_i^2 & \sum z_i \\ \sum x_i & \sum y_i & \sum z_i & n \end{bmatrix} \times \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} = \begin{bmatrix} \sum x_iK \\ \sum y_iK \\ \sum z_iK \\ \sum K \end{bmatrix}$$

Riešením tejto matice (sústavy lineárnych rovníc) nájdeme model, t.j. všeobecnú rovnicu roviny.

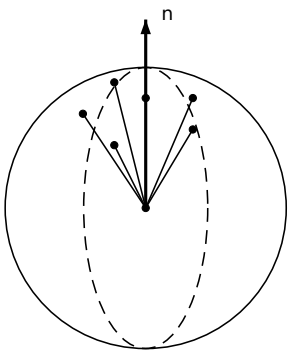
Poznámka: Ak by platilo $K = 0$, tak sústava rovníc je homogénna. Každá homogénna sústava má aspoň 1 riešenie (nulové riešenie). Ak determinant matice homogénnej sústavy je nenulový, tak sústava má práve 1 riešenie. Až teraz vidno, prečo je nenulová konštanta K dôležitá. V implementácii sme použili $K = 1$.

4.2.2 SphereMLR

Tento prístup sa snaží minimalizovať vzdialenosť gule od každého bodu okolia (obr. 4.9). Formálne chybovú funkciu zapíšeme vzťahom (4.2). Normálový vektor potom

bude vektor smerujúci zo stredu gule k danému bodu, pre ktorý vektor hľadáme.

$$f\big(G(M,r)\big) = \sum_{i=0}^{n} \big|G(M,r)X_i\big|^2 \tag{4.2}$$



Obr. 4.9: SphereMLR.

Linearizácia problému a použitie metódy najmenších štvorcov

Všeobecná rovnica gule v priestore, ak $M = [m, n, o]$ je stred gule a r je polomer (M a r sú neznáme):

$$(x-m)^2 + (y-n)^2 + (z-o)^2 = r^2$$

Po roznásobení a usporiadaní:

$$x^2 + y^2 + z^2 - 2xm - 2yn - 2zo + m^2 + n^2 + o^2 - r^2 = 0$$

Pomocou substitúcie $A = 2m$, $B = 2n$, $C = 2o$, $D = -m^2 - n^2 - o^2 + r^2$ je možné tento vzťah zlinearizovať na:

$$x^2 + y^2 + z^2 - Ax - By - Cz - D = 0$$

Kvôli zjednodušeniu zápisu zavedieme substitúciu $K_i = x_i^2 + y_i^2 + z_i^2$:

$$Ax + By + Cz + D - K_i = 0$$

Túto rovnosť použijeme pri tvorbe chybovej funkcie:

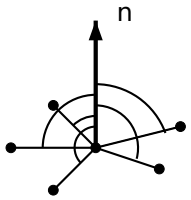
$$f(A,B,C,D) = \sum_{i=1}^{n} \big[Ax_i + By_i + Cz_i + D - K_i\big]^2$$

A riešime to rovnako ako PlaneMLR. Riešením nájdeme parametre A, B, C, D a z nich vypočítame hľadané paramatre gule takto: $M = \big[\frac{A}{2}, \frac{B}{2}, \frac{C}{2}\big]$,
$r = \sqrt{D + \big(\frac{A}{2}\big)^2 + \big(\frac{B}{2}\big)^2 + \big(\frac{C}{2}\big)^2}$.

4.2.3 PlanePCA

Namiesto minimalizovania chybovej funkcie, táto metóda minimalizuje disperziu (t.j. rozptýlenosť hodnôt okolo strednej hodnoty). Podľa [3] je tento problém ekvivalentný problému maximalizácie uhlov medzi normálovým vektorom a vektormi od daného bodu k bodom jeho okolia (obr. 4.10). Formálne chybovú funkciu môžeme zapísať takto:

$$f(\overrightarrow{n}) = \sum_{i=1}^{n} \left| \angle \overrightarrow{n} \overrightarrow{CX_i} \right|^2 \tag{4.3}$$



Obr. 4.10: PlanePCA.

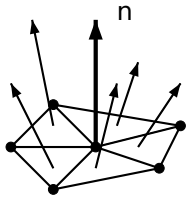
4.2.4 VectorMLR

Na konštrukciu roviny sú potrebné 3 body. Na tomto princípe je založená metóda VectorMLR. K danému bodu si vezmeme dvoch susedov z okolia. Tým vieme získať pomocný normálový vektor. Tento postup opakujeme, pokým nepoužijeme všetky body z okolia daného bodu (obr. 4.11). Hľadaný normálový vektor je ten, ktorý má minimálny uhol ku každému pomocnému vektoru. Formálne zapísaná chybová funkcia vzťahom (4.4).

$$f(\overrightarrow{n}) = \sum_{i=1}^{n} \left| \angle \overrightarrow{n} \overrightarrow{n_i} \right|^2 \tag{4.4}$$

Jednoduchšia interpretácia tohto problému môže byť: súčet/priemer pomocných normálových vektorov je hľadaný normálový vektor. Pre odlíšenie ho nazveme VectorMean. Nevýhoda tohto prístupu je náchylnosť na šum.

$$\overrightarrow{n} = \frac{1}{n-1} \sum_{i=1}^{n-1} \overrightarrow{n_i} \quad , \overrightarrow{n_i} \perp \overrightarrow{CX_iX_{i+1}} \tag{4.5}$$



Obr. 4.11: VectorMLR.

4.3 Priradenie bodov objektom

Posledný krok segmentačných algoritmov založených na podobných vlastnostiach je samotné pridelenie bodov k jednotlivým objektom. Je možné použiť jeden z dvoch prístupov:

- *Prístup zhora nadol* – na začiatku pridelíme všetky body jednému segmentu. Ak je tento segment nehomogénny, tak ho rozdelíme na viac segmentov. Toto pravidlo použijeme na každý vzniknutý segment.
- *Prístup zdola nahor* (tzv. region-growing) – začíname s nejakými semienkami, ktoré predstavujú jednotlivé segmenty. Pomocou kritéria homogenity nechávame tieto segmenty narastať.
- *Kombinovaný prístup* – na začiatku pridelíme všetky body jednému segmentu. Ak je tento segment nehomogénny, tak ho rozdelíme na viac segmentov. Ak spojením vzniknutých segmentov vznikne homogénny segment, tak ich spojíme. Toto pravidlo použijeme na každý vzniknutý segment.

Väčšina metód využíva práve prístup zdola nahor.

4.3.1 Priradenie bodov objektom v metóde hladkého obmedzenia

Podľa práce [1] autori metódy využili algoritmus prehľadávania komponentov grafu (pomocou prehľadávania do šírky) ako prístup zdola nahor. Body mračna prehlásili za vrcholy grafu. Kritérium homogenity: 2 vrcholy sú spojené hranou, ak bod zodpovedajúci jednému vrcholu grafu je v okolí bodu, ktorý zodpovedá druhému vrcholu grafu a naviac medzi nimi platí hladké obmedzenie (t.j. dané body majú podobné lokálne vlastnosti - normálový vektor a zakrivenie povrchu). Semienka, v ktorých spúšťame prehľadávanie do šírky, vyberáme v poradí od bodu s najmenším zakrivením povrchu.

4.4 Vyhladenie šumu segmentácie

Výsledkom segmentácie je zoznam objektov a ich zoznam bodov. Lenže zoznam objektov bude uchovávať aj objekty, ktoré pozostávajú z malého počtu bodov, ktoré môžeme prehlásiť za šum. Existujú 2 riešenia vyhladenia šumu:

1. Spojiť tento malý objekt so susedným, ktorý ma viac bodov - môže dôjsť k istej nehomogenite objektov (hladké obmedzenie ich nepriradilo k sebe).

2. Odstrániť celý tento malý objekt zo zoznamu.

Metóda, ktorá rieši oddelenie pre nás zaujímavých objektov od šumu, sa nazýva prahovanie. Pomocou zvoleného prahu (v našom prípade minimálny počet bodov objektu) určí porovnaním hodnôt, či odsegmentovaný objekt je alebo nie je šum.

Kapitola 5

Návrh vylepšení metódy hladkého obmedzenia

Riešenie bude pozostávať z vylepšenia prístupu segmentácie využívajúcej hladké obmedzenie. Jeho slabinou je interpolácia normálových vektorov v ostrých hranách obrazu. Prístup rozdelí obraz na malé plochy podobné rovinám alebo pospája objekty oddelené ostrými hranami.

Zameriame sa na prípad, keď viac objektov je pospájaných. To znamená, že segmentácia má príliš voľné parametre. Mračno bodov získané z 3D skenera je pomerne husté a nachádza sa v ňom podložka, na ktorej stoja naše objekty záujmu (viď obr. 5.12). V princípe podložka je rovina, ktorá po odstránení z obrazu oddelí objekty záujmu, lebo objekty nebudú hladko prepojené. Pomocou hladkého obmedzenia získame jednotlivé objekty záujmu.

Obr. 5.12: Snímané telesá, ktoré sú objektami záujmu, ležia na podložke. Zdroj: internet

Postup riešenia segmentácie a ich samostatný cieľ:

1. Redukcia mračna bodov - slúži na zrýchlenie výpočtov a potlačenie šumu.
2. Hľadanie najbližších susedov.

3. Odhad normálových vektorov a zakrivenia povrchu - získanie deskriptorov pre každý bod.

4. Hľadanie rovín v obraze - potrebné na detekciu podložky.

5. Odstránenie podložky - slúži na oddelenie objektov záujmu. Zautomatizovaná verzia odstráni najväčšiu rovinu v obraze.

6. Hľadanie segmentov v obraze.

7. Vyhladenie šumu segmentácie.

5.1 Redukcia dát

Efektívne riešenie bude tzv. digitalizácia (podobné riešenie ako pri zhlukovacej technike hľadania najbližších susedov). Určíme veľkosť najmenšieho dielu (alebo na koľko dielov sa rozdelí priestor v 1 rozmere). Rozdelíme každý rozmer priestoru podľa tohto dielu. Ak padne viac bodov do rovnakého dielu, tak urobíme ich ťažisko (=priemer). Novovzniknutý bod si bude pamätať, z ktorých bodov vznikol, aby vo výsledku figurovali pôvodné dáta. Takto vzniknuté mračno bodov bude značne redukované, rovnomerne rozmiestnené a odolnejšie voči šumu.

Odolnosť voči šumu

Predpokladajme, že viac bodov z okolia popisuje len jeden skutočný bod p. Vychýlenie bodov vzniklo aditívnym šumom, t.j. $p_i = p + n_i$, kde n_i je vektor šumu. Môžeme predpokladať, že náhodné veličiny, ktoré popisujú aditívny šum v jednotlivých bodoch sú nezávislé. Šum je popísaný nulovou strednou hodnotou a odchýlkou σ v každom smere. Priemerom získame:

$$p' = \frac{\sum_{i=0}^{n} p}{n} + \frac{\sum_{i=0}^{n} n_i}{n} \tag{5.1}$$

Priemerom rovnakých bodov (prvý sčítanec) získame ten istý bod, ale priemerom šumu (druhý sčítanec) získame náhodnú veličinu s nulovou strednou hodnotou a odchýlkou rovnou $\frac{\sigma}{\sqrt{n}}$.

Z toho vyplýva, že redukované mračno bodov vznikne z bodov s menším aditívnym šumom, čo spresní detekciu rovín.

5.2 Hľadanie najbližších susedov

Kvôli presnejším výsledkom sme sa rozhodli použiť metódu FDNN čiže hľadanie najbližších susedov v guľovom okolí. Otázka zostáva, ako si vybrať vhodný polomer gule.

5.2.1 Automatický výber polomeru gule

V práci [1] sa píše o tom, že mračná bodov majú rôznu hustotu bodov a prístup FDNN kazil výsledky. Navyše experimenty ukázali, že ak sa vyberá 50 bodov z okolia, tak je to vhodný počet bodov na dobrý odhad normál a zakrivenia povrchu.

Prvý fakt pre nás neplatí, keďže pracujeme s hustými a po redukcii s rovnomernými mračnami bodov. Druhý fakt môžeme použiť v náš prospech tak, že vieme nájsť vzdialenosť 50. najbližšieho suseda pre každý bod. Toto riešenie je len inak napísané hľadanie kNN. Vyberieme náhodne rovnomerne niekoľko bodov z mračna bodov a pre nich získame vzdialenosť 50. najbližšieho suseda a uložíme si ich do zoznamu. Z týchto vzdialeností vyberieme medián z druhej polovice zoznamu, aby bola podmienka zachovaná (50 bodov v okolí). Pri výbere maxima pravdepodobne vyberieme hraničný prípad obrazu a lokálne okolie bude obsahovať priveľa bodov a naviac každým spustením algoritmu by dával odlišné výsledky.

Algoritmus

Na vstupe máme mračno bodov M a počet bodov n, pre ktoré chceme túto vzdialenosť hľadať.

List $zoznamPolomerov$;
for *int* $i \leftarrow 1$ *to* n **do**
    List $l \leftarrow \emptyset$;
    Point $p \in M$;
    foreach *Point* $q \in M$ **do**
        $l \leftarrow vzdialenost(p, q)$;
    end
    double $dist = vratKnajmensiPrvok(l, 50)$;
    $zoznamPolomerov \leftarrow dist$;
end
return $vratKnajmensiPrvok(zoznamPolomerov, n * 3/4)$;

Týmto algoritmom získame polomer gule na hľadanie najbližších susedov. Optimálne riešenie je, ak výber k-najmenšieho prvku bude bežať v čase $O(n)$ (pomocou algoritmu *QuickSelect* alebo $MediansOfMedians$, ktorých popis a analýzu nájdete v [9] v kapitole 9: Medians and Order Statistics) a zároveň každá iterácia algoritmu môže bežať v samostatnom vlákne.

5.3 Odhad normálových vektorov a zakrivenia povrchu

Tento problém riešime štandardnou technikou PlaneMLR, popísanou v predchádzajúcej kapitole. Ako vstupné dáta použijeme najbližších susedov z výstupu predchádzajúceho kroku.

Zakrivenie povrchu sme nahradili priemernou kvadratickou odchýlkou bodov z okolia bodu od nájdenej roviny. Čím je odchýlka menšia, tým lokálne okolie viac popisuje rovinu. Naopak, čím je táto odchýlka väčšia, tým lokálne okolie je viac oblejšie až hranatejšie. Túto odchýlku nazveme chybou okolia.

5.4 Priradenie bodov objektom

Priradenie bodov objektom sa skladá z dvoch fáz:

1. *Hľadanie rovín v obraze* – pomocou pozície bodov a ich normálových vektorov necháme narastať oblasť (začíname s bodmi, ktoré majú minimálne zakrivenie povrchu). Roviny, ktoré prehlási používateľ za podložku, odstránime z mračna bodov. Zautomatizovaná verzia bude predpokladať, že podložka je najväčšia rovina v obraze.

2. *Hľadanie segmentov v obraze* – pomocou normálových vektorov a chyby necháme narastať oblasť, ktorú tvorí jednotlivý objekt.

5.4.1 Hľadanie rovín v obraze

Body s najmenšou chybou okolia najviac pripomínajú rovinu. Z toho usporiadame body mračna podľa chyby vzostupne. Vytvoríme graf, kde vrcholy grafu sú jednotlivé priestorové body a hrany sú medzi 2 vrcholmi, ak k nim prislúchajúce body sú v lokálnom okolí.

Algoritmus narastania vyzerá nasledovne:

1. každý vrchol grafu prehlásime za nenavštívený
2. vyberieme taký nenavštívený vrchol grafu, ktorý má najmenšiu chybu
3. zostrojíme rovinu, ktorá prechádza bodom, ktorý prislúcha vybranému vrcholu a jej normálový vektor je odhadnutý z okolia tohto bodu
4. na vybranom vrchole spustíme prehľadávanie do šírky (BFS) s novou značkou, ktoré navštívi iba tých susedov, ktorí majú podobný normálový vektor a ich pozícia bodu je blízko od zostrojenej roviny
5. pokračujeme krokom 2

Poznámka: podobný normálový vektor znamená $|\angle\alpha_U\alpha_V| \leq k$, pozícia je blízko od roviny znamená $|\rho V| \leq l$, pozorovaním vychádza $l = \frac{\text{polomer gule okolia}}{2}$.

Výsledok je segmentácia mračna bodov do jednotlivých rovín. Kvôli lepšiemu prehľadu môžeme použiť vyhladenie šumu segmentácie. V tomto kroku používateľ označí, ktoré roviny sú podložka a odstránime ich z mračna bodov.

5.4.2 Hľadanie segmentov v obraze

Táto fáza je analogická s predchádzajúcou. Jediný rozdiel je v tom, že BFS navštevuje susedov, ktorí spĺňajú hladké obmedzenie (majú podobné normálové vektory a podobnú chybu).

5.5 Vyhladenie šumu segmentácie

Ako bolo spomínané v predchádzajúcej kapitole, použijeme prístup prahovania na oddelenie objektov od pozadia. Čiže dôležitá otázka je voľba prahu. Prah sme zvolili ako priemerný počet bodov vo výsledných segmentoch,
t.j. $prah = \frac{\text{počet bodov v mračne}}{\text{počet segmentov}}$. Tento prah sa môže v aplikácií interaktívne meniť.

5.5.1 Vyhladenie dier v jednotlivých segmentoch

Predchádzajúce prahovanie odstráni šum z obrazu, ale za šum môže byť prehlásený bod objektu záujmu, ktorý tesne nevyhovuje hladkému obmedzeniu. Taktiež objekt záujmu môže hladké obmedzenie rozdeliť na viac segmentov.

Zoberieme do úvahy prvý problém. V odstránenom segmente, čiže aj v odstránenom bode, je v lokálnom okolí veľký výskyt iného segmentu. To nám dáva algoritmus na vyplnenie dier:

1. nájdeme všetky body, ktoré boli odstránené vyhladením šumu
2. pre každý bod nájdeme najviac vyskytujúci sa iný segment v lokálnom okolí a pripojíme bod do tohto segmentu

Ak bol odstránený väčší segment, tak iba jeho okraje sa pripoja k vedľajšiemu segmentu. Ak spustíme predchádzajúci algoritmus ešte raz, nové okraje odstráneného segmentu sa pripoja k vedľajšiemu segmentu. Algoritmus, ktorý rieši tento problém, je len viacnásobné opakovanie predchádzajúceho algoritmu dovtedy, pokým nenastane zmena segmentov. To nám dáva riešenie na druhý problém, ak objekt záujmu sa rozdelí na viac segmentov. Stačí zvýšiť prah vyhladenia šumu na takú hodnotu, aby z objektu záujmu ostal iba najväčší segment.

Kapitola 6

Implementácia

Na implementáciu sme zvolili moderný objektovo-orientovaný jazyk C#. Mono je open-source cross-platform implementácia jazyka C# kompatibilná s Microsoft .NET Framework-om. Implementácia sa skladá z niekoľkých častí:

- pomocné knižnice = *OpenTK*, *Mapack*
- spoločné jadro projektov = *Core*
- implementácia segmentačných metód = *Clustering*, *Smoothness Constraint*

6.1 OpenTK

OpenTK = Open Toolkit library je nízkoúrovňová .NET open-source knižnica, ktorá slúži ako .NET wrapper pre knižnice OpenGL (graphics library), OpenCL (computing library) a OpenAL (audio library). Knižnica je voľne stiahnuteľná z `http://www.opentk.com/`.

6.2 Mapack

Mapack je .NET open-source knižnica určená pre základné vypočty lineárnej algebry. Podporuje základné maticové operácie a vlastnosti ako sčítanie, odčítanie, násobenie, determinant, inverziu matíc, riešenie sústav lineárnych rovníc a mnoho ďalších.

Knižnica je voľne stiahnuteľná z `https://github.com/lutzroeder/Mapack/`.

6.3 Core

Knižnica *Core* je spoločné jadro pre naše segmentačné metódy, ktoré obsahuje:

- základné dátové štruktúry mračien bodov *Point3*, *Vector3* a *Plane3*
- triedy pre prácu s týmito štuktúrami *Point3OffFactory*, *Point3Transformation*, *Point3XYZComparer*
- utility pre parsovanie (*Scanner*), meranie času behu operácií (*SystemTimer*) a rýchlu manipuláciu s poľami(*ArrayOperations*)

ArrayOperations je trieda obsahujúca metódy pre rýchlu manipuláciu s poľami. Obsahuje implementáciu nájdenia k-teho najmenšieho prvku v poli pomocou randomizovaného Quick Selectu (jeho zložitosť je O(n)).

Point3 je dátová štruktúra uchovávajúca pozíciu trojrozmerného bodu a jeho príslušnej farby vo farebnom modeli RGB. Okrem getterov a setterov obsahuje metódu na určenie vzdialenosti od ďalšieho priestorového bodu.

Plane3 je dátová štruktúra reprezentujúca všeobecnú rovnicu roviny. Obsahuje normálový vektor roviny *Vector3* a priestorový bod *Point3*, ktorým rovina prechádza. Jej úlohou je počítať vzdialenosť ľubovoľného priestorového bodu od roviny.

Point3OffFactory je trieda určená pre vstupno–výstupné operácie s mračnami bodov formátu OFF. Parsovanie vstupných dát uskutočňuje trieda *Scanner* a následne ich ukladá do dátovej štruktúry *Point3*.

Point3XYZComparer porovnávač dvoch priestorových bodov *Point3* pre usporiadanie podľa ich súradníc (usporiadanie podľa prvej súradnice; ak majú súradnicu rovnakú, tak porovnávanie pokračuje v ďalšej súradnici). Porovnávač je samostatný z dôvodu, že priestorový bod môžeme usporiadať viacerými spôsobmi. Porovnávač je potrebný pre operácie usporiadania a binárneho vyhľadávania v mračne bodov.

Scanner je vlastná implementácia reťazcového parsera založená na čítaní reťazca po znakoch. Nepoužíva regulárne výrazy a tým je efektívnejšia ako robustné metódy implementované v jazyku.

SystemTimer je vlastná implementácia stopiek podobných ako *StopWatch* založených na systémovom čase. Slúži na meranie času vykonávaných operácií.

Vector3 je dátová štruktúra uchovávajúca trojrozmerný vektor a naviac obsahuje niektoré operácie s ním.

6.4 Clustering

Clustering je implementácia segmentačnej metódy využívajúca data–miningovú techniku K–Means. Clustering obsahuje dátové štruktúry potrebné pre K–Means (*Color3f*, *VectorKD*), implementáciu samotného algoritmu a metódy transformácií farebných priestorov.

Color3f je dátová štruktúra obsahujúca 3 farebné zložky v inom dátovom type ako štruktúra *Point3*.

ColorSpaces je trieda určená na transformáciu farebných priestorov. Implementuje transformáciu z RGB na uniformný priestor CIE Lxy a transformáciu z CIE Lxy na CIE Lab.

GLForm sa stará o vizualizáciu. Pomocou knižnice *OpenTK* trieda volá metódy knižnice *OpenGL* a zobrazuje mračno bodov v okne.

KMeans je všeobecná implementácia algoritmu K–Means využívajúca štruktúru *VectorKD*. Inicializácia algoritmu je založená na náhodnom výbere počiatočných klastrov.

VectorKD je dátová štruktúra k–dimenzionálneho vektora pre všeobecnú implementáciu K–Means. Naviac v sebe ukladá kvôli optimalizácii číslo klastra, ku ktorému je priradený.

Main je inicializačná trieda, ktorá obsahuje *main* a vytvára objekt *GLForm*.

6.5 SmoothnessConstraint

SmoothnessConstraint je implementácia segmentačnej metódy využívajúca hladké obmedzenie s naším rozšírením.

ClusteringStruct štruktúra reprezentuje priestorové zhluky bodov. Obsahuje metódu, ktorá priraďuje priestorovému bodu jemu prislúchajúci zhluk bodov. Táto trieda sa používa pri hľadaní najbližšieho suseda pomocou zhlukovania a pri redukcii objemu dát.

FixedDistanceNearestNeighbours je trieda starajúca sa o hľadanie najbližších susedov ku každému bodu mračna bodov. Má implementované metódy hľadania pomocou hrubej sily, projekcie a zhlukovania. S dnešným trendom až štandardom mať viacjadrové procesory sme implementovali aj paralelizáciu týchto algoritmov na úrovni vlákien.

GLWindow je určená na vizualizáciu celej segmentácie. Pomocou knižnice *OpenTK* trieda volá metódy knižnice *OpenGL* a zobrazuje mračno bodov v okne. V ľubovoľnom čase vieme zobrazovať aktuálne mračno bodov, jeho normálové vektory, jeho jednotlivé segmenty a naviac ich vieme odstrániť resp. uložiť do súboru.

GLWindowDesgin je parciálna trieda *GLWindow*, v ktorej je implementovaný vzhľad okna.

NormalApproximation je interface pre odhadovanie normálových vektorov. Tento interface implementujú triedy *PlaneMLR* a *SphereMLR*.

PlaneMLR je implementácia metódy PlaneMLR. Trieda sa stará o zostavenie sústavy lineárnych rovníc pre výpočet všeobecnej rovnice roviny okolia daného bodu. Túto sústavu rovníc rieši pomocou knižnice *Mapack*.

PointStruct je rozšírená dátová štruktúra odvodená od *Point3*. Priestorový bod má okrem pozície a farby aj ďalšie rozšírené vlastnosti ako zoznam svojích susedov, normálový vektor plochy (v ktorej leží), zakrivenie plochy, priradenie bodu k objektu a validita (či bod nie je šum). Tieto rozšírené vlastnosti sa postupne dopĺňajú počas behu segmentácie.

PointStructFactory obsahuje metódu na posunutie a škálovanie celého mračna bodov na určité súradnice. Využitie má pri vizualizácií pomocou *OpenGL* (súradnice objektu sa pohybujú v intervale $\langle -1, 1 \rangle$).

SmoothnessConstraintCore pradstavuje jadro segmentačnej techniky. Spája funkcionalitu pomocných tried a obsahuje segmentačnú techniku *Smoothness Constraint* a aj našu úpravu tohto algoritmu.

SphereMLR je implementácia metódy SphereMLR. Trieda sa stará o zostavenie sústavy lineárnych rovníc pre výpočet gule popisujúcej okolie daného bodu. Túto sústavu rovníc rieši pomocou knižnice *Mapack*.

Main je inicializačná trieda, ktorá obsahuje *main* a vytvára objekt *GLWindow*.

6.6 Binárny formát

Kvôli zrýchleniu načítavania a testovania mračien bodov sme ich konvertovali do binárneho formátu, ktorý vyzerá nasledovne:

- 4 bajty - počet bodov v mračne
- 1 bajt - obsah (0 - máme iba zoznam bodov, 1 - farba je prítomná)
- zoznam bodov: súradnica bodu typu float (x, y, z), farba bodu typu byte (ak je prítomná) (r, g, b)

Kapitola 7

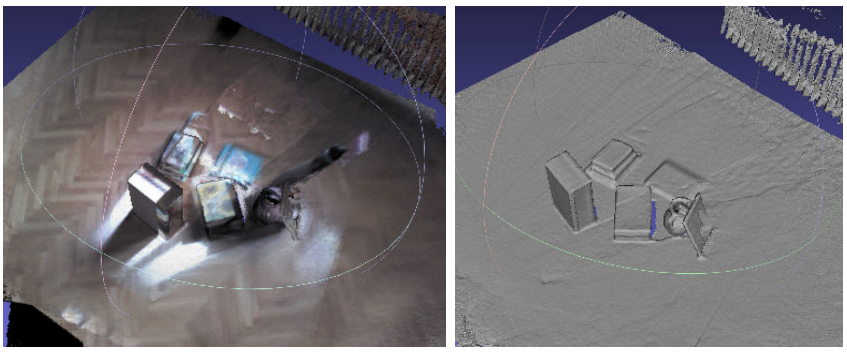
Výstupy segmentácie a ich analýza

Navrhnutú aplikáciu sme testovali na mračnách bodov získaných z prístroja pre prirodzenú interakciu – Kinect. Samotná scéna bola snímaná z viacerých uhlov pohľadu a spájaná dokopy pomocou technológie Kinect Fusion.

Vytvorených bolo 8 rôznych mračien bodov rekonštruovaných bez farby a s farbou získavanou z každej 5., 10. a 15. snímky. Jedno mračno bodov obsahuje zhruba 8 až 12 miliónov bodov.

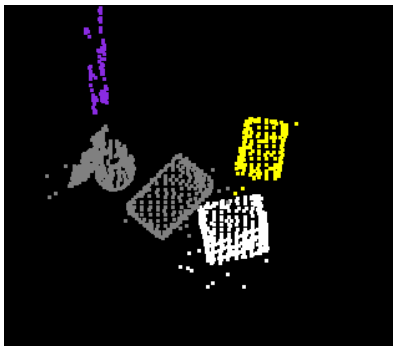
Vyhodnotenie výsledku prebiehalo iba vizuálnou formou, keďže označkovať každý bod v 10 miliónovom mračne bodov by trvalo netriviálne veľa času.

Prvé mračno bodov je nasnímaná izba, kde na podlahu sme uložili krabicu, jednu knihu, viac kníh na seba, džbán a pod. Mračno bodov je zobrazené na obr. 7.13.



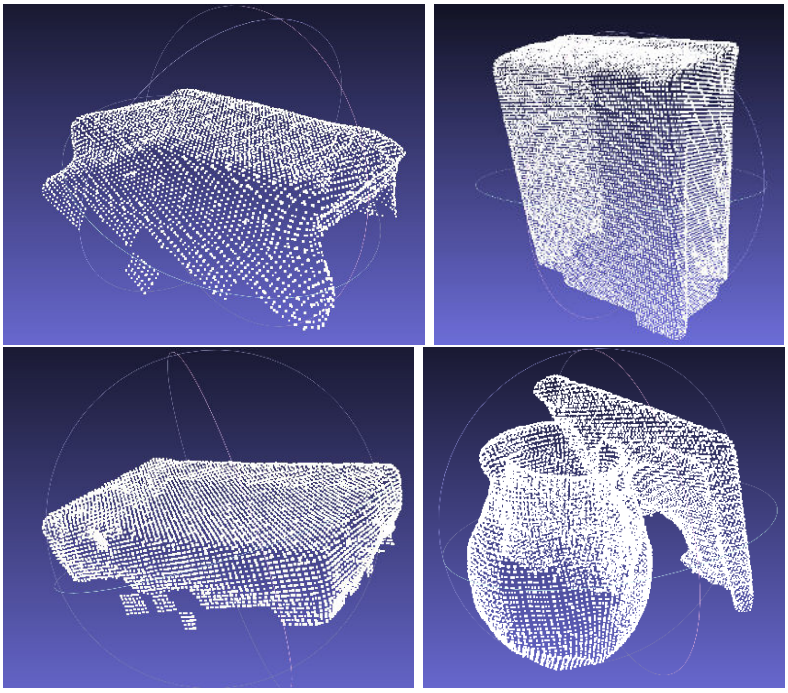
Obr. 7.13: Prvé mračno bodov - vľavo farebná verzia, vpravo verzia bez farby.

Po spustení zautomatizovanej segmentácie dostaneme prvotný výsledok, kde je odstránená podložka a získame úvodné segmenty, ktoré čiastočne korelujú s realitou. Na obr. 7.14 vidno, že knihy sú spojené s džbánom. Taktiež tam vidno, že kniha, ktorá je príliš tenká, bola odstránená spolu s podložkou.



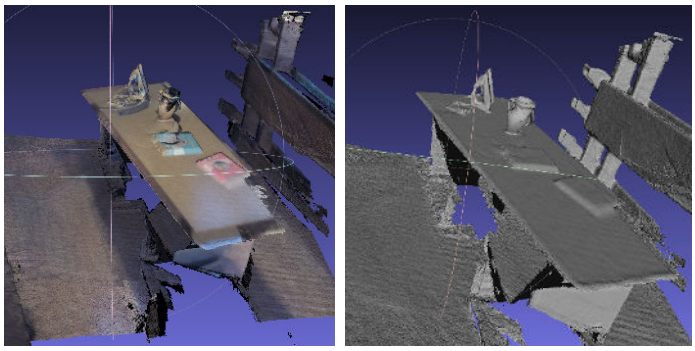
Obr. 7.14: Prvotný výsledok automatickej segmentácie.

Po vybraní jednotlivých objektov záujmu zautomatizovaná segmentácia (bez detekcie a odstraňovania rovín) odstráni okolitý šum od segmentov a oddelí blízke objekty, ktoré boli spojené v predchádzajúcom kroku. Výsledok vidíme na obr. 7.15.



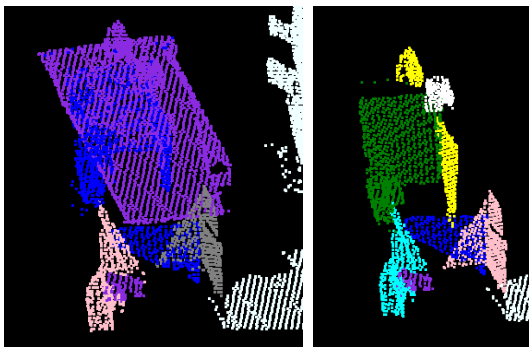
Obr. 7.15: Jednotlivé objekty odsegmentované od seba v poradí podľa pôvodného mračna bodov.

Druhé mračno bodov zobrazuje izbu, v ktorej máme telesá uložené na stole viď obr. 7.16.



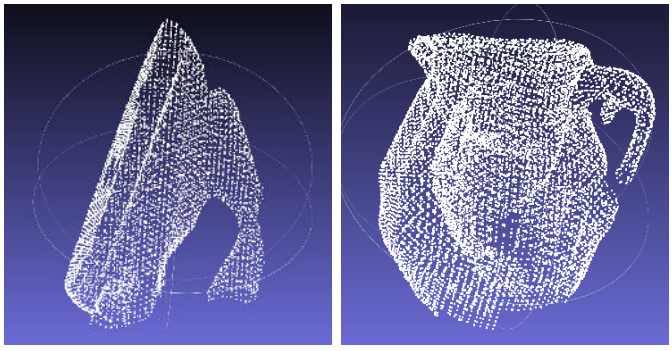
Obr. 7.16: Druhé mračno bodov - vľavo farebná verzia, vpravo verzia bez farby.

Zautomatizovaná segmentácia v tomto mieste zlyhala, keďže podložka telies je stôl, čo nie je najväčšia rovina v obraze. To znamená, že metóda spojila stôl s objektami záujmu. Ak použijeme manuálnu verziu prístupu, tak môžeme odstrániť nie len podlahu ale aj samotný povrch stola, čo vidno na obr. 7.17.



Obr. 7.17: Prvotný výsledok automatickej segmentácie vľavo a manuálnej vpravo.

Po vybraní jednotlivých objektov záujmu dostávame samostatné mračná bodov žehličky a džbánu vyobrazené na obr. 7.18.



Obr. 7.18: Jednotlivé objekty odsegmentované od seba v poradí podľa pôvodného mračna bodov.

7.1 Efektívnosť segmentačnej metódy

Navrhnutá aplikácia dáva vizuálne dobré výsledky, ale neprezentovali sme výsledky rýchlosti spracovávania. Aplikácia bola testovaná notebookom Lenovo U310 s parametrami:

- Hardvér – Procesor Intel i3 3217U 1,8GHz, 4GB RAM DDR3 1600MHz, ...
- Operačný systém – Manjaro Linux 0.8.9 64-bit (založený na Arch Linux-e).
- .NET platforma – Mono verzia 3.2.3-2.

Predtým ako spustíme segmentáciu na mračne bodov, tak potrebujeme ho načítať. Keďže vstupné mračná bodov sú veľké a husté, tak sme sa zaoberali aj rýchlosťou načítavania. Rýchlosti načítavania sme merali na 4 rôznych mračnách bodov. Priemerné časy otvárania súborov v milisekundách nájdeme v tejto tabuľke:

Časy otvárania v ms	17148 bodov	199907 bodov	439832 bodov	1220439 bodov
Parsovanie	96	1208	2680	7619
Skenovanie	167	2010	4488	12292
Binárne čítanie	14	211	470	1419

Z tabuľky vyplýva, že binárne čítanie je najrýchlejší spôsob načítavania súborov. Skenovanie je založené na postupnom čítaní číslic po sebe, ktoré bolo efektívnejsie ako parsovanie, ale parsovanie súborov bolo nejskôr optimalizované novšou verziou projektu Mono a začalo dávať lepšie výsledky.

Ak už máme načítane mračno bodov, tak je potrebné nájsť najbližších susedov a odhadnúť normálové vektory a zakrivenia povrchu. V nasledujúcej tabuľke prezentujeme priemerné časy metód hľadania najbližších susedov, metódu odhadu normálových vektorov PlaneMLR a ich paralelné verzie. Časy boli merané na 22836 bodovom mračne s fixne nastaveným polomerom guľového okolia.

Časy výpočtov v ms	1 vlákno	4 vlákna
Brute force	9062	4058
Projekcia	470	270
Zhlukovanie	251	136
PlaneMLR	439	221

V tabuľke vidieť, že tieto metódy sú dobre paralelizovateľné.

Grafická aplikácia má naviac konzolový výstup, ktorý vyhodnocuje čas výpočtov jednotlivých krokov. Nasledujúci príklad je automatická segmentácia mračna bodov *KV_01.bin* z priečinka *no_color*:

Loaded in 10390.36 ms

Reducing point cloud... 8136639 -> 25340; 3684.515 ms

Detecting nearest neighbours radius... 0.0632707001640646; 64.401 ms

Finding nearest neighbours... 417.228 ms

Approximating normals... 538.379 ms

Detecting planes 134.61 ms

Deleting max segment... 5.516 ms

Segmenting... 51.254 ms

Noise smoothing... 88.65 ms

done in 1300.038 ms

Kapitola 8

Záver

V práci analyzujeme možné prístupy segmentácie RGBD obrazov. Neskôr sa zameriavame na segmentačnú metódu pomocou hladkého obmedzenia a implementujeme ju čo najefektívnejšie a aplikujeme tento prístup na husté mračná bodov získané z 3D skenerov.

Navrhnutá aplikácia je schopná nájsť a oddeliť od seba nasnímané objekty uložené na podložke. Jej dôraz bol kladený na husté, nerovnomerne rozložené mračná bodov a na efektivitu výpočtov. Predstavené výsledky práce splnili očakávania z hľadiska kvality výstupu navrhnutej segmentačnej metódy ako aj jej časovej zložitosti s ohľadom na možnosti efektívnej paralelizácie.

Vzhľadom na to, že aplikácia bola navrhnutá pre rýchle spracovanie, výsledné okraje objektov pri podložke nemusia byť plynulé. Ich neplynulosť vzniká pri redukcii dát a následnom odstraňovaní podložky z obrazu. Pre dosiahnutie vyššej presnosti okrajov by bola potrebná analýza pôvodného obrazu, čo by viedlo k výraznému spomaleniu výpočtov. Naviac redukcia dát pomohla pri presnejšej detekcii rovín.

Tak ako pôvodná metóda hladkého obmedzenia, tak aj naše vylepšenie pracuje iba s geometriou obrazu a nerieši jeho farebnú informáciu. Táto informácia môže byť použitá pri vyhladzovaní dier v segmentoch alebo pri vyhladzovaní ich okrajov.

Zoznam použitej literatúry

[1] RABBANI T., HEUVEL F. A., VOSSELMAN G.: Segmentation of point cloud using smoothness constraint. ISPRS Commission V Symposium 'Image Engineering and Vision Metrology', 2006.

[2] BENTLEY J. L.: A survey of techniques for fixed radius near neighbor searching. Stanford University, August 1975.

[3] KLASING K., ALTHOFF D., WOLLHERR D., BUSS M.: Comparison of Surface Normal Estimation Methods for Range Sensing Applications. Technische Universitat Munchen, 2009.

[4] DAL MUTTO C., ZANUTTIGH P., CORTELAZZO G. M.: Scene Segmentation by Color and Depth Information and its Applications. University of Padova, 2010.

[5] ARTHUR D., VASSILVITSKII S.: K–Means++: The Advantages of Careful Seeding. Society for Industrial and Applied Mathematics Philadelphia, 2007.

[6] SEDLACEK D., ZARA J.: Graph Cut Based Point-Cloud Segmentation for Polygonal Reconstruction. Czech Technical University in Prague, 2009.

[7] FUNKHOUSER T., A. GOLOVINSKIY: Min-Cut Based Segmentation of Point Clouds. Princeton University, 2009.

[8] BOYKOV Y., KOLMOGOROV V.: An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision. Computer Science Department at the University of Western Ontario, 2004.

[9] CORMEN T. H., LEISERSON C. E., RIVEST R. L., STEIN C.: Introduction to Algorithms Third Edition. The MIT Press, 2009.

Príloha A

Obsah DVD

- zdrojový kód aplikácie - Visual Studio 2010 Solution (./workspace/)
- skomprimované testovacie mračná bodov:
  - mračná bodov bez farby (./pc/no_color/)
  - mračná bodov s farbou (./pc/color_5/; ./pc/color_10/; ./pc/color_15/)
- PDF súbor tejto práce (./dip.pdf)