SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

DOKUMENTÁCIA K ŠTVRTÉMU ZADANIU

POČÍTAČOVÉ VIDENIE A SPRACOVANIE OBRAZU

Študijný program:	Robotika a kybernetika
Študijný odbor:	kybernetika
Školiace pracovisko:	Ústav robotiky a kybernetiky

Bratislava 2023 **Bc. Daniel Kiš, Bc. Martin Kochan**

Zadanie:

Cieľom zadania, je oboznámiť sa s pracou s mračnom bodov (point cloud) a segmentácia objektov v priestore. Študent si vyskúša vytvorenie vlastného mračna bodov a aplikáciu metód na získanie segmentovaného priestoru. Použitie externých knižníc ako open3d, sklearn, opencv, a iných je dovolené a odporúčané. Zadanie pozostáva z viacerých úloh:

1. Vytvorenie mračna bodov pomocou Kinect v2 pre testovanie. Nájdite online na webe mračno bodov popisujúce väčší priestor (väčší objem dát aspoň 4x4 metre) pre testovanie algoritmov a načítajte mračno dostupného datasetu (2B)
2. Pomocou knižnice (open3d - python) načítate vytvorené mračno bodov a zobrazíte. (2B)
3. Mračná bodov očistite od okrajových bodov. Pre tuto úlohu je vhodne použiť algoritmus RANSAC. (5B)
4. Segmentujete priestor do klastrov pomocou vhodne zvolených algoritmov (K-means, DBSCAN, BIRCH, Gausian mixture, mean shift ...). Treba si zvoliť aspoň 2 algoritmy a porovnať ich výsledky. (5+5B)
5. Detailne vysvetlite fungovanie zvolených algoritmov. (4B) (Keďže neimplementujete konkrétny algoritmus ale používate funkcie tretích strán je potrebné rozumieť aj ako sú funkcie implementované)
6. Vytvorte dokumentáciu zadania (popis implementovaných algoritmov, Grafické porovnanie výstupov, vysvetlite rozdiel v kvalite výstupov pre rozdielne typy algortimov) (2B).

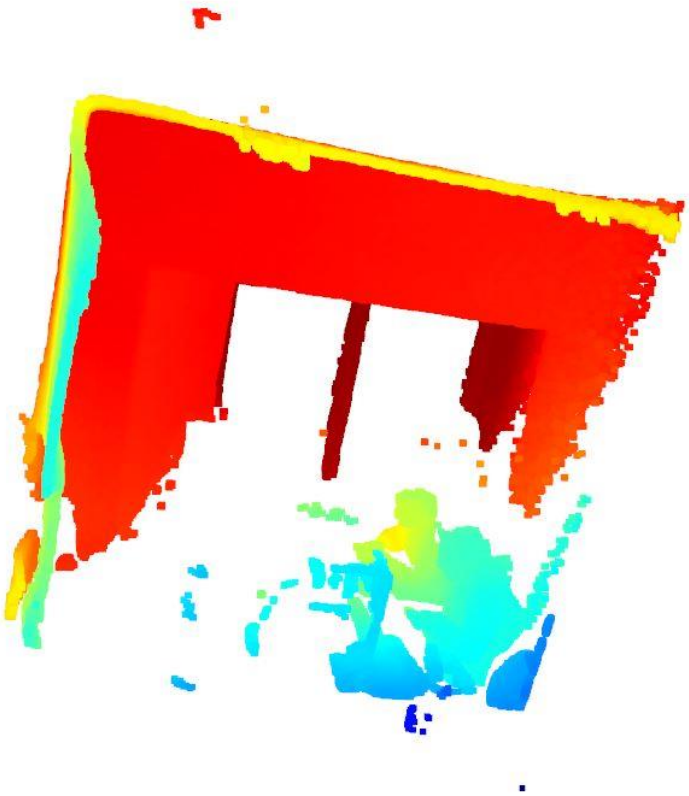
Obsah

1 Predspracovanie

Cieľom zadanie je sa zoznámiť s prácou a segmentáciou point cloudov v priestore. Pre tento účel bol vytvorený point cloud pomocou Kinect V2, ktorý bude použitý na overenie rôznych typov algoritmov na prácu s point cloudamy. Pre segmentáciu a čistenie bodov sa tiež zvolil aj externý point cloud, ktorý bol stiahnutý s online stránky.

1.1 Použité mračna bodov

Pre prácu s Kinect-om bol použitý rovnaký program ako z 3. zadania. Pomocou tohto programu bol vytvorený point cloud, ktorý je znázornený na obr. 1.



Obr. 1: Mračno bodov z Kinectu

Z horeuvedeného mračna bodov vidno, že existujú skupiny bodov, ktoré sú oddelené od iných skupín. Na takéto body bude vhodne použiť algoritmus RANSAC, ktorý niektoré s týchto bodov eliminuje. Dôvod, prečo v mračne bodov sa nepodarilo uložiť hodnoty farieb, je že vznikov problém pri zápise mračna bodov do *.pcd* súboru, ktoré *o3d* nie je schopný spracovať.

Na obr. 2 je znázornené mračno bodov, ktoré bolo stiahnuté z weba.

Obr. 2: Mračno bodov z weba

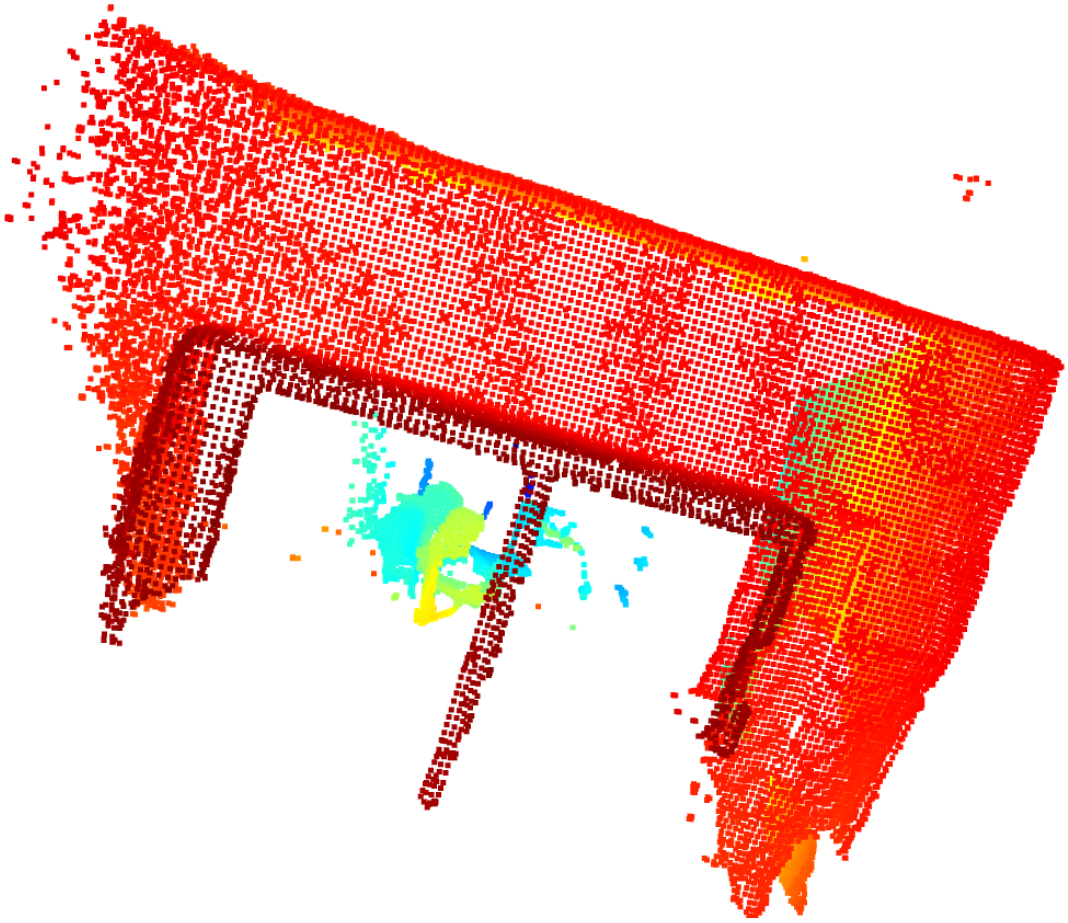
Horeuvedené mračno bodov bolo zvolené preto, že obsahuje množstvo bodov, ktoré sú oddelené od seba, čo bude slúžiť veľmi dobre na overenie funkčnosti algoritmov pre segmentáciu a čistenie okrajových bodov.

1.2 Očistenie od okrajových bodov

Pri zbieraní dát a vytváraní mračien bodov sa stáva kvôli rôznim chybám merania, zašumení alebo rušivého prostredia. Že, vznikajú vo vytvorených mračnách bodov body ktoré nepatria do nami snímaného modelu alebo sú inač rušivé taktiež známe ako „ noise alebo artifacts“ , je nevhodné ich mat pri ďalšom pracovaní s polom. Pre toto existuje viac metód, to čo sme použili je funkcia „voxel_down_sample“ pre zníženie počtu bodov a pre odstránenie okrajových rušivých bodov sme použili funkciu „remove_radius_outlier“.

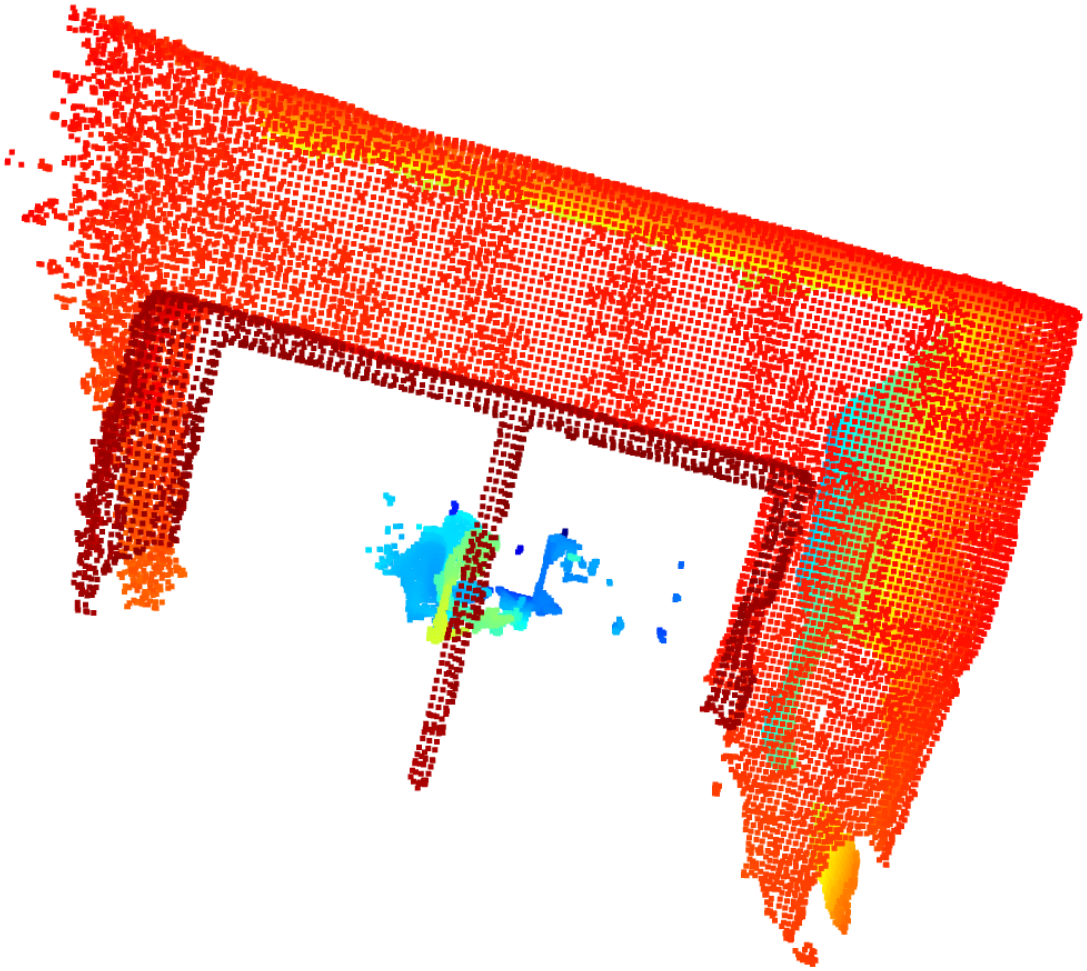
Funkcia „voxel_down_sample“ Znižuje počet bodov vo vstupnom mračne bodov pomocou voxelu ako voxel_size vstupní argument ktorého hodnota býva obyčajne v stotinách čísiel nami požité bolo 0,03 viď. Obr.3 ale čím väčšie tím viac bodov odfiltruje. Samotný voxel je častica objemu, predstavujúca hodnotu v pravidelnej mriežke trojdimenzionálneho priestoru počítačovej grafiky. Je to vlastne analógia

k pixelu, ktorý reprezentuje 2D grafiku. Voxely sa používajú najčastejšie pri vizualizácií a analýze lekárskych a vedeckých dát.



Obr. 3: Mračno bodov z Kinectu zo znížením počtu bodov použitím voxel_down_sample

Pre očistenie takto zredukovaného mračna bodov sme použili funkciu „remove_radius_outlier“. Táto funkcia berie ako input počet bodov a polomer sféry okolo kontrolovaného bodu. Odstraňovanie bodov realizuje tak že prechádza cez všetky body v mračne bodov a okolo každého kontrolovaného bodu vytvorí sféru s zadaným polomerom a kontroluje či nami zadaný počet bodov sa nachádza vo vnútri ak nie vymaže kontrolovaný bod ak áno tak ho ponechá. Viď. Obr. 4



Obr. 4: Mračno bodov z Kinectu po odstránení okrajových bodov.

2 Segmentácia

Segmentácia do klastrov je proces rozdeľovania obrovských údajov na menšie časti. Existujú mnohé algoritmy pre klastrovanie, z ktorých sú niektoré vhodnejšie na použitie keď je dataset nízkej kvality, alebo keď dataset obsahuje ohromné množstvo dát. Na účel klastrovania point cloudov, ktoré sú znázornené v častiach vyššie boli použité algoritmy K-means a BIRCH.

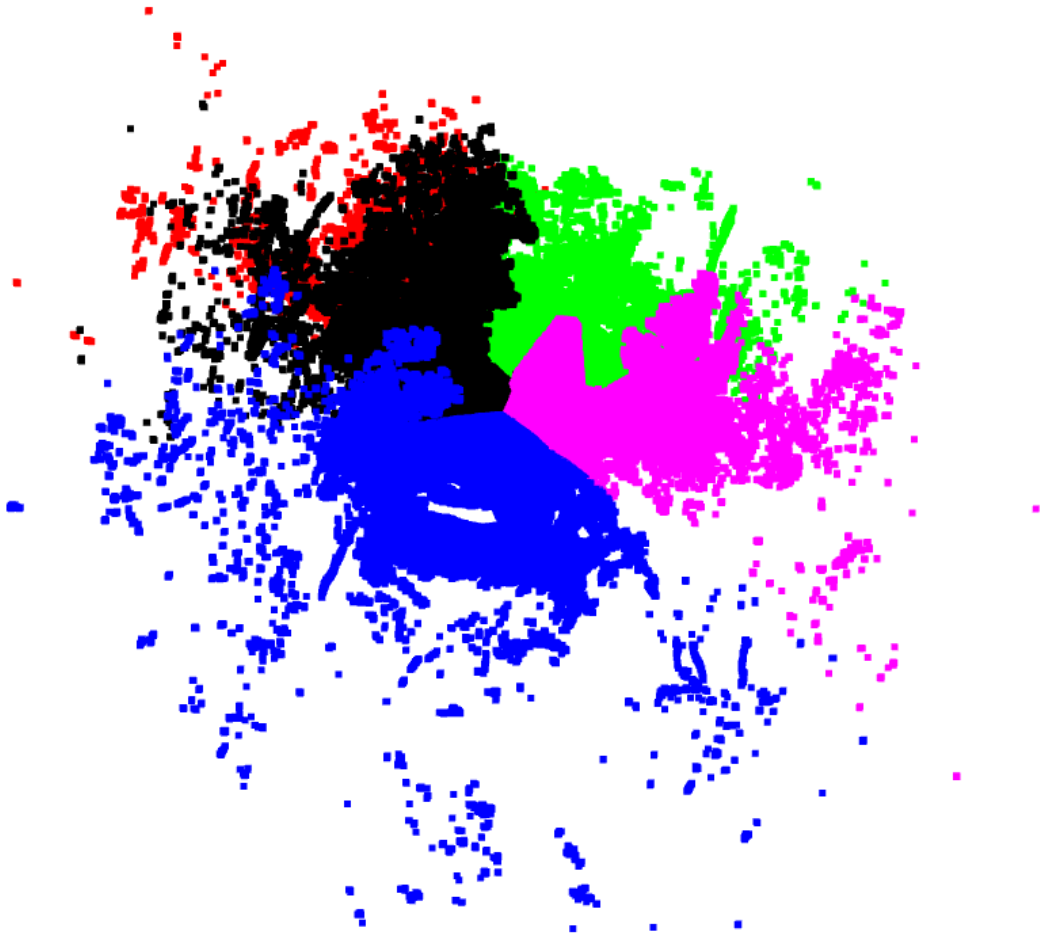
2.1 K-means

K-means je metóda relatívne jednoduchá metóda od ktorej sú ostatné segmentačné metódy odvodzované a nadstavované. Metóda k-menas ktorá je aj implementovaná ako aj funkcia KMeans v knižnici open3d , sklearn.cluster je základnou metódou segmentácie ktorá pre jednoduché dobre separované dáta k-menas nájde vhodné rozdelenie klastrov, pri zložitejších nie až tak dobre separovaných dátach ich je tiež schopná nájsť ale môžu sa vyskytnúť chyby. Napríklad, ak máme jednoduché dáta, algoritmus k-means dokáže tieto klastre bodov rýchlo označiť spôsobom, ktorý sa presne zhoduje s tým, čo by sme mohli odhadnúť okom.

Intuitívne by sme mohli očakávať, že priradenie klastrov pre niektoré body je istejšie ako iné, napríklad sa môže stať, že medzi dvoma strednými klastrami je veľmi malé prekrytie, takže nemusíme mať úplnú dôveru v priradených klastrov bodov medzi nimi. Bohužiaľ, model k-means nemá žiadnu vnútornú mieru pravdepodobnosti alebo neistoty priradenia klastrov.

Jedným zo spôsobov, ako uvažovať o modeli k-means, je umiestniť kruh (alebo vo vyšších dimenziách hyperguľu) do stredu každého zhluku s polomerom definovaným najvzdialenejším bodom v zhluku. Tento polomer funguje ako tvrdá hranica pre priradenie klastra v rámci trénovanej množiny, akýkoľvek bod mimo tohto kruhu sa nepovažuje za člena klastra. Nevýhodou k-means je že nedokáže pracovať s inými tvarmi čo môže spôsobiť prekrývanie segmentovaných množín kde niektoré body sa nachádzajú v oboch množinách. V takomto prípade sú body priradené ku klastru ktorého stred je najbližšie.

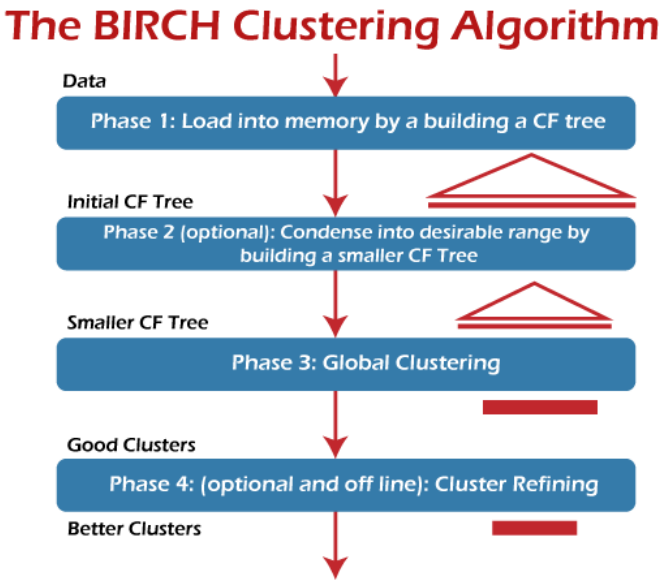
Vstupom funkcie pre algoritmus k-means ktorú sme použili „KMeans“ je počet klastrov štandardne referovaný ako „K“. Ktorej výstup sa ukladá do masky a následne fituje na dáta mračna bodov. My sme použili 5 klastrov viď. Obr.5.



Obr. 5: Mračno bodov z weba segmentované použitím k-means pre 5 klastrov.

2.2 BIRCH algoritmus

BIRCH (Balanced Iterative Reducing and Clustering hierarchies) je algoritmus na klastrovanie, ktorý sa hlavne používa na veľkých datasetoch kvôli tomu, že je rýchly a že je ľahko implementovateľný. Základný princíp tohto algoritmu je ten, že najprv klastruje do malých množín, ktoré sú následne klástrované. To znamená, že tento algoritmus nerobí klastrovanie priamo na dátach, ale je možné využiť ďalší algoritmus, ktorý bude klástrovať množiny, ktoré boli vytvorené pomocou tohto algoritmu.



Obr. 6: BIRCH algoritmus

Tento algoritmus je založený na *CF (clusterring features) tree*, ktorý je *height-balanced tree*, ktorý zhromažďuje a spravuje funkcie klastrovania a uchováva potrebné informácie o daných údajoch pre ďalšie hierarchické klastrovanie. Kvôli tomu nie je potrebné pracovať s celými vstupnými údajmi.

Stromový klaster dátových bodov je možné reprezentovať ako CF s tromi číslami:

- N – počet bodov v podskupinách
- LS – vektor sumy hodnôt bodov
- SS – suma štvorcov hodnôt bodov

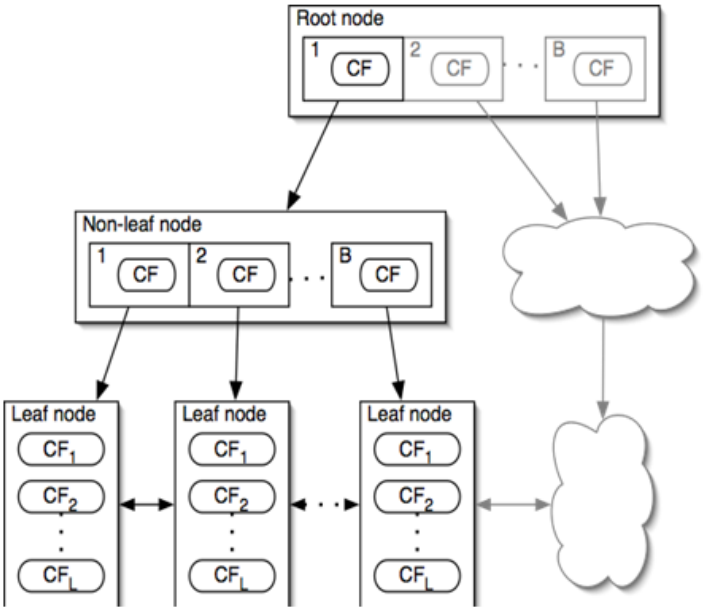
Napr. ak máme nasledovné body:

- (1, 3), (2, 5), (5, 9), (6, 7)

Tieto dáta môžeme reprezentovať ako CF nasledovne:

- N = 4
- LS = [14, 24]
- SS = [66, 164]

Na nasledovnom obrázku je znázornená štruktúra *CF tree*:



Obr. 7: Štruktúra CF tree

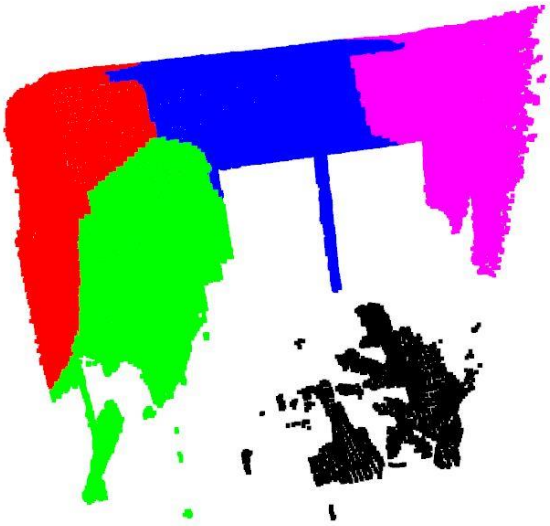
Na horeuvedenej štruktúre vidno, že sa CF strom pozostáva z koreňových uzlov, *non-leaf* uzlov a *leaf* uzlov v takom počte, aby splnili prahovú hodnotu T, ktorá predstavuje maximálny priemer polomeru stromu.

Z obrázka 6. vidno hlavné kroky, ktoré BIRCH algoritmus vykonáva počas segmentácii:

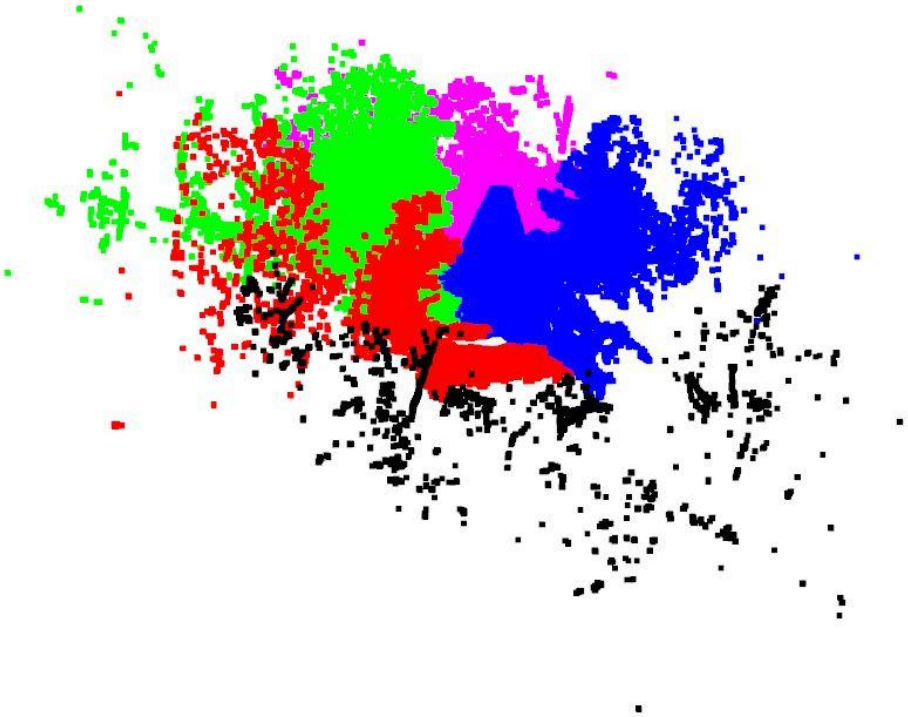
- Naskenovanie dát do pamäti
- Menenie veľkosti
- Globálne klastrovanie
- Cluster refining

Z horeuvedených krokoch, dva sú voliteľné (menenie veľkosti dát a cluster refining). Keď sa načítajú dáta, algoritmus ich naskenuje a ich fituje do CF stromu. V druhom kroku im zmení veľkosť, aby sa lepšie zmestili do CF stromu. V treťom kroku posiela CF stromy na klastrovanie s použitím iných algoritmov na klastrovanie. Nakoniec, posledný krok rieši problém, kde rovnaká hodnota môže byť pridelená viacerým uzlom.

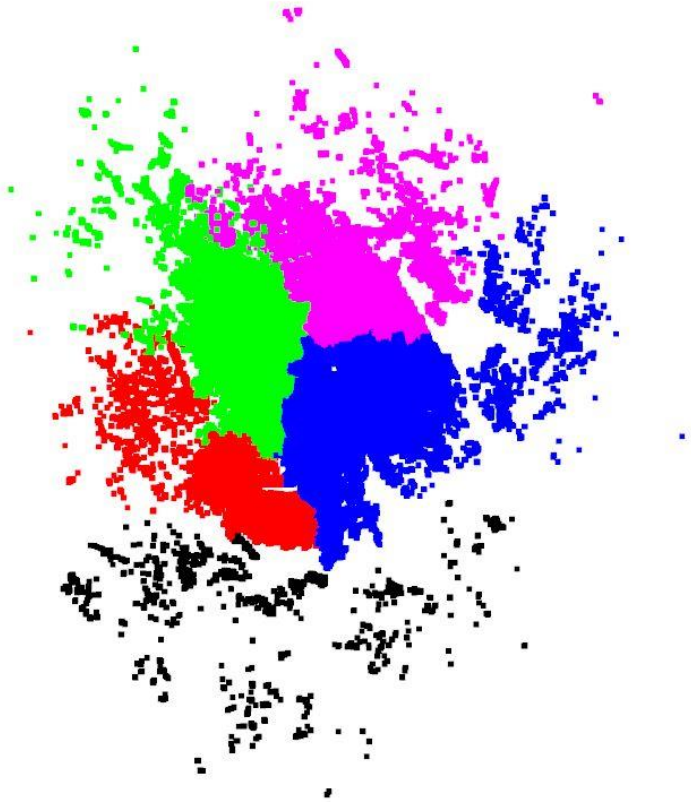
Na obr. 8 je znázornený point cloud získaný pomocou Kinectu a na obr. 9 je znázornený point cloud z weba, segmentovaný pomocou BIRCH algoritmu. Na obr. 10 je znázornený rovnaký point cloud ako na obr. 9, ale z pohľadu smerom nadol. V oboch prípadoch je počet klastrov zvolený na hodnotu 5.



Obr. 8: Point cloud z Kinecta segmentovaný na klastre pomocou BIRCH algoritmu



Obr. 9: Point cloud z weba segmentovaný na klastre pomocou BIRCH algoritmu



Obr. 10: Point cloud z weba segmentovaný na klastre pomocou BIRCH algoritmu (pohľad z hore)

Porovnaním obr. 10. a 5. vidno rozdiel segmentácie zvolených algoritmov. K-Means v tomto prípade vytvoril "ostrejšie" množiny, ktoré rozdeľuje takmer rovná priamka, kým BIRCH vytvoril klastre, ktoré nie sú až tak rozdelené. Tiež treba pripomenúť, že aj keď BIRCH by mal byť jeden z rýchlejších algoritmov segmentácie, v tomto prípade je to naopak: segmentácia pomocou BIRCH trvala výraznejšie dlhšie ako pomocou K-Means. Je možné, že väčšina času je strávená načítaním dát do modelu, lebo v samotnom programe segmentácia pomocou BIRCH algoritmu je prvá v poradí, čo výrazne zjednoduší prácu K-Means algoritmu. Ale aj keď sa zmenilo poradie algoritmov v kóde, BIRCH aj ďalej trval výraznejšie dlhšie než K-Means (čas potrebný na zbehnutie algoritmu sa takmer nezmenil).