Obrázok, na ktorom je text

Automaticky generovaný popis

Na získanie mračna bodov sme použili script s tutoriálu na nainštalovanie Kinectu. Pomoocu tohto scriptu sme dostali mračno bodov vo formáte *.pcd.* Tento typ sme museli preformátovať pomocou Open3D knižnice a vytvoriť si súbor vo formáte *.ply* ktorý sme použili následne v nasledujúcich úlohách.

V druhej úlohe sme mali vytvoriť algoritmus ktorý využije RNSAC algoritmus na vyčistenie mračna bodov

Obrázok, na ktorom je text

Automaticky generovaný popis

V pcd načítavame už uložene ply súbory s predošleho test scriptu nato používame Open3D knižnicu. Nasledne použijeme RNASEC algoritmus na vyčistenie mračna bodov od šumov.

Metóda remove\_statistical\_outlier prijíma dva argumenty:

nb\_neighbors: počet susedov, ktorí majú byť zahrnutí do výpočtu štatistického outlieru pre každý bod.

std\_ratio: maximálny počet štandardných odchýlok od strednej hodnoty, ktorý môže byť bod vzdialený od priemeru susedov, aby bol stále považovaný za platný bod.

Výstupom tejto metódy sú dva objekty:

pcd\_cleaned: mračno bodov bez odstránených outlierov.

ind: indexy bodov, ktoré boli označené ako outlieri a boli odstránené z pôvodného mračna b odov.

Obrázok, na ktorom je text

Automaticky generovaný popis

Vytvoríme prazdne kontajnery kde si neskôr uložíme mračno bodov a vyčistené mračno bodov aby sme ich vedeli porovnať

Obrázok, na ktorom je text

Automaticky generovaný popis

Vytvoríme okno kde vložíme mračno bodov ktoré chceme zobraziť v okne. Zobrazujeme mračno bodov nevyčistené a vyčistené.

Obrázok, na ktorom je text

Automaticky generovaný popis

Vytvoríme nové interaktívne okná kde nám pôsobí funkcia ktorá aktualizuje okno pri každom jednom pohybe. Môže sa to použiť rôzna geometria ale my používame naše mračno bodov a vyčistené mračno bodov.

Obrázok, na ktorom je text

Automaticky generovaný popis

Táto časť je rovnaka ako s predošlého kódu lebo potrebujeme vytvoriť znova čísté mračno bodov a klasické mračno bodov na ktoré nasledne môžeme použiť K-means a DBSCAN algoritmus na segmentáciu.

RANSAC (Random Sample Consensus) je algoritmus používaný na odstránenie šumu z dát. Často sa používa v počítačovom videní, spracovaní obrazu a strojovom učení na riešenie úloh ako detekcia a matchovanie objektov, rekonštrukcia 3D modelov a kalibrácia kamerových systémov.

RANSAC algoritmus pracuje na základe náhodného výberu vzoriek z dát, ktoré sú potom použité na vytvorenie modelu. Tento model sa použije na označenie bodov, ktoré sú v súlade s modelom ako "inliers" (čiže patria k objektu) a ostatné body ako "outliers" (šum).

Postup algoritmu:

1. Vyberieme náhodne minimálny počet bodov potrebných na vytvorenie modelu (napríklad, v prípade lineárneho modelu potrebujeme minimálne dva body).
2. Vypočítame parametre modelu pomocou vybraných bodov.
3. Označíme body, ktoré sú v súlade s modelom ako "inliers" a ostatné body ako "outliers".
4. Opakujeme kroky 1 až 3, až kým nedosiahneme požadovanú presnosť alebo počet iterácií.
5. Získaný model a označené body sa použijú na riešenie požadovanej úlohy.

RANSAC algoritmus je schopný riešiť aj problémy s outliers v dátach, keďže náhodný výber bodov minimalizuje vplyv šumu na výsledný model. Výhodou je aj jeho jednoduchosť a použiteľnosť na rôzne typy modelov.

Nevýhodou RANSAC algoritmu je, že môže byť veľmi pomalý pri veľkom počte dát alebo pri výbere zlého modelu. Okrem toho, RANSAC môže vrátiť nesprávny výsledok, ak šumové body vytvoria iný model ako skutočné objekty.

Obrázok, na ktorom je text

Automaticky generovaný popis

K-means algoritmus je jedným z najjednoduchších algoritmov pre úlohu clusteringu, teda segmentáciu priestoru do klastrov. Cieľom algoritmu je rozdeliť dáta do určitého počtu K skupín (klastrov), pričom každý bod dátového priestoru patrí do jedného z týchto K klastrov.

Postup algoritmu:

1. Zvolíme počet klastrov K, do ktorých chceme dáta rozdeliť.
2. Náhodne vyberieme K bodov (centroidy) ako počiatočné stredy klastrov.
3. Pre každý bod v dátach určíme, ktorému centroidu je najbližší (napríklad podľa Euklidovskej vzdialenosti).
4. Pre každý vytvorený klaster vypočítame nový stred, teda priemer všetkých bodov patriacich do tohto klastra.
5. Bodom centroidov priradíme nové súradnice, teda súradnice nových stredov klastrov.
6. Opakujeme kroky 3 až 5, až kým sa súradnice centroidov stabilizujú, teda zmena súradníc v určitej iterácii je menšia ako určená tolerancia.

Výsledkom sú K klastrov s ich stredmi a prislúchajúcimi bodmi v jednotlivých klastroch.

K-means algoritmus sa používa najmä pre dáta, ktoré sú dobre rozdelené a majú jasné klastre. Keďže algoritmus vyžaduje náhodný výber počiatočných stredov klastrov, môže viesť k rôznym výsledkom, pretože vyber počiatočných stredov môže ovplyvniť výsledné klastre. Preto sa niekedy používa opakované spúšťanie algoritmu s rôznymi náhodnými počiatočnými stredmi a výberom toho s



DBSCAN (Density-Based Spatial Clustering of Applications with Noise) je algoritmus pre úlohu clusteringu, teda segmentáciu priestoru do klastrov. Jeho hlavnou výhodou je, že dokáže pracovať s dátami, ktoré majú rôzne tvary a hustotu, a zároveň dokáže detekovať a odlíšiť bodové šumy.

Postup algoritmu:

1. Zvolíme dva parametre, Eps a MinPts, ktoré určujú, aké sú vzdialenostné a počtové kritériá pre vytvorenie klastrov.
2. Náhodne vyberieme bod z dátového priestoru, ktorý ešte nepatrí do žiadneho klastrov.
3. Nájdeme všetky body, ktoré sú vzdialené najviac Eps od vybraného bodu.
4. Ak je počet týchto bodov väčší alebo rovný ako MinPts, vytvoríme nový klaster a priradíme mu vybraný bod a všetky body, ktoré sme našli v kroku 3.
5. Ak sme ešte nespájali žiadne body do klastrov, vyberieme nový náhodný bod a opakujeme kroky 3 a 4. Ak sme už niektoré body priradili do klastrov, hľadáme najbližších susedov iba medzi bodmi, ktoré ešte nie sú súčasťou žiadneho klastrov.
6. Pokračujeme v krokoch 3 až 5, až kým nepriradíme všetky body do klastrov.
7. Výsledkom sú klastre, ktoré sme vytvorili, a body, ktoré sme označili ako šumy.

Hlavným kritériom DBSCAN algoritmu je hustota bodov v dátovom priestore. Body, ktoré majú veľa susedov v určitej vzdialenosti, patria do rovnakej skupiny (klasteru). Týmto spôsobom algoritmus dokáže pracovať s dátami, ktoré majú rôzne tvary a hustotu, a dokáže detekovať a odstrániť bodové šumy.

DBSCAN algoritmus má aj niekoľko nevýhod. Jeho výkon závisí od výberu parametrov Eps a MinPts, ktoré môžu byť ťažké nastaviť pre neznáme dáta. Okrem toho môže mať problémy s nájdením klastrov, ktoré majú rôznu hustotu.

Parameter **print\_progress=True** zabezpečí výpis informácií o priebehu klastrovania na konzolu.

Obrázok, na ktorom je text

Automaticky generovaný popis

Tento kód vytvára náhodné farby pre každý klastrový label

Následne sa pre každý label vytvorí náhodná farba pomocou funkcie np.random.rand(3), ktorá vráti pole s tromi náhodnými číslami medzi 0 a 1. Tieto farby sú potom uložené do 2D poľa colors, pričom prvý index poľa predstavuje label klastrového algoritmu a druhý index farbu, t.j. [label][0] predstavuje červenú zložku farby pre daný label, [label][1] predstavuje zelenú zložku a [label][2] predstavuje modrú zložku.

Obrázok, na ktorom je text

Automaticky generovaný popis

Pcd\_cleaned.colors slúži na priradenie farieb k jednotlivým bodom v oblaku a o3d.utility.Vector3dVector slúži na vytvorenie 3D vektoru farieb. Tento 3D vektor farieb sa priradí k pcd\_cleaned.colors, čím sa farby priradia k jednotlivým bodom v oblaku.

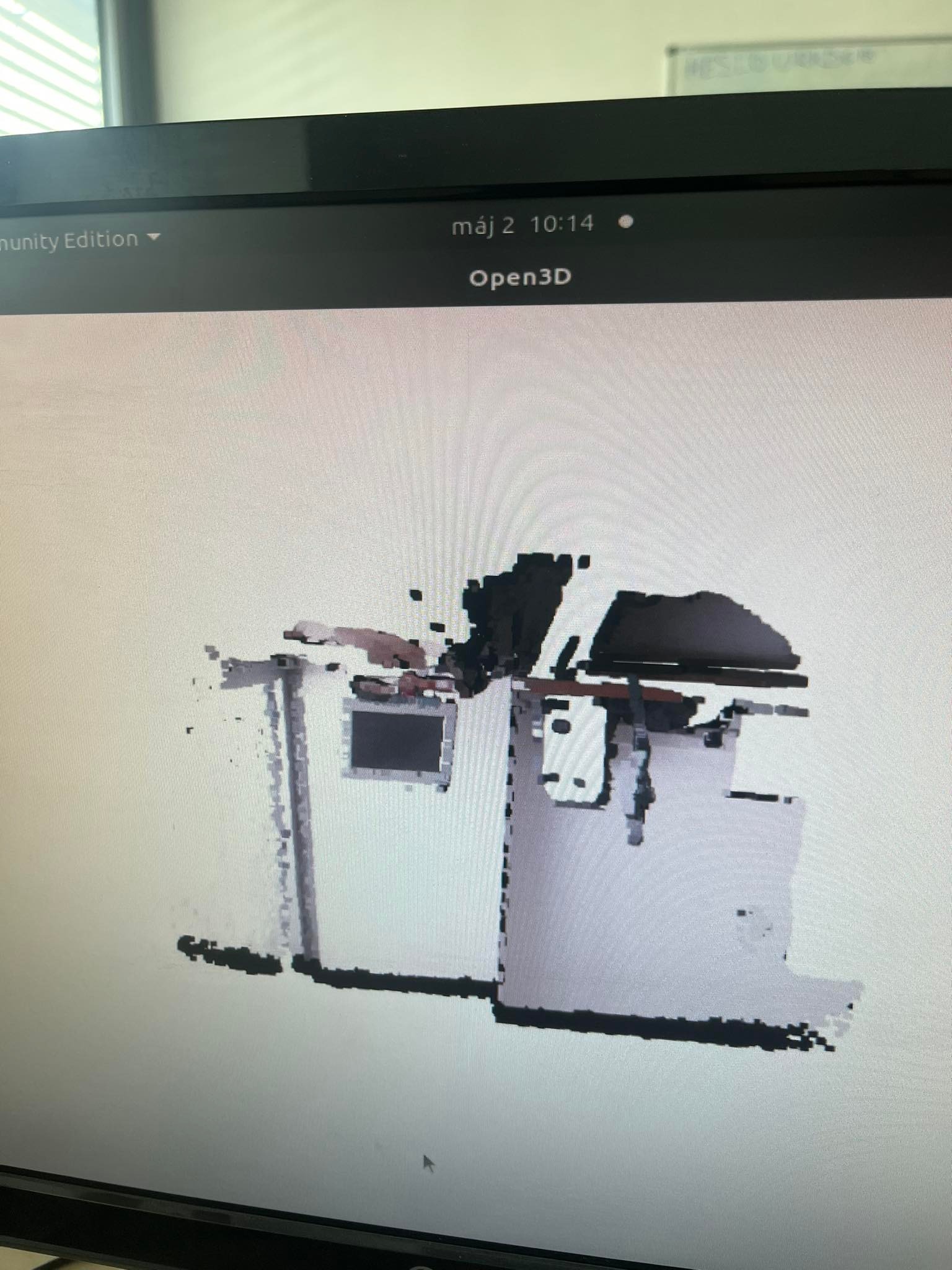
Colors[kmeans\_labels] vráti pole farieb pre každý bod v pcd\_cleaned oblaku, pričom každému labelu je priradená farba z premennej colors, ktorá bola vypočítaná v predchádzajúcom kóde.

Obrázok, na ktorom je text

Automaticky generovaný popis

Zobrazovanie sme vyriešili rovnako ako v predošlom kóde.





Obrázok, na ktorom je text, vnútri, elektronika, vizitka

Automaticky generovaný popisObrázok, na ktorom je text, elektronika, displej, počítač

Automaticky generovaný popis

