Klasifikacija objekata na RGB-D slikama primjenom metode skupa funkcija oblika

Varga, Hrvoje

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Preddiplomski studij računarstva

KLASIFIKACIJA OBJEKATA NA RGB-D SLIKAMA PRIMJENOM METODE SKUPA FUNKCIJA OBLIKA

Završni rad

Hrvoje Varga

Osijek, 2018.

SADRŽAJ

1. UVOD

Jedan od osnovnih problema svakog robota koji je namijenjen za rad u nestrukturiranim okolinama, kao što su kućanstva, zdravstvene ustanove, poljoprivredne površine i sl., je prepoznavanje objekata na slici jer je potrebno odvojiti objekt od pozadine, te ga klasificirati u jednom od prethodno naučenih klasa objekata. Postoje razne metode koje se koriste za klasifikaciju objekata kao što su The Viewpoint Feature Histogram, The Ensemble of Shape Function (u daljnjem tekstu ESF), te Global Fast Point Feature Histogram. Zadatak ovog završnog rada je klasificiranje objekata na RGB-D slikama korištenjem metode skupa funkcija oblika. U drugom poglavlju govori se o klasifikaciji objekata na slikama i raznim funkcijama koje se koriste za klasifikaciju, kao i za ovaj rad najznačajniju funkciju ESF. Treće poglavlje opisuje programsku implementaciju klasifikacije objekta sa slike, te opis biblioteke koja je korištena u realizaciji. U četvrtom poglavlju je opisana evaluacija programa na 3DNet ispitni podatkovni skup.

1.1 Zadatak završnog rada

Pomoću implementacije metode skupa funkcija oblika (Ensamble of Shape Functions - ESF) sadržane u okviru programske biblioteke Point Cloud Library (PCL) izraditi program za klasifikaciju objekata na RGB-D slikama i ispitati ga odgovarajućim pokusima.

2. KLASIFIKACIJA 3D OBJEKATA

Temelj klasifikacije 3D objekata je pronalazak sličnosti između dva različita oblaka točaka, s time da jedan oblak sadržava objekt koji tražimo, a drugi predstavlja dio scene snimljene 3D senzorom. Objekti se prikazuju kao skupovi točaka u prostoru. Postoji mnogo različitih deskriptora koji se koriste za klasifikaciju, te svaki od njih ima svoju određenu metodu. Neke od metoda se razlikuju ovisno o kutu iz kojeg se promatra objekt, neke ovise o udaljenosti između točaka. Ostali se razlikuju u tome jesu li potrebne normale, te koliki je minimalni broj pogleda na objekt. Normala je vektor koji je okomit na površinu objekta u okolini neke točke. Postoji jedna veća podjela deskriptora na lokalne i globalne. Razlika je u tome da se lokalni deskriptori izvode za točke u okolini koja je prethodno određena, te su sporiji, dok globalni deskriptori određuju točke modela uz prethodno pojednostavljenu scenu, kao na primjer razdvajanje objekata. Globalni deskriptori su brži, ali im je potrebna pojednostavljena scena [1]. Tablica 2.1 prikazuje često korištene deskriptore i njihova glavna svojstva kao što su tip i veličina deskriptora koji vraćaju.

Ime	Tip	Veličina deskriptora
PFH(Point Feature Histogram)	Localni	125
FPFH(Fast Point Feature Histogram)	Lokalni	33
RSD(Radius-Based Surface Descriptor)	Lokalni	289
3DSC(3D shape Context)	Lokalni	1980
USC(Unique Shape Context)	Lokalni	1960
SHOT(Signatures of Histogram of Orientations)	Lokalni	352
Spin image	Lokalni	153
RIFT(Rotation-Invariant Feature Transform)	Lokalni	32
NARF(Normal Aligned Radial Feature)	Lokalni	36
RoPS(Rotational Projection Statistics)	Lokalni	135
VFH(Viewpoint Feature Histogram)	Globalni	308
CVFH(Clustered Viewpoint Feature Histogram)	Globalni	308
OUR-CVFH(Oriented, Unique and Repeatable Clustered Viewpoint Feature Histogram)	Globalni	308
ESF(Ensamble of Shape Function)	Globani	640
GFPFH(Global Fast Point Feature Histogram)	Globalni	16
GRSD(Global Radius-Based Surface Descriptor)	Globalni	21

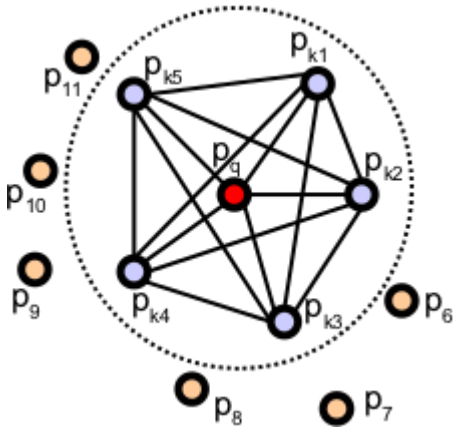
Tablica 2.1 3D deskriptori.

2.1 Lokalni deskriptori

Lokalni deskriptori kao ulazni podatak imaju skup 3D točaka, koje se nazivaju ključne točke. Oni opisuju kakva je geometrijska okolina *ključnih točaka*. U ovom radu, ključnu točku zajedno s njezinom lokalnom okolinom nazvat ćemo značajka (engl. *feature*). Spori su jer se mogu koristiti za složene scene i potrebno im je prethodno definirati okolinu u kojoj se mogu kretati odnosno uspoređivati. Najviše se koriste za prepoznavanje objekata, ali i za njihovo detektiranje.[1]

2.1.1 PFH(Point Feature Histogram)

PFH deskriptor pokušava opisati informacije u geometrijskom rasporedu točaka i analizira razliku između smjerova normala.

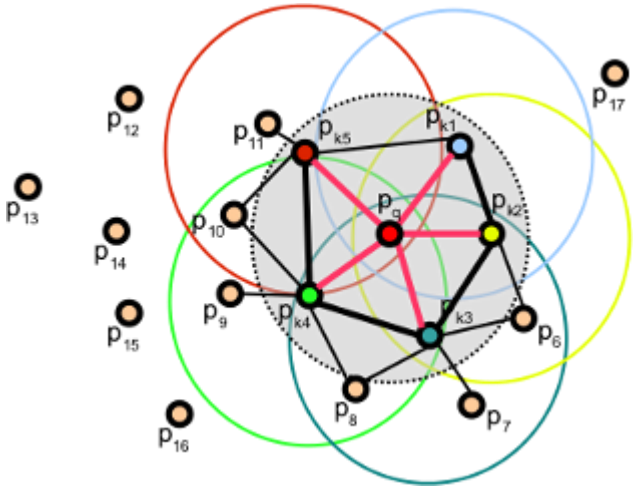


Sl. 2.1 Parovi točaka uspostavljeni prilikom računanja PFH. Slika je preuzeta iz [1].

Ulazne varijable su točke, normale, metode traženja susjeda i radijus u kojem se susjedi traže, dok je izlazna varijabla PFH deskriptor kojih ima 125 elemenata [1].

2.1.2 FPFH(Fast Point Feature Histogram)

FPFH uvažava samo izravnu povezanost između trenutne ključne točke i njezinih susjeda, zanemarujući dodatne poveznice između susjeda. Koordinatni sustav značajke i kutne varijable izračunavaju se na neki od uobičajenih načina[1].

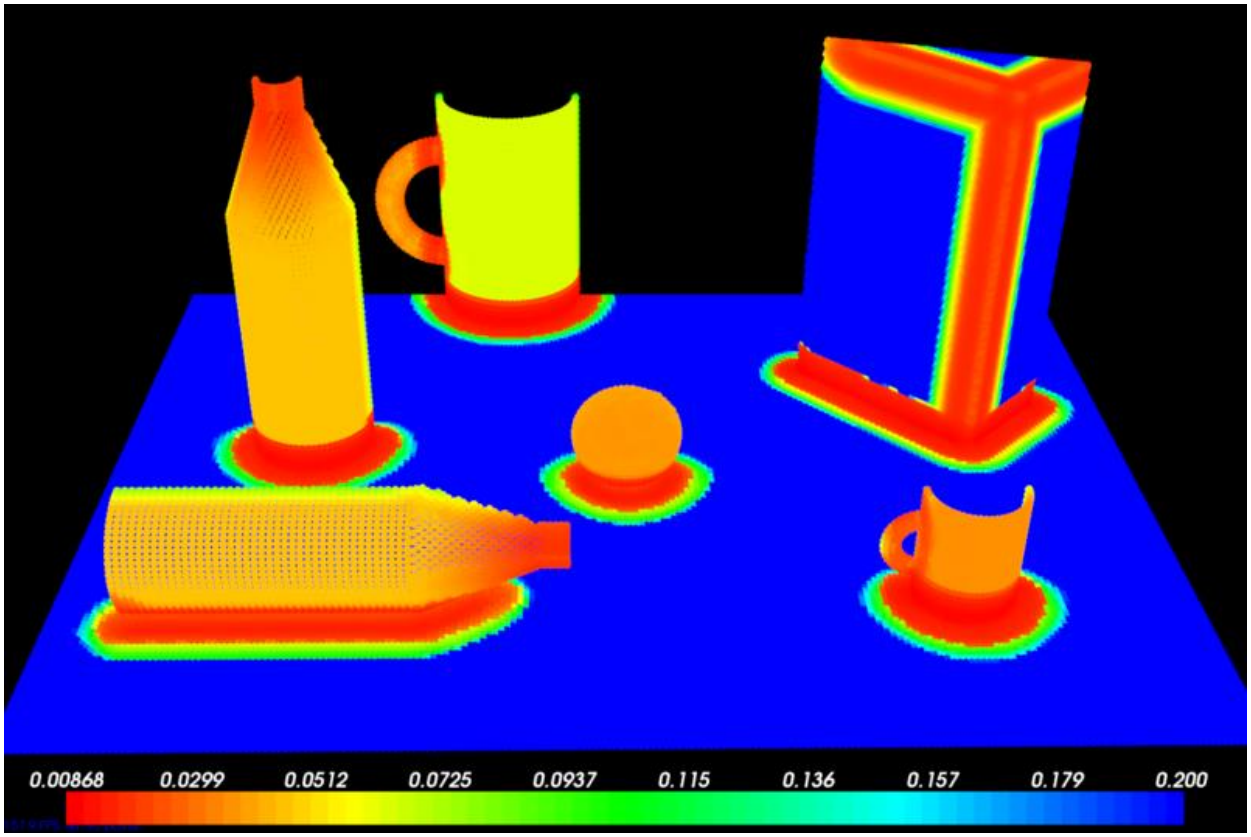


Sl. 2.2 Parovi točaka uspostavljeni prilikom računanja FPFH za točke. Slika preuzeta iz [1].

2.1.3 RSD(The Radius-Based Surface Deskriptor)

RSD kodira kutni odnos između točaka i njihovih susjeda. RSD računa udaljenost između svakog para, i udaljenost od normale. Zatim pretpostavlja da su obje točke na površini sfere. Konačno sa svim susjednim točkama na sferi uzima one koje imaju maksimalnu i minimalnu vrijednost radijusa i sprema ih u deskriptor točaka. Ulazne varijable su točke, normale, metode pretraživanja

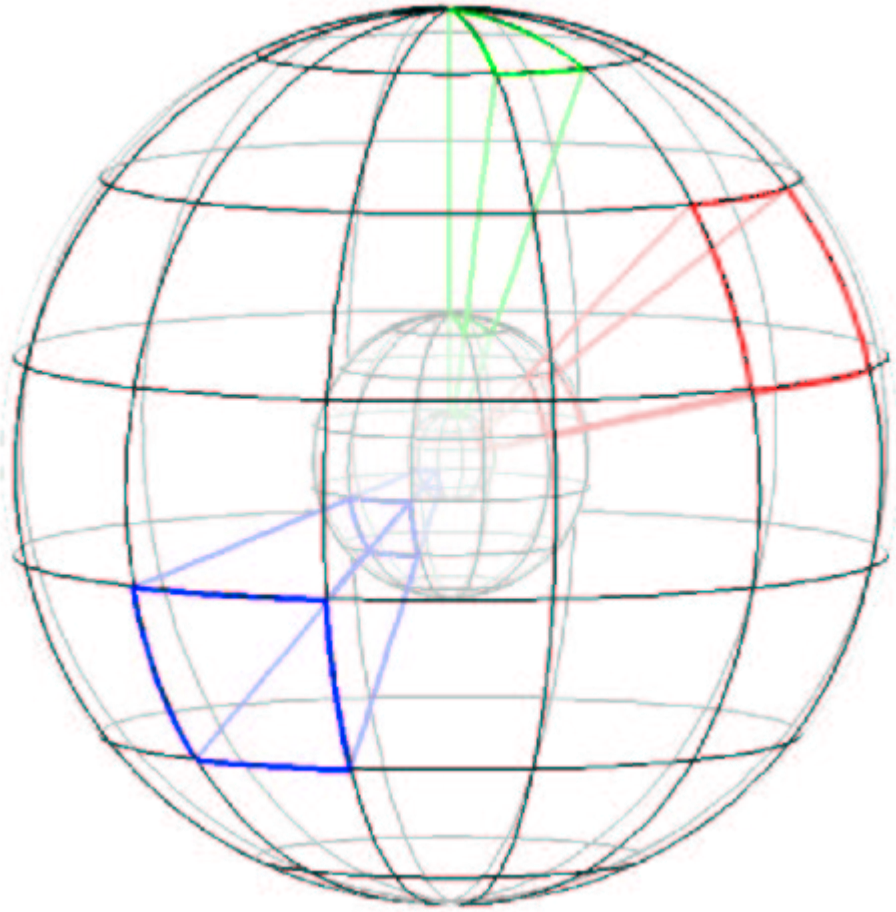
susjeda, radijus unutar kojega se traže susjedi, najveći radijus koji se smatra ravnom plohom, dok je izlazna RSD deskriptor [1].



Sl. 2.3 RSD vrijednosti oblaka, točke na ravnoj površini imaju maksimalnu vrijednost. Slika preuzeta iz [1].

2.1.4 3DSC(The 3D Shape Context)

3DSC je deskriptor koji je 3D verzija prethodno postojećeg 2D deskriptora. Radi tako da stvara sferu sa središtem u točki za koju računamo deskriptor, s danim radijusom pretraživanja. „Sjeverni pol“ sfere odgovara normali u središnjoj točki. Nakon toga se sfera dijeli na područja i binove. U prve dvije koordinate (azimut i elevacija) regije su jednako raspoređene, dok u trećoj koordinati(radijus) regije su logaritamski raspoređene tako da su manje prema sredini. Minimalni radijus se može zadati da se spriječi nastajanje malih binova, ali to bi metodu učinilo osjetljivom na male promjene na površini. Ulazne varijable su točke, normale, metode pretraživanja susjeda, radijus unutar kojeg se traže susjedi, minimalni radijus za traženje sfere, broj točaka u radijusu. Izlazna varijabla je 3DSC deskriptor koji se sastoji od broja točaka koje zadovoljavaju uvjet težine, a težina se računa ovisno o volumenu regije i gustoći točaka u regiji [1].



Sl. 2.4 Struktura okoline za izračun 3DSC-a za točku. Slika preuzeta iz [1].

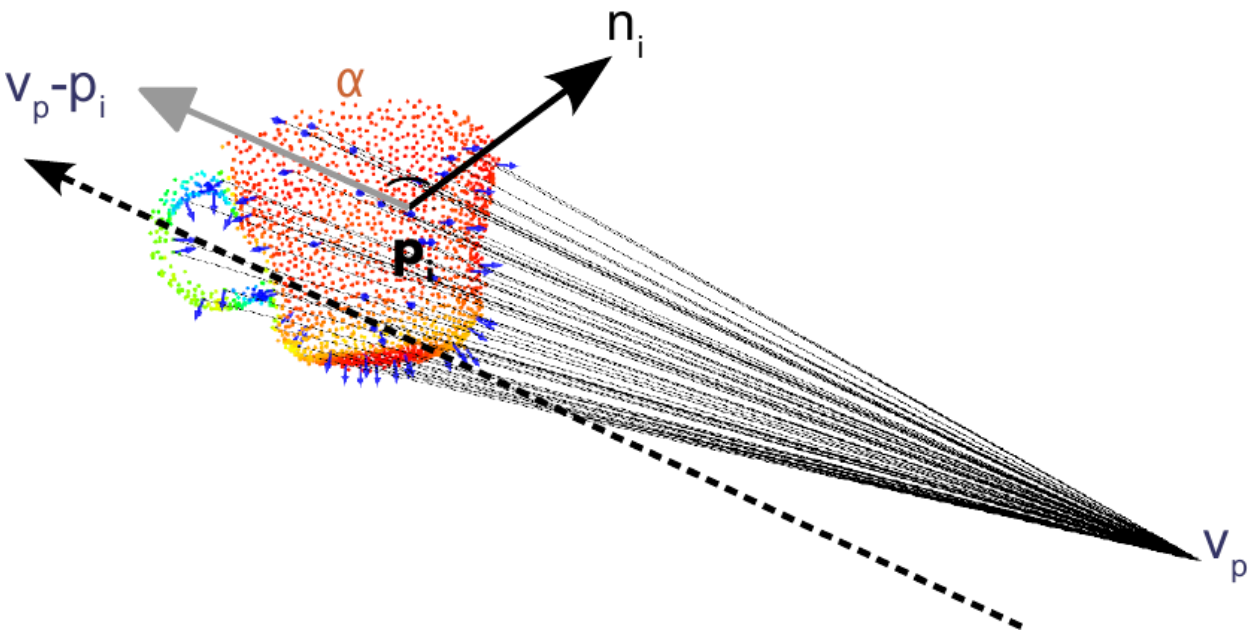
2.2 Globalni deskriptori

Globalni deskriptori uspoređuju točke puno brže od lokalnih. Razlog tome je taj što obično jedan deskriptor opisuje cijeli objekt, za razliku od lokalnih deskriptora, kod kojih je objekt opisan većim brojem deskriptora. S druge strane za njihovu je uporabu potrebno segmentirati sliku. To znači da se od objekata mora odvojiti podloga, te da objekti trebaju biti razdvojeni. Globalni deskriptori se koriste za prepoznavanje objekata i klasifikaciju, te geometrijsku analizu [1].

2.2.1 VFH(The Viewpoint Feature Histogram)

VFH je napravljen na bazi FPFH-a. Sastoji se od dva dijela; komponente smjera gledanja, i proširene FPFH komponente. Pri računanju prvog dijela, određuje se centroid objekta, što je točka koja je rezultat prosječne vrijednosti x,y i z koordinate svih točaka. Tada je vektor između smjera

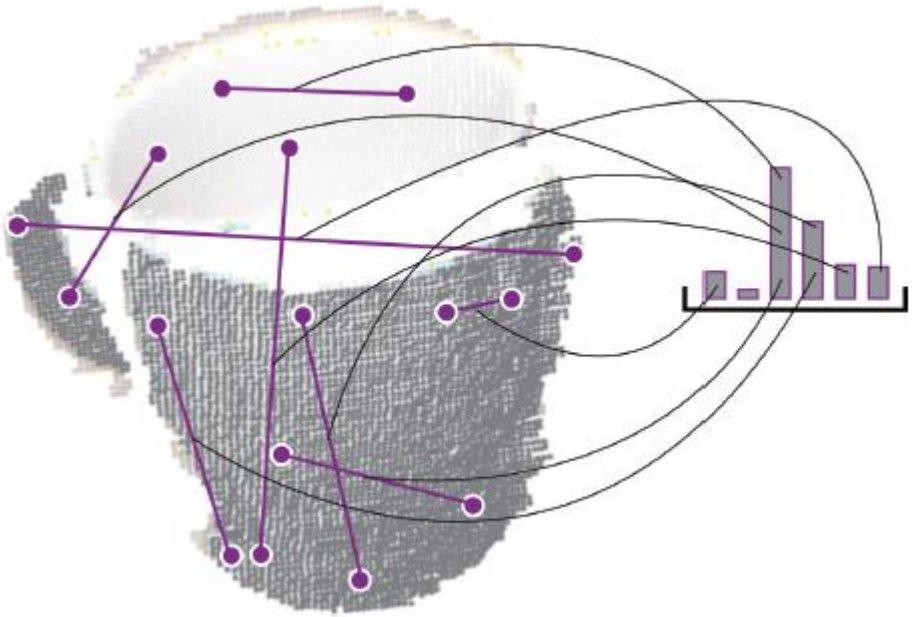
gledanja i centroida normaliziran. Konačno, za sve točke u klasteru izračunava se kut između tog vektora i njihove normale, a rezultat je histogram. Vektor je translatiran na svaku točku prilikom računanja kuta zbog toga što se na taj način postiže neovisnost deskriptora o skali. Drugi dio se računa kao FPFH s malim razlikama. Samo se računa za centroid, koristeći izračunat vektor smjera gledanja kao normalu i postavlja sve točke klastera kao susjede.[1]



Sl. 2.5 Komponenta smjera gledanja VFH. Slika preuzeta iz [1]

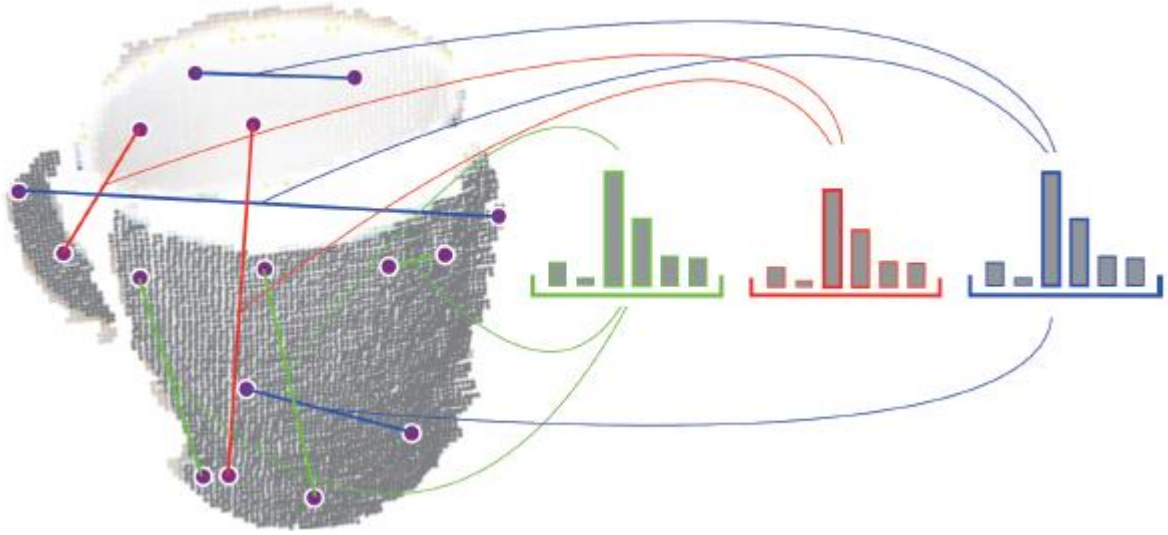
2.3 Skup funkcija oblika - ESF(Ensamble of Shape Function)

ESF deskriptor je skup od deset histograma od po 64 bita, od kojih svaki predstavlja jednu funkciju oblika. Funkcije oblika opisuju karakteristična svojstva klastera oblaka točaka. Od navedenih deset histograma, tri histograma predstavljaju kutove, tri predstavljaju udaljenosti, tri predstavljaju površine trokuta, dok jedan predstavlja omjer udaljenosti. D2 funkcija oblika napravljena je uzorkovanjem parova točaka iz oblaka točaka i stvaranjem histograma udaljenosti između tih parova točaka[3].



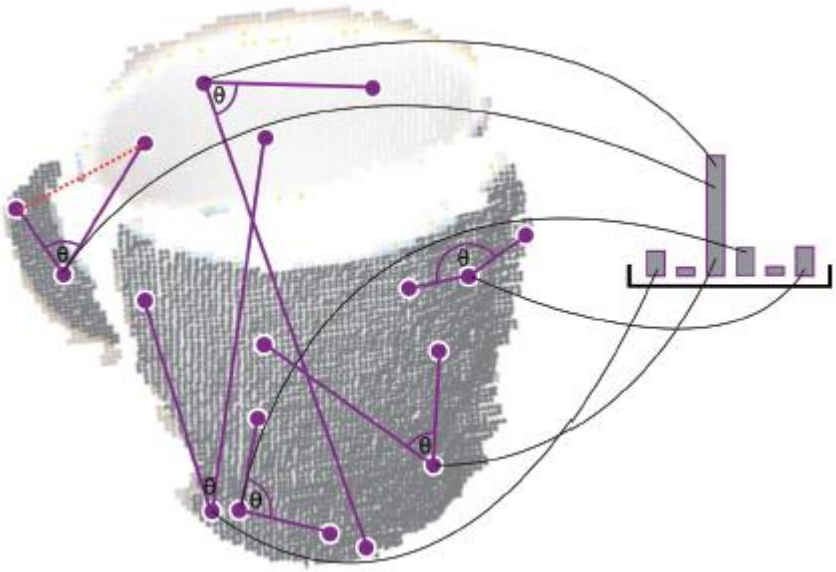
Sl. 2.6 Udaljenost između nasumično određenih točaka s jednog dijela objekta, s odgovarajućim dužinom koje stvaraju odgovarajući histogram koji formira D2 deskriptor. Slika preuzeta iz [2].

D2 funkvija oblika može ugrubo razlikovati oblike, ali jednom kada se broj klasa poveća ili ako je samo djelomični prikaz objekta dostupan, svojstvo prepoznavanja nije dovoljno. Ovaj osnovni oblik D2 funkcija proširen je tako da se za svaku liniju sačinjenu od dvije nasumično odabrane točke doda informacija je li ona izvan 3D modela, unutar ili mješavina oboje[4]. To omogućava da se metoda može primijeniti na parcijalne oblake točaka. Linije između dvije nasumično uzorkovane točke su klasificirane da budu na ili da ne budu na površini oblaka točaka. To se postiže praćenjem linije 3D Bresenham-ovim algoritmom u volumnoj rešetki s bočnom dužinom od 64 volumnih kockica (engl. *vovel*) . Takva krupna volumna rešetka služi za aproksimaciju stvarnih površina. Nakon traženja zauzetih volumnih kockica sastavljaju se tri različita histograma koja predstavljaju točke izvan 3D modela, unutar ili mješavanu oboje[2].



Sl. 2.7 Klasificirane udaljenosti. Slika preuzeta iz [2].

Na slici 2.7. zelena boja predstavlja spojene linije koje leže na površini objekta (u daljnjem tekstu ON), crvena boja predstavlja linije na kojima su samo točke završetka na površini i linija koja ih spaja nije na površini (u daljnjem tekstu OFF), dok plava boja predstavlja udaljenost linija koje su klasificirane kao mješavina(u daljnjem tekstu MIXED) jer su djelomice na a djelomice nisu na površini.



Sl. 2.8 Funkcija oblika A3 se računa uzimanjem kuta zatvaranja između dvije linije stvorene histogramom oblika. Slika preuzeta iz [2].

A3 funkcija oblika kodira kut koji zatvaraju dvije linije stvorene nasumičnim uzorkovanjem tri točke oblaka točaka. Slika 2.8. prikazuje konstruiranje A3 funkcije oblika. Kao i kod D2 funkcije oblika, A3 funkcija daje bolji rezultat ako se odvoje kutevi histograma u tri različita histograma koji predstavljaju ON, OFF i MIXED kuteve. Posljednja tri histograma su sastavljena od D3 funkcije oblika, što je D2 funkcija oblika s jednom dimenzijom više. Naime, of tri točke formira se trokut, a D3 funkcija predstavlja histogram površina takvih trokuta. Ista metodologija klasificiranja područja ON, OFF i MIXED je primijenjena i za računanje D3 funkcije oblika[2].

Klasificiranje objekata, kao cilj ovog završnog rada, se ovom metodom postiže na način da se histogram, koji se dobije primjenom deskriptora na novu sliku koju želimo klasificirati, usporedi s prethodno spremljenim histogramima. Ti spremljeni histogrami su generirani za više pogleda na svaki od modela. Taj se proces odvija na način da se za svaki model generira 42 oblaka točaka, od kojih svaki predstavlja 3D snimku modela iz različitog kuta. Za svaki od tih oblaka točaka stvara se deskriptor računanjem funkcija D2, A3 i D3.

3. PROGRAMSKA REALIZACIJA

Program za klasificiranje objekata napisan je u C++ programskom jeziku. Za njegovu realizaciju koristi se PCL biblioteka u kojoj je sadržana klasa koja omogućava izračunavanje ESF deskriptora. Program se može podijeliti u nekoliko dijelova. Prvi dio programa je učenje, odnosno spremanje deskriptora svih pogleda na modele. U drugom dijelu se unosi RGB-D slika, koja se u trećem dijelu uspoređuje sa svim deskriptorima do sada „naučenim“. I konačno, zadnji dio programa ispisuje o kojem je objektu riječ na slici.

3.1 PCL (The Point Cloud Library) biblioteka

PCL je veliki otvoreni projekt za obradu oblaka 3D točaka. PCL okruženje sadrži brojne algoritme koji uključuju filtriranje, određivanje značajki, detektiranje površine, registraciju, segmentiranje. PCL je besplatan za komercijalne i istraživačke potrebe. PCL je dostupan na gotovo svim platformama uključujući Linux, MacOS, Windows, Android i iOS[5].

3.2 Učenje modela

Prvi dio programa je učenje modela, drugim riječima stvaranje deskriptora na svakom od 42 pogleda jednog modela, te njihovo spremanje u datoteku . Svi nazivi modela su spremljeni u jednu .txt datoteku zbog jednostavnosti i efikasnosti kreiranja pogleda na svaki model. Ta datoteka predstavlja ulaznu varijablu programa, jer su u njoj spremljeni 3D modeli predmeta iz svih klasa objekata koje program treba prepoznati. Navedeni 3D modeli su .ply formata. Za svaki model generira se 42 pogleda jednako razmaknutih od središnje točke i usmjerenih prema njoj, ali pod različitim kutevima snimanja jednoliko raspoređenih tako da pokrivaju sve strane objekta. Svaki pogled predstavlja oblak točaka koji se funkcijom savePCDFileASCII sprema u .pcd datoteku u obliku niza od po tri koordinate, x, y, z, za svaku točku. Nakon spremanja u datoteku, otvaramo odgovaranjuću .pcd datoteku, te pokrećemo funkciju za izračunavanje ESF deskriptora. Izračunati

deskriptori se spremaje u novu datoteku. Taj se proces ponavlja sve dok se nisu generirali deskriptori za sve poglede svih modela.

```
45          vtkSmartPointer<vtkPolyData> polydata;
46          vtkSmartPointer<vtkPLYReader> reader = vtkSmartPointer<vtkPLYReader>::New();
47
48          printf("%s\n", filepath);
49          strcpy(c, filepath.c_str());
50          reader->SetFileName(c);
51          reader->Update();
52          polydata = reader->GetOutput();
53
54          float resx = 64;
55          float resy = resx;
56
57          vector<PointCloud<PointXYZ>, Eigen::aligned_allocator<PointCloud<PointXYZ> > >  views;
58          vector<Eigen::Matrix4f, Eigen::aligned_allocator<Eigen::Matrix4f> > poses;
59          vector<float> entropies;
60
61          visualization::PCLVisualizer vis;
62          vis.addModelFromPolyData(polydata, "mesh1", 0);
63          vis.setRepresentationToSurfaceForAllActors();
64          vis.renderViewTesselatedSphere(resx, resy, views, poses, entropies, 1);
```

Sl. 3.1 Stvaranje pogleda na model.

Na slici 3.1 može se vidjeti kreiranje čitača(reader) (46-ta linija koda) i postavljanje vrijednosti za funkciju ViewTesselatedSphere (54-59-ta linija koda), kojoj je potrebna rezolucija odnosno duljina i visina slike, te pogledi i pozicije. Funkcija ViewTesselatedSpheres pripada klasi PCL Visualizer, i traži kao ulazne varijable rezolucije prozora u kojem se prikazuje model, zatim vektor oblka točaka u koji će se spremiti točke nakon pogleda, sljedeći je vektor u koji se spremaju pozicije pogleda, nakon toga vektor entropija, broj pogleda na model. Dodatno je moguće definirati kut pod kojim se gleda na središnju točku objekta, te radijus sfere. Funkcija vraća 42 pogleda jednako razmaknutih od središnje točke i usmjerenih prema njoj.

Nadalje slijedi kreiranje oblaka točaka, postavljanje širine i visine oblaka točaka, te pridruživanje koordinata x,y,z s pogleda u oblak, što je izvedeno programskim kodom prikazanim na slici 3.2. Zatim slijedi spremanje oblaka u datoteku, učitavanje tog oblaka iz datoteke radi izvršavanja deskriptora, te njegovo izračunavanje korištenjem funkcije compute, kao što je prikazano na slici 3.3. Zadnji korak prvog dijela je prolazak kroz petlju i spremanje vrijednosti histograma u binarnu datoteku, kao što prikazuje slika 3.3.

```
70          PointCloud<PointXYZ> cloud;
71          cloud.width = views.size();
72
73          cloud.is_dense = false;
74          ESFEstimation<PointXYZ, ESFSignature640> esf;
75
76
77          for (int i = 0; i < views.size(); i++)
78          {
79              cloud.height = views[i].size();
80              cloud.width = 1;
81              cloud.points.resize(cloud.width * cloud.height);
82
83              for (int j = 0; j < views[i].size(); j++)
84              {
85                  cloud.points[j].x = views[i][j].x;
86                  cloud.points[j].y = views[i][j].y;
87                  cloud.points[j].z = views[i][j].z;
```

Sl. 3.2 Spremanje koordinata u oblak.

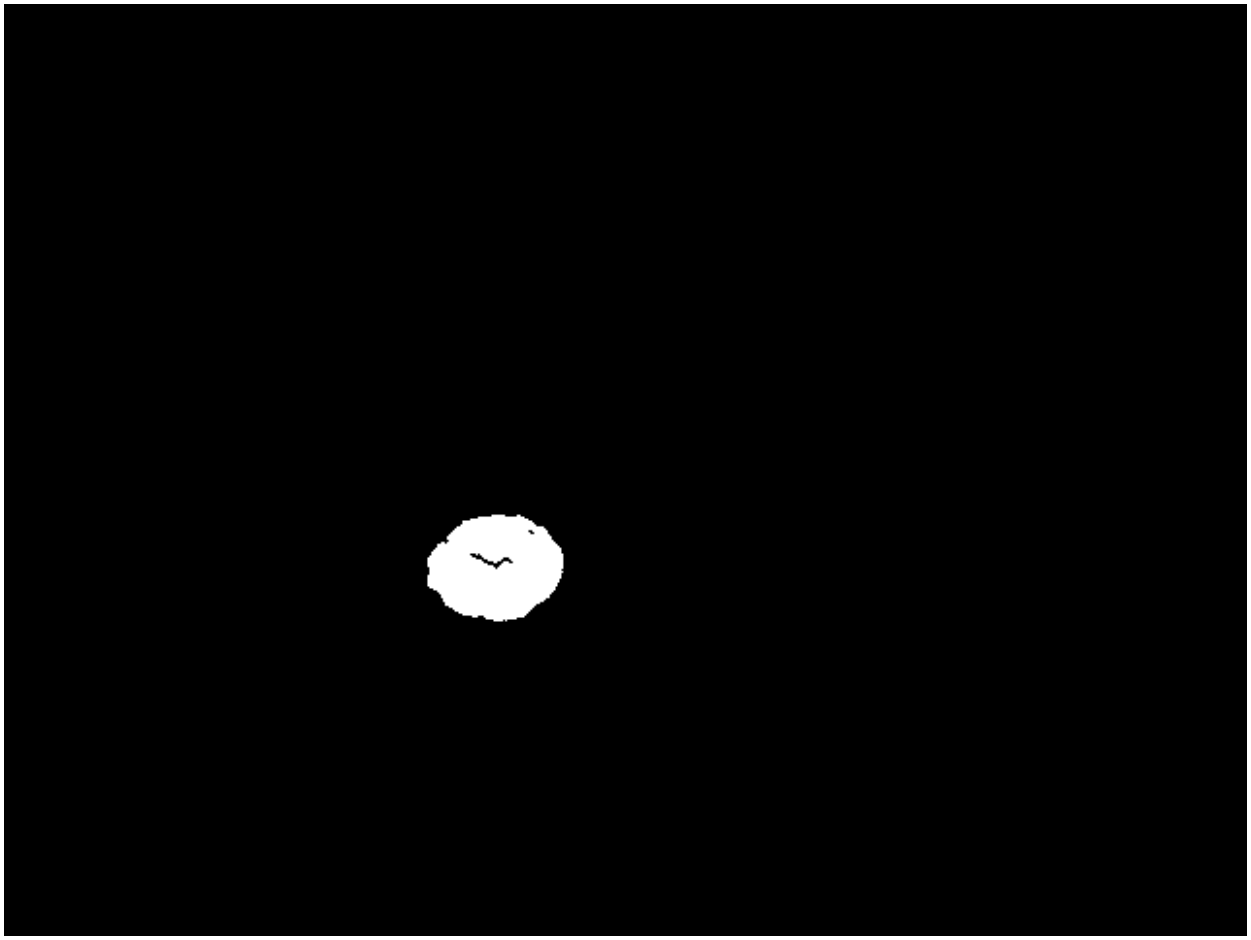
```
90          savePCDFileASCII("test_pcd.pcd", cloud);
91
92          if (loadPCDFile<PointXYZ>("test_pcd.pcd", *object) != 0)
93          {
94              return -1;
95          }
96
97          esf.setInputCloud(object);
98
99          esf.compute(*descriptor);
100
101
102         FILE *fpDescriptor = fopen("descriptor.txt", "ab");
103         for (int k = 0; k < descriptor->size(); k++)
104         {
105             for (int l = 0; l < descriptor->at(k).descriptorSize(); l++)
106             {
107                 double h = descriptor->at(k).histogram[l];
108                 fwrite(&h, 1, sizeof(h), fpDescriptor);
109             }
110         }
111         fclose(fpDescriptor);
```

Sl. 3.3 Kreiranje deskriptora.

3.3 Unos slike

Idući dio programa se odnosi na učitavanje segmentirane slike objekta, te njegovo skaliranje na jediničnu kružnicu, izračunavanje deskriptora i spremanje u datoteku.

Prvi dio odnosi se na učitavanje segmentirane slike koja je prikazana na slici 3.4. Na slici se mogu vidjeti crni i bijeli pikseli koji predstavljaju pozadinu (crni pikseli) i objekt (bijeli pikseli). U ovom programu je cilj prepoznati objekt na slici stoga se na segmentiranoj slici promatraju samo bijeli pikseli te se prebrojavaju kako bi se znala postaviti veličina oblaka u koje će se ti bijeli pikseli spremiti. Na slici 3.5. je prikazano učitavanje datoteka koje sadržavaju nazive i lokacije segmentiranih slika, dok slika 3.6. prikazuje spremanje cijelih točkica sa segmentirane slike u oblak točaka.



Sl. 3.4 Prikaz segmentirane slike krofne.

```
137     char c[300];
138
139     ifstream infile("C://Users/hvarg/Desktop/scene.txt");
140     ifstream infile_png("C://Users/hvarg/Desktop/scene_png.txt");
141     string file_path, file_path_png;
142     FILE *Descriptor_scene_2 = fopen("descriptor_scene_bottle.txt", "wb");
143     fclose(Descriptor_scene_2);
144
145     while (std::getline(infile, file_path) && std::getline(infile_png, file_path_png))
146     {
147         istringstream iss_png(file_path_png);
148         string filepath_png;
149         iss_png >> filepath_png;
150         if (filepath_png.compare("end") == 0)
151             break;
152
153         Mat image;
154         image = imread(filepath_png);
155         cvtColor(image, image, COLOR_BGR2GRAY);
156
157         istringstream iss(file_path);
158         string filepath;
159         iss >> filepath;
160         if (filepath.compare("end") == 0)
161             break;
```

Sl. 3.5 Učitavanje segmentirane slike.

```
163         PointCloud<PointXYZ> oblak;
164         oblak.is_dense = false;
165         PointCloud<PointXYZ>::Ptr objekt(new PointCloud<PointXYZ>);
166         strcpy(c, filepath.c_str());
167         loadPCDFile<PointXYZ>(c, *objekt);
168         int br255 = 0, br = 0;
169         for (int rows = 0; rows < image.rows; rows++)
170             for (int cols = 0; cols < image.cols; cols++)
171                 if (image.at<uchar>(rows, cols) == 255)
172                 {
173                     br255++;
174                 }
175         oblak.width = br255;
176         oblak.height = 1;
177         oblak.points.resize(br255);
178         for (int rows = 0; rows < image.rows; rows++)
179             for (int cols = 0; cols < image.cols; cols++)
180                 if (image.at<uchar>(rows, cols) == 255)
181                 {
182                     oblak.points[br].x = objekt->at(rows*image.cols + cols).x;
183                     oblak.points[br].y = objekt->at(rows*image.cols + cols).y;
184                     oblak.points[br].z = objekt->at(rows*image.cols + cols).z;
185                     br++;
186                 }
```

Sl. 3.6 Pridruživanje točaka u oblak.

Na slici 3.7 je prikazano skaliranje oblaka točaka na jediničnu kružnicu. To se postiže određivanjem centralne točke u oblaku točaka zvane centroid. Centroid se izračunava funkcijom compute3DCentroid. Nakon toga se računa udaljenost svake točke oblaka točaka od centroida, i traži se najveća udaljenost. Ta najveća udaljenost je potrebna kako bi cijeli oblak točaka mogli podijeliti s najvećom udaljenošću i tako skalirati oblak točaka da stane unutar jedinične kružnice. Odgovarajući kod je prikazan na slici 3.7.

```
188         long double najveci = 0;
189         Eigen::Vector4f centroid;
190         compute3DCentroid(oblak, centroid);
191         for (int broj = 0; broj < oblak.points.size(); broj++)
192         {
193             long double udaljenost = sqrt(pow(oblak.points[broj].x - centroid(0), 2.0) +
194                 pow(oblak.points[broj].y - centroid(1), 2.0) + pow(oblak.points[broj].z - centroid(2), 2.0));
195             if (udaljenost >= najveci)
196                 najveci = udaljenost;
197         }
198
199         int br1 = 0;
200         for (int br1 = 0; br1 < oblak.points.size(); br1++)
201         {
202             oblak.points[br1].x = (oblak.points[br1].x - centroid(0)) / najveci;
203             oblak.points[br1].y = (oblak.points[br1].y - centroid(1)) / najveci;
204             oblak.points[br1].z = (oblak.points[br1].z - centroid(2)) / najveci;
205         }
206
```

Sl. 3.7 Skaliranje oblaka točaka.

Završni dio drugog dijela programa je spremanje oblaka točaka u .pcd oblik i izračunavanje ESF deskriptora.

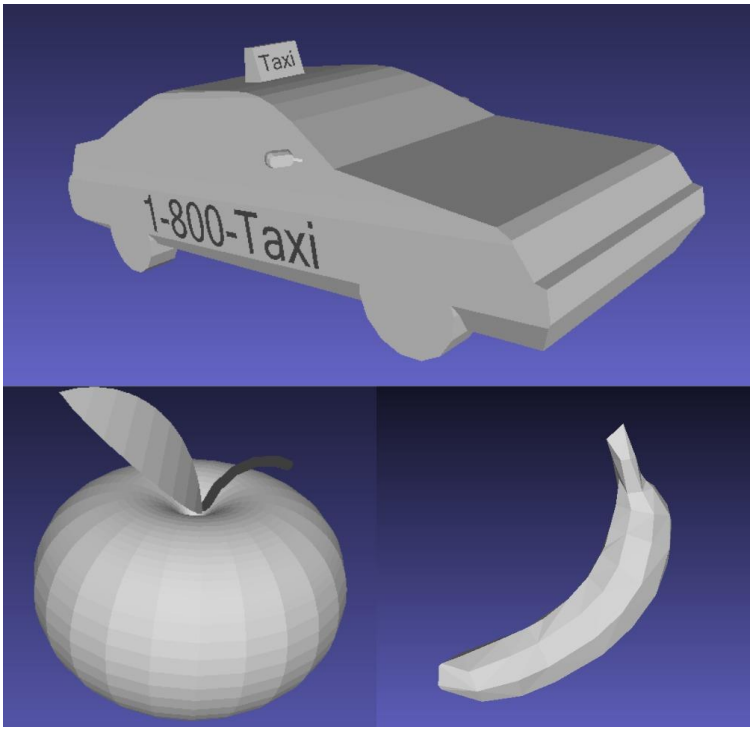
3.4 Uspoređivanje scene i modela

Konačni dio programa je uspoređivanje trenutno promatrane scene i postojećih modela. Ovaj dio programa se odvija u Matlab-u na načina da se prvo učita datoteka koja sadrži sve modele, zatim datoteka koja sadrži deskriptor slike koju želimo identificirati. Nakon toga se odvija dio u kojem se računa najmanja apsolutna razlika između modela i scene. Najmanja apsolutna razlika predstavlja najmanju razliku deskriptora između promatrane scene i modela. Razlika dvaju deskritora se računa kao suma apsolutnih vrijednosti razlika komponenti tih deskriptora. U ovom zadatku su korištena i uspoređena dva načina za određivanje klase promatrane scene, to su određivanje pomoću najbližeg susjeda i određivanje pomoću deset najbližih susjeda. Prvi način računa najmanju razliku između deskriptora promatrane scene i svih modela i za određivanje klase koristi samo jedan deskriptor modela koji ima najmanju razliku. Objekt se pridružuje klasi kojoj pripada najbliži model(najbliži susjed). Drugi način za klasifikaciju pronalazi deset najmanjih

razlika između promatrane scene i modela i kao odgovarajuću klasu uzima u obzir onu klasu koja je najzastupljenija u tih deset najbližih susjeda.

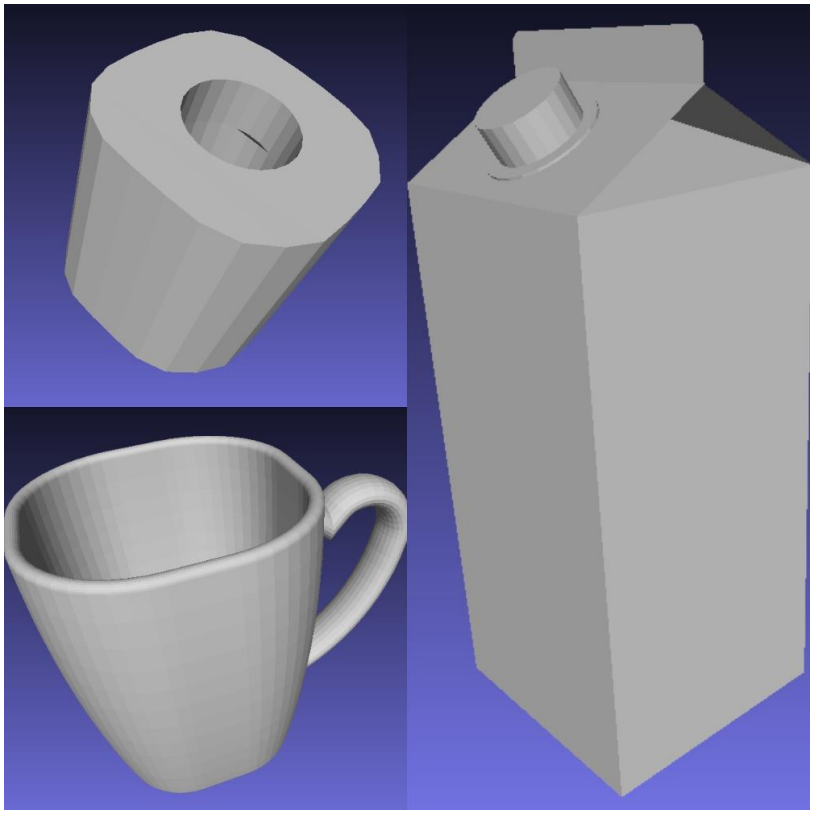
4. EVALUACIJA PROGRAMA NA 3DNET ISPITNOM PODATKOVNOM SKUPU

Evaluacijom programa se ispituje točnost programa za klasifikaciju objekta na RGB-D slikama. Evaluacija je provedena na skupu RGB-D slika raznih modela, koje su segmentirane radi uporabe deskriptora. Podatkovni skup se sastoji od skupa za učenje, koji predstavlja skup 3D modela objekata, te ispitnog dijela koji se sastoji od RGB-D slika objekata postavljenih na ravnu površinu. Objekti su svrstani u 10 klasa, a to su klasa jabuka, klasa banana, klasa auto, prikazane na slici 4.1, klasa boca, klasa krofna, klasa čekić, prikazane na slici 4.2, klasa šalica, klasa tetrapak, klasa toaletni papir, prikazane na slici 4.3 i klasa zdjele prikazane na slici 4.4. Ukupan broj modela svih klasa je 351, dok ispitnih slika ima 1641[6]. Postupak obrade jedne slike iz ispitne skupine objasnit ću na primjeru čekića. Sa slika je odvojena pozadina i površina. Slika 4.9 prikazuje RGB-D sliku čekića, zatim slika 4.10 prikazuje segmentirani oblik slike 4.9. To se postiže odvajanjem pozadine i ravne površine sa slike te grupiranje povezanih skupova točaka unutar znaprijed definiranog prostora ispred kamere.

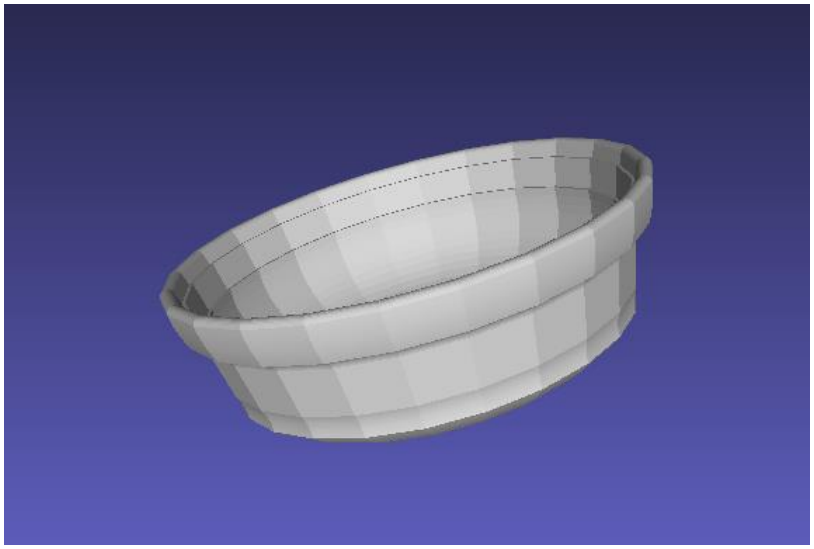


Sl. 4.1 Model jabuke, banane i auta.

Sl. 4.2 Model krafne, boce i čekića.



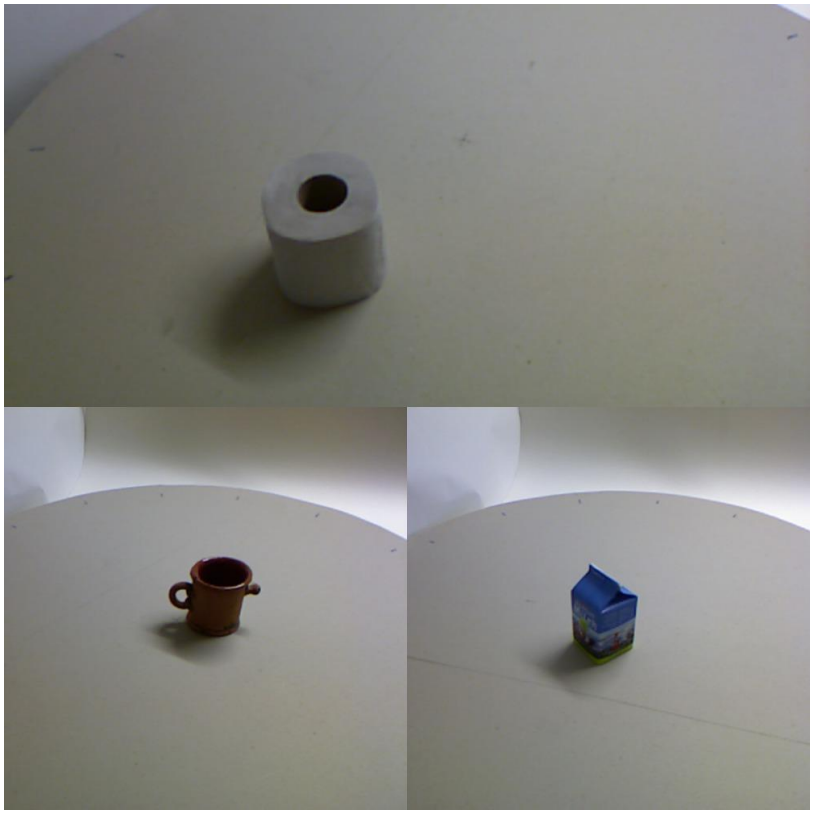
Sl. 4.3 Model šalice, tetrapaka i toaletnog papira.



Sl. 4.4 Model zdjele.

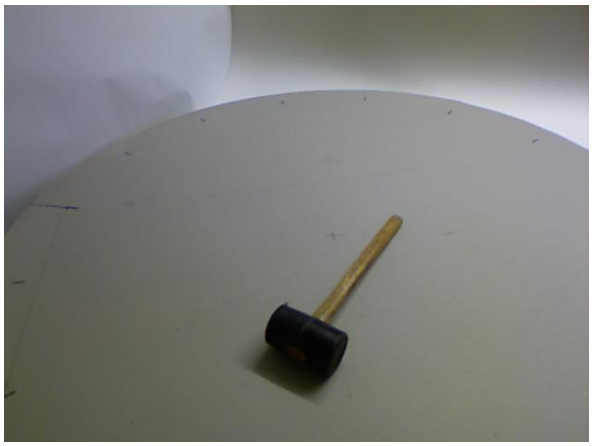
Sl. 4.5 Testna slika auta, banane i jabuke

Sl. 4.6 Testna slika boce, čekića i krofne.

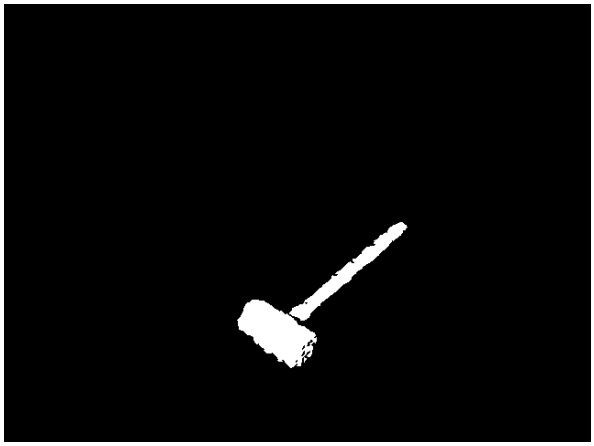


Sl. 4.7 Testna slika toaletnog papira, šalice i tetrapaka.

Sl. 4.8 Testna slika zdjele.

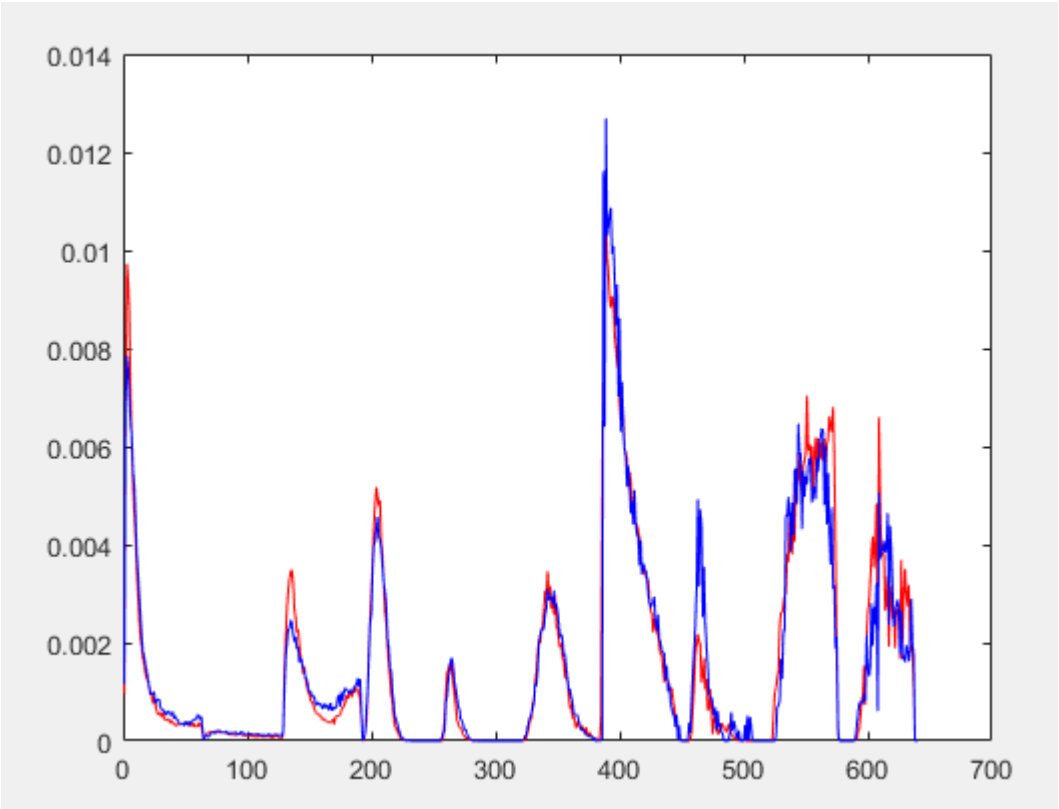


Sl. 4.9 RGB-D slika čekića. Slika preuzeta iz [6].

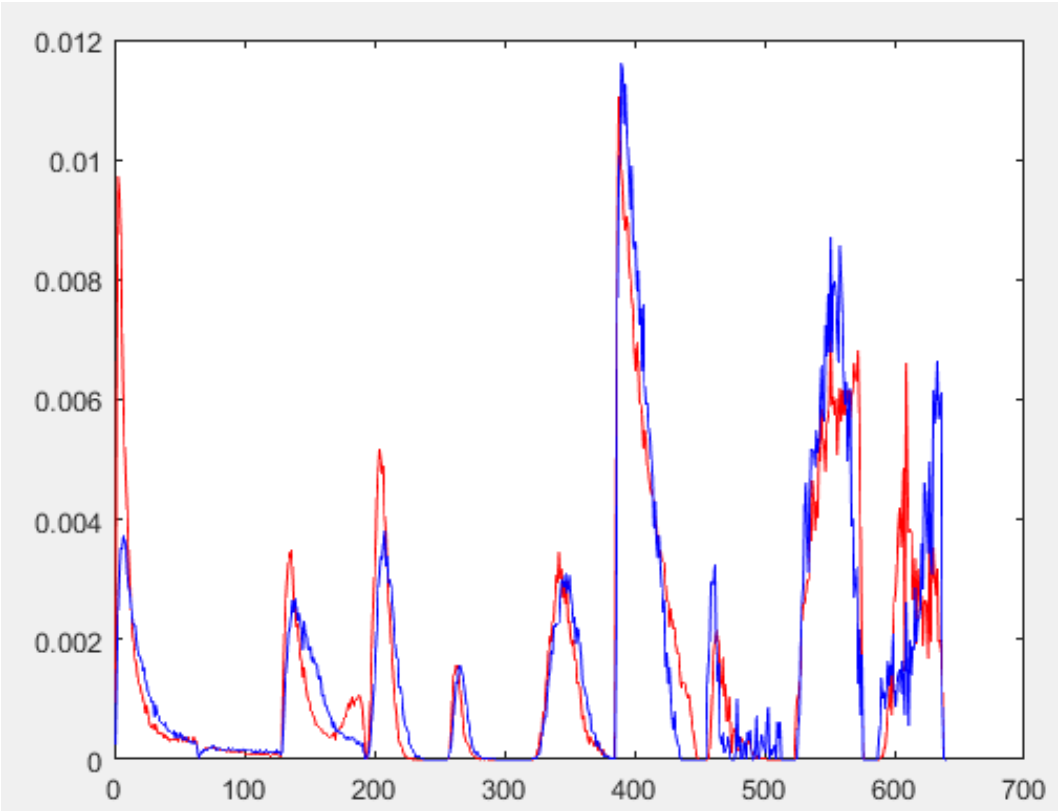


Sl. 4.10 Segmentirana slika čekića.

Nakon izračunavanja ESF deskriptora na segmentiranoj slici čekića, program uspoređuje trenutan ESF deskriptor čekića s prijašnje naučenim ESF deskriptorima raznih modela i metodom apsolutne razlike traži ESF deskriptor koji mu je najsličniji. U konačnici, slika 4.11 prikazuje deskriptor scene i najsličniji ESF deskriptor korištenjem metode apsolutne razlike između promatrane scene i modela. Deskriptor scene je prikazan crvenom bojom i ima ih 640, dok je najsličniji ESF deskriptor iz datoteke modela prikazan plavom bojom kojih također ima 640. Na slici 4.12 se može vidjeti usporedba deskriptora banane obojano u plavo s deskriptorom čekića obojanog u crveno. Također slika prikazuje najbolji mogući deskriptor banane koji odgovara deskriptoru čekića.



Sl. 4.11 Usporedba dva ESF deskriptora primjenom prve metode.



Sl. 4.12 Usporedba dva ESF deskriptora banane i čekića

4.1 Usporedba metoda klasificiranje

Usporedbom prve i druge metode na svim scena dostupnim na 3D-net.org nastaju tablica 4.1 i tablica 4.2 koje sadržavaju klase koje su detektirane, te postotak prepoznavanja određenog objekta. Za svaku tablicu je izračunat prosjek detektiranja točnih objekata s kojim se može usporediti koja je metoda preciznija. U tablici 4.1 u retku u kojem su prikazane banane, može se vidjeti kako je program detektirao 64 banane, 2 boce, 6 auta i jedan čekić od ukupno 73 banane na ispitnim slikama , što znači da se točnost detektiranja banane dobija dijeljenjem detektiranih banana s ukupnim brojem banana provedenih kroz program. Isto tako u tablici 4.2 na istom primjeru banana se može vidjeti kako je u manjem broju program prepoznao banane.

Tab. 4.1 Rezultati dobiveni prvom metodom.

Best Model	classApple	classBanana	classBottle	classBowl	classCar	classDonut	classHammer	classMug	classTetraPak	classToiletPaper	SUM	%
classApple	7	0	0	93	0	27	0	1	0	0	128	5,47%
classBanana	0	64	2	0	6	0	1	0	0	0	73	87,67%
classBottle	0	1	258	0	59	1	2	3	0	0	324	79,63%
classBowl	0	0	0	38	0	12	0	18	0	0	68	55,88%
classCar	0	1	19	3	115	3	0	8	1	4	154	74,68%
classDonut	0	0	0	0	0	48	0	2	0	0	50	96,00%
classHammer	0	0	1	0	0	0	176	0	0	0	177	99,44%
classMug	4	0	0	30	2	6	0	325	1	4	372	87,37%
classTetraPak	0	0	12	11	20	0	0	25	103	1	172	59,88%
classToiletPaper	1	0	0	51	1	30	0	9	0	2	94	2,13%
												64,81%

Tab. 4.2 Rezultati dobiveni drugom metodom.

Best Class	classApple	classBanana	classBottle	classBowl	classCar	classDonut	classHammer	classMug	classTetraPak	classToiletPaper	SUM	%
classApple	3	0	0	86	0	39	0	0	0	0	128	2,34%
classBanana	0	62	3	0	8	0	0	0	0	0	73	84,93%
classBottle	0	0	272	0	48	1	0	2	1	0	324	83,95%
classBowl	0	0	0	42	0	14	0	12	0	0	68	61,76%
classCar	0	0	29	0	115	3	0	7	0	0	154	74,68%
classDonut	0	0	0	0	0	50	0	0	0	0	50	100,00%
classHammer	0	0	4	0	0	0	173	0	0	0	177	97,74%
classMug	1	0	0	28	3	10	0	329	0	1	372	88,44%
classTetraPak	0	0	25	16	14	0	0	39	76	2	172	44,19%
classToiletPaper	0	0	0	4	0	78	0	11	0	1	94	1,06%
												63,91%

5. ZAKLJUČAK

U ovom završnom radu je izrađen, predstavljen i testiran program za klasifikaciju objekata na RGB-D slikama pomoću ESF deskriptora. ESF deskriptor se temelji na tri funkcije oblika: D2, A3, D3, od kojih svaka na svoj način opisuje dani oblak točaka. Klasifikacija primjenom tog deskriptora se izvodi na način da se izračunava apsolutna razlika između promatrane scene i svih modela, te se objekt klasificira odabirom onog modela koji najbolje zadovoljava izračunati uvjet. Pri klasificiranju objekta potrebno je prvo naučiti program na određene modele s kojima je u daljnjem programu moguća usporedba. Potom je objašnjen tijek programa, te sve ključne funkcije i metode prilikom izvođenja, kao i biblioteka u kojoj je sve to implementirano. Izrađeni program je ispitan na ispitnom podatkovnom skupu 3DNet, koji se sastoji od skupa za učenje i ispitnog skupa. Skup za učenje je skup modela podijeljenjih u 10 klasa, dok ispitni skup sadrži RGB-D slike. U nekim slučajevima bolja je metoda najmanje apsolutne razlike između promatrane scene i modela, a u nekim metoda deset najbližih. U konačnici, metoda deset najbližih susjeda nije pokazala bolje rezultate od metode najbližeg susjeda, što je u suprotnosti s rezultatima dobivenim u [2]. Jedan mogući razlog je što je u [2] korišteno više pogleda za svaki model.

LITERATURA

[1]http://robotica.unileon.es/index.php/PCL/OpenNI_tutorial_4:_3D_object_recognition_(descriptors), PCL/OpenNI tutorial 4:3D object recognition (descriptors), 12.04.2018

[2]Walter Wohlkinger and Markus Vincze, Ensemble of Shape Function for 3D Object Classification In *Proc. IEEE Int. Conf. on Robotics and Biomimetics (ROBIO)*, 2011, pp. 2987–2992.

[3]R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin. Matching 3d models with shape distributions. In *Shape Modeling and Applications,SMI 2001 International Conference on.*, pages 154 –166, May 2001.

[4]Cheuk Yiu Ip, Daniel Lapadat, Leonard Sieger, and William C. Regli.Using shape distributions to compare solid models. In *Proceedings of the seventh ACM symposium on Solid modeling and applications*, SMA '02, pages 273–280, New York, NY, USA, 2002. ACM.

[5]http://pointclouds.org/about/, pcl, 12.04.2018

[6]W. Wohlkinger, A. Aldoma, R. B. Rusu, and M. Vincze. "3DNet: Largescale object class recognition from cad models." In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2012, pp. 5384–5391.

SAŽETAK

Klasifikacija objekata prikazanim na 2D ili 3D slikama postaje sve više i više popularna i tehnilogije koje se razvijaju su efikasnije i jednostavnije, dok kvaliteta i točnost ostaju visoki. Metoda korištena u ovom radu je ESF(Ensemble of shape functions). Ona je jedna od metoda uključena u PCL biblioteku. ESF je funkcija oblika koja je jednostavna, a omogućuje puno načina za korištenje. Jedan od najčešćih načina korištenja je klasificiranje objekata, ali ima i ostalih kao što su primjećivanje, računanje raznoraznih udaljenosti, normala u geometrijskoj okolini. U radu je također opisan program za klasifikaciju objekata na RGB-D slikama primjenom ESF metode, te ispitan na ispitnom podatkovnom skupu 3DNet.

Ključne riječi: Deskriptor, ESF(Ensemble of shape functions), klasifikacija objekta, PCL(Point Cloud Library), računalni vid, RGB-D slike.

ABSTRACT

Object detection in RGB-D images using Ensemble of shape function

Object recognition for 2D or 3D images becomes more and more popular and technologies that are being developed are more efficiend and simpler, while quality and precision remain high. The method used in this project is ESF(Ensemble of shape functions). It is one of many methodes included in the PCL library. ESF is shape function that is simple and has various applications. This method is primarily designed for object recognition, but there are other applications like registration, calculating distances or calculating normals in geometric environment. In this project, a program for object classification using ESF method is described and it is tested on the 3DNet data set.

Key words: Descriptor, ESF(Ensemble of shape functions), object recognition, PCL(Point Cloud Library), machine vision, RGB-D pictures.

ŽIVOTOPIS

Hrvoje Varga rođen 29. siječnja 1997. godine u Osijeku. Pohađao osnovnu školu u Bilju, te nakon završetka osnovne škole pohađao je Prirodoslovno – matematičku gimnaziju u Osijeku. Nakon završetka srednje škole, 2015. godine upisuje Elektrotehnički fakultet u Osijeku, preddiplomski sveučilišni studij Računarstva.