Dokumentácia PVSO

Aleš Melichar, Jakub Mihálik

May 9, 2023

Dáta

Mračno bodov (Point Cloud), ktorým sa v tejto dokumentácií budeme zaoberať vyzerá nasledovne:

Figure 1: *Základný point cloud*

Každý bod v point cloude reprezentuje súradnicu (X, Y, Z) a môže mať dodatočné atribúty ako sú farba, intenzita a čas.

RANSAC

Na očistenie point cloud-u od odkrajových bodov použijeme algoritmus RANSAC. RANSAC funguje na základe náhodného výberu minimálneho počtu bodov zo súboru dát. V našom prípade chceme nájsť rovinu v 3D priestore.

Náhodne vyberieme 3 body v priestore z ktorých vypočítame rovnicu roviny. Teraz ohodnotíme rovinu tak, že sa pozrieme koľko bodov z point cloudu je v súlade s rovinou napr. pomocou euklidovskej vzdialenosti. Takto prejde Ak je počet bodov v súlade s rovinou (INLIERS) vačší ako predchádzajúci počet bodov tak budeme túto rovinu považovať za najlepšiu. Toto opakujeme xy iterácií. Body, ktoré nie sú v súlade s rovinou nazývame OUTLIERS (patrí sem aj šum).

Po spustení kódu dostávame obr. č. 2

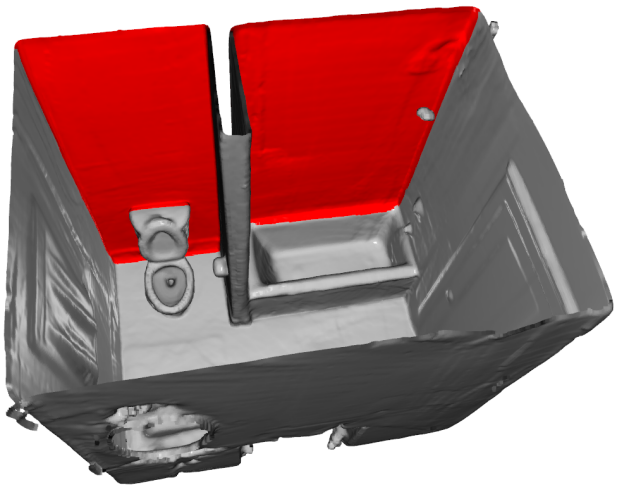

Figure 2: *Point cloud s rovinou*

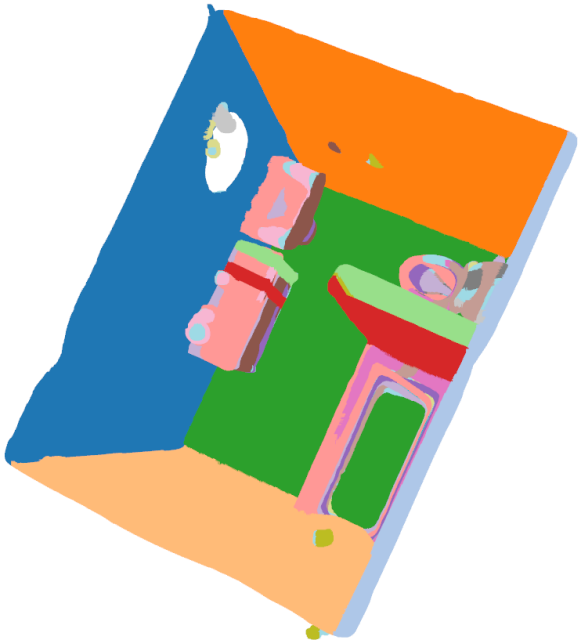

Figure 3: *Point cloud s rovinou DBSCAN*

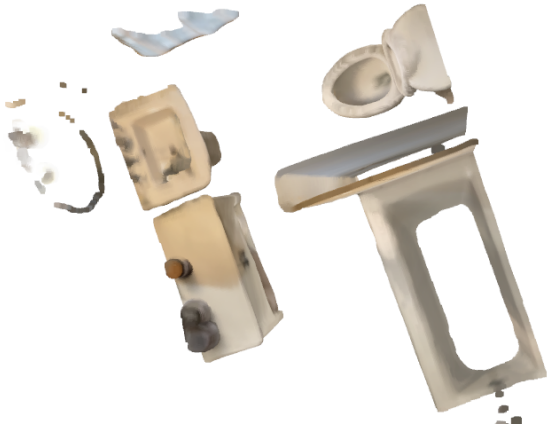

Figure 4: *Point cloud bez rovín*

Figure 5: *Point cloud bez rovín DBSCAN*

DBSCAN

DBSCAN je algoritmus zhlukovania, ktorý automaticky rozdeľuje dáta do skupín (zhlukov) na základe ich hustoty. Hustota je definovaná ako koncentrácia bodov v určitej oblasti. Algoritmus funguje tak, že najprv sa vyberie náhodný bod z dátového súboru. Potom sa overí, či daný bod má dostatočný počet susedov v určitom okolí ϵ. Ak áno, vytvorí nový zhluk a pridá doň tento bod a všetkých jeho susedov. Tento proces sa opakuje, až kým sa nedajú pridať ďalšie body do zhluku. Ak sa nevytvoril zhluk, bod sa označí ako šum. Po spustení kódu dostávame obr. č. 3 Vymažme roviny použitím kódu kódu dostávame obr. č. 4 kde po spustení DBSCAN dostávame obr. č. 5

BIRCH

BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) je algoritmus zhlukovania určený pre veľké dátové sady. Základná myšlienka BIRCH je, že vstupné dáta prechádza iba raz a vytvára kompaktnú a presnú reprezentáciu dát, ktorú nazývame CF (Clustering Feature) strom.

Každý uzol v CF strome obsahuje niekoľko zhlukových prvkov, ktoré sú reprezentované ako trojice hodnôt (N, LS, SS), kde:

- N je počet vzoriek v zhluku
- LS je lineárna suma vzoriek v zhluku
- SS je suma štvorcov vzoriek v zhluku

Tieto trojice hodnôt sa dajú efektívne aktualizovať pri pridávaní nových prvkov a umožňujú výpočet strednej hodnoty a rozptylu zhluku bez nutnosti prechádzať všetkými prvkami zhluku.

Tu je základný postup algoritmu BIRCH:

1. Inicializácia: Vytvára sa prázdny CF strom.
2. Vytvorenie CF stromu: Pre každý bod v dátach sa aktualizuje CF strom. Bod sa pridá do najbližšieho CF zhluku v strome, ak je to možné bez prekročenia určitej hranice (threshold). Ak to nie je možné, vytvorí sa nový zhluk.
3. Zlepšenie CF stromu: Môže sa voliteľne vykonať dodatočná fáza zhlukovania, napríklad pomocou aglomeratívnej hierarchickej metódy, na zlepšenie kvality zhlukov.

BIRCH je efektívny pre veľké dátové sady, pretože vytvára kompaktnú reprezentáciu dát, ktorá sa zmestí do pamäte, a vstupné dáta prechádza len raz. Má však obmedzenia pri práci s vysokodimenzionálnymi dátami a môže mať problémy pri detekcii zhlukov rôznych tvarov a veľkostí.

Figure 6: *Point cloud bez rovín - BIRCH*

```
pcd = o3d.io.read_point_cloud("bathroom.ply")
pcd.estimate_normals(search_param=o3d.geometry.KDTreeSearchParamHybrid(radius=0.1,
                                                    max_nn=16), fast_normal_computation=True)
pcd.paint_uniform_color([0.6, 0.6, 0.6])

# RANSAC algoritmus
# distance_threshold - najmensia vzdialenost medzi rovinou a bodom
# ransac_n - pocet bodov (3 kedze rovina)
# num_iterations - pocet iteracii
plane_model, inliers = pcd.segment_plane(
            distance_threshold=0.06,
            ransac_n=3,
            num_iterations=1000)
# dostavame plane_model co je rovnica roviny (nie tak uplne): ax + by + cz + d = 0
[a, b, c, d] = plane_model
# Ziskanie inlierov a outlierov
inlier_cloud = pcd.select_by_index(inliers)
# V podstate invertnuty inlier_cloud
outlier_cloud = pcd.select_by_index(inliers, invert=True)
# Farba pre inliery a outliery
inlier_cloud.paint_uniform_color([1.0, 0, 0])
outlier_cloud.paint_uniform_color([0.6, 0.6, 0.6])
# Vykreslenie roviny v point cloude
o3d.visualization.draw_geometries([inlier_cloud, outlier_cloud])

```

Figure 7: *Kód pre rovinu*

```
segment_models={}
segments={}
max_plane_idx=40
rest=pcd
d_threshold=0.001
for i in range(max_plane_idx):
  if len(rest.points) < 3:
    break
  colors = plt.get_cmap("tab20")(i)
  plane_model, inliers = rest.segment_plane(distance_threshold=0.05, ransac_n=3,
                                                    num_iterations=3000)
  segments[i] = rest.select_by_index(inliers)
  labels = np.array(segments[i].cluster_dbscan(eps=d_threshold*10, min_points=30))
  candidates = [len(np.where(labels == j)[0]) for j in np.unique(labels)]
  best_candidate = int(np.unique(labels)[np.where(candidates == np.max(candidates))[0]][0])
  print("the best candidate is: ", best_candidate)
  rest = rest.select_by_index(inliers,invert=True) + segments[i].select_by_index(
                              list(np.where(labels != best_candidate)[0]))
  segments[i] = segments[i].select_by_index(list(np.where(labels == best_candidate)[0]))
  segments[i].paint_uniform_color(list(colors[:3]))
  print("pass", i+1, "/", max_plane_idx, "done.")
o3d.visualization.draw_geometries([segments[i] for i in segments.keys()]+[rest])

```

Figure 8: *Kód pre DBSCAN*

```
    pcd = o3d.io.read_point_cloud("bathroom.ply")
    plane_model, inliers = pcd.segment_plane(distance_threshold=0.04, ransac_n=3,
                                             num_iterations=3000)
    inlier_cloud = pcd.select_by_index(inliers)
    outlier_cloud = pcd.select_by_index(inliers, invert=True)

    while True:
      plane_model, inliers = outlier_cloud.segment_plane(distance_threshold=0.04, ransac_n=3,
                                             num_iterations=3000)
      inlier_cloud = outlier_cloud.select_by_index(inliers)
      outlier_cloud = outlier_cloud.select_by_index(inliers, invert=True)

      if len(outlier_cloud.points) < 230000:
        break

    pcd_without_planes = outlier_cloud
    o3d.visualization.draw_geometries([pcd_without_planes])
    # DBSCAN
    labels = np.array(pcd_without_planes.cluster_dbscan(eps=0.1, min_points=10))
    max_label = labels.max()
    colors = plt.get_cmap("tab20")(labels / (max_label if max_label > 0 else 1))
    colors[labels < 0] = 0
    pcd_without_planes.colors = o3d.utility.Vector3dVector(colors[:, :3])
    o3d.visualization.draw_geometries([pcd_without_planes])

```

Figure 9: *Kód pre DBSCAN bez rovín*

```
    pcd = o3d.io.read_point_cloud("bathroom.ply")
    plane_model, inliers = pcd.segment_plane(distance_threshold=0.07, ransac_n=3,
                                             num_iterations=30000)
    inlier_cloud = pcd.select_by_index(inliers)
    outlier_cloud = pcd.select_by_index(inliers, invert=True)

    while True:
      plane_model, inliers = outlier_cloud.segment_plane(distance_threshold=0.04, ransac_n=3,
                                             num_iterations=30000)
      inlier_cloud = outlier_cloud.select_by_index(inliers)
      outlier_cloud = outlier_cloud.select_by_index(inliers, invert=True)

      if len(outlier_cloud.points) < 274000:
        break

    pcd_without_planes = outlier_cloud
    data_points = np.asarray(pcd_without_planes.points)
    # Train Birch model on the data points
    birch = Birch(threshold=0.8, branching_factor=5, n_clusters=None)
    birch.fit(data_points)

    labels = birch.predict(data_points)
    max_label = labels.max()
    colors = plt.get_cmap("tab20")(labels / (max_label if max_label > 0 else 1))
    colors[labels < 0] = 0
    pcd_without_planes.colors = o3d.utility.Vector3dVector(colors[:, :3])
    o3d.visualization.draw_geometries([pcd_without_planes])

```

Figure 10: *Kód pre BIRCH bez rovín*

Zdroje

https://www.slideshare.net/KailashShaw/birch-algorithm-with-solved-example
https://analyticsindiamag.com/guide-to-birch-clustering-algorithmwith-python-codes/
https://towardsdatascience.com/how-to-automate-3d-point-cloud-segmentation-and-clustering-with-python-
https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html
https://scikit-learn.org/stable/modules/generated/sklearn.cluster.Birch.html
https://gisgeography.com/point-cloud/#:~:text=Point%20Cloud%20Attributes&text=At%20the%
20most%20basic%20level,color%2C%20and%20time%20as%20well.