VJEŽBA 6: ODREĐIVANJE DOMINANTNE RAVNINE NA 2.5D SLICI RANSAC METODOM

I. Cilj vježbe: *Naučiti primijeniti RANSAC postupak u određivanju dominante ravnine iz 2.5D slike.*

II. Opis vježbe:

Robot se može smatrati autonomnim ako je sposoban obavljati zadane zadatke u nestrukturiranim okolinama bez ljudske intervencije. U skoroj budućnosti se očekuje da roboti budu u velikoj mjeri samostalni. Dakle, od njih će se zahtijevati visoko razvijena percepcija i način predstavljanja svoje radne okoline. Ovisno o traženom zadatku, robot mora moći prepoznati površinu ili plohu od interesa, npr. pod prilikom kretanja, površinu stola ukoliko treba uzeti ili spustiti zadani predmet, zid ili bilo koju drugu površinu koju treba obraditi na odgovarajući način kao što je bojanje, brušenje i sl.
U ovoj vježbi, primijenit će se RANSAC metoda za određivanje dominante ravnine iz oblaka točaka (2.5D slike) dobivenog pomoću RGB-D senzora (Kinect).

Potrebno je implementirati RANSAC postupak opisan u prilogu, te detektirati i označiti dominantne ravnine na RGB-D slikama priloženim uz ovu vježbu. Zbog visoke vremenske složenosti RANSAC algoritma, poželjno je omogućiti izvođenje na više procesorskih jezgri.

III. Rad na vježbi:

a) Implementirati RANSAC postupak za detektiranje dominantne ravnine opisan u prilogu vježbe.
b) Učitati i prikazati RGB i dubinsku sliku. (Za učitavanje dubinske slike upotrijebiti funkciju *read_kinect_pic*)
c) Detektirati dominantu ravninu, označiti i prikazati na dubinkoj slici.
d) Ponoviti postupak (b. i c.) za sve slike priložene uz ovu vježbu.

IV. Bonus zadatak:

Prikladno izmijeniti algoritam napravljen pod **III. Rad na vježbi**, te segmentirati cijelu jednu odabranu dubinsku sliku na ravninske segmente.

Prilog:

Random Sample Consensus (RANSAC) metoda za detekciju dominantne ravnine iz 2.5D slike

Neka je $\boldsymbol{S}$ oblak točaka
Ponavljaj korake 1 do 6 određeni broj puta.

1. $T^* \leftarrow \varnothing$, $R^* \leftarrow \varnothing$
2. Ponavljaj korake a) i b) dok se ne dobije dobro kondicionirana ravnina
   a. Nasumično izaberi tri točke (u_i, v_i, d_i), i = 1,2,3 iz $\boldsymbol{S}$.
   b. Pomoću jednažbe (2), odredi ravninu R opisanu parametrima a, b i c koja prolazi kroz te točke.

      Ravnina R, s parametrima a, b i c, u uvd prostoru se može opisati izrazom (1)

      $$au + bv + c = d \tag{1}$$

      Ravnina R koja prolazi kroz tri točke (u_i, v_i, d_i), i = 1,2,3 zadovoljava sljedeću jednažbu (2):

      $$\begin{bmatrix} u_1 & v_1 & 1 \\ u_2 & v_2 & 1 \\ u_3 & v_3 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix}, \tag{2}$$

3. Odredi skup točaka $T \subseteq \boldsymbol{S}$ koje leže na ravnini R takav da vrijedi

   $$|d' - (au' + bv' + c)| \leq \varepsilon$$

4. Ako je $|T| > |T^*|$, onda
5. $T^* \leftarrow T$
6. $R^* \leftarrow R$
7. Rezultat je dominantna ravnina R^* odnosno skup točaka T^* iz dubinske slike koje leže na dominantnoj ravnini.

Prilog:

Korištenje biblioteka *tqdm* i *multiprocessing*:

Metoda *tqdm* iz biblioteke *tqdm* služi za ispis trenutnog stanja izvođenja neke petlje, primjerice:

```
for i in tqdm(range(10)):
   func(i)
```

Biblioteka *multiprocessing* služi za izvođenje programa na više procesorskih jezgri.
Klasa *Pool* služi za kreiranje bazena procesa koji će izvršavati zadatke koji im se podnesu. Za potrebe izvođenja ove vježbe, potrebna su prva tri parametra konstruktora klase *Pool*:

```
Pool(processes, initializer, initargs)
```

Parametar *processes* označava broj procesa koji se želi koristiti, parametar *initializer* predstavlja funkciju koju će pokrenuti svaki proces prije izvođenja svog dijela posla, parametar *initargs* predstavlja listu (ili neki drugi *iterable* (*range*, *tuple*…)) parametara koji će se predati funkciji *initializer*. Funkcija *initializer* je potrebna ukoliko se žele koristiti globalne varijable na Windows operacijskom sustavu (na Unix operacijskim sustavima se može lakše izbjeći korištenje funkcije *initializer*). Primjerice, ako imamo globalnu varijablu *coef* kojoj smo vrijednost postavili unutar funkcije *main* gdje kreiramo i bazen procesa, potrebno je napraviti funkciju *initializer* koja će postaviti vrijednost globalne varijable *coef* svakom procesu:

```
def my_first_initializer(coef_):
   global coef
   coef = coef_
```

Uz ovako definiranu funkciju *initalizer* možemo kreirati bazen 4 procesa na sljedeći način:

```
Pool(4, my_first_initializer, [coef])
```

Uz ovako definiran bazen procesa, u svim procesima možemo koristiti globalnu varijablu *coef*.

Možemo definirati funkciju *func* na sljedeći način:

```
def func(x):
   global coef
   return coef * x
```

Za ovako definiranu funkciju *func* koja prima jedan parametar *x* i vraća jednu vrijednost, možemo izvesti tu funkciju koristeći više procesa s listom parametara *args* na sljedeći način:

```
with Pool(4, my_first_initializer, [coef]) as p:
   res = p.map(func, args)
```

Nakon izvođenja ovih linija, u varijablu res će nam biti spremljena lista vrijednosti koje nam vrati funkcija *func* za zadanu listu parametara *args*. Ekvivalent ovome bi bio sljedeći kod:

```
res = []
for arg in args:
   res.append(func(arg))
```

Lista parametara *args* može biti bilo kakav *iterable* objekt, primjerice *range(n)*.

Primjer korištenja biblioteke *multiprocessing*

```
from multiprocessing import Pool

def func(x):
   global coef
   return coef * x

def my_first_initializer(coef_):
   global coef
   coef = coef_

def main():
   coef = 4
   args = [0, 2, 4, 6]
   with Pool(4, my_first_initializer, [coef]) as p:
      res = p.map(func, args)
   print(res)

if __name__ == '__main__':
   main()
```

Ovaj primjer će ispisati:
[0, 8, 16, 24].

Ukoliko želimo izvoditi funkciju *func* pomoću više procesa, ali u isto vrijeme i ispisivati stanje izvođenja, možemo to učiniti na sljedeći način:

```
from tqdm import tqdm

(…)

with Pool(4, my_first_initializer, [coef]) as p:
   res = list(tqdm(p.imap(func, args), total=len(args)))
```

Više o biblioteci *multiprocessing* možete pročitati na sljedećoj poveznici:
https://docs.python.org/3/library/multiprocessing.html