Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Dejan Štepec

Razvoj sistema za prepoznavanje objektov na osnovi oblakov točk

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: doc. dr. Danijel Skočaj

Ljubljana 2015

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

S pojavom nizkocenovnih barvno-globinskih kamer je zajemanje 3-dimenzionalne oblike predmetov postalo zelo dostopno. Zato se poleg tradicionalnih metod za razpoznavanje predmetov, ki temeljijo na analizi slik, pojavljajo tudi pristopi, ki za razpoznavanje predmetov uporabljajo 3D informacijo. V diplomski nalogi implementirajte sistem, ki oblak 3D točk, dobljen z zajemom globinske slike predmeta, ustrezno obdela, zgradi ustrezno predstavitev predmeta ter nato na osnovi predhodno naučenih modelov predmetov izmed njih razpozna pravega. Analizirajte uspešnost različnih opisnikov, ki jih boste uporabili pri gradnji predstavitve predmetov. Na osnovi rezultatov implementirajte sistem, ki bo deloval v realnem času ter bo sposoben zajeti globinsko sliko predmeta in ga klasificirati na osnovi tako dobljenega oblaka 3D točk. Razviti sistem tudi primerno ovrednotite.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Dejan Štepec sem avtor diplomskega dela z naslovom:

Razvoj sistema za prepoznavanje objektov na osnovi oblakov točk.

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Danijela Skočaja,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, 1. septembra 2015 Podpis avtorja:

Zahvaljujem se mentorju doc. dr. Danijelu Skočaju za strokovno pomoč in usmerjanje pri izdelavi diplomskega dela. Hvala tudi družini in vsem prijateljem, ki so me podpirali pri študiju in izdelavi diplomskega dela.

Kazalo

Povzetek

Abstract

1 Uvod **1**

1.1 Motivacija 1

1.2 Sorodna dela 2

1.3 Cilj diplomskega dela 4

1.4 Zgradba diplomskega dela 4

2 Teoretični del **5**

2.1 Opis sistema 5

2.2 Zajem 3D podatkov 6

2.3 Obdelava oblaka 3D točk 14

2.4 Selekcija značilnih točk 16

2.5 Opis značilnih točk 18

2.6 Redukcija dimenzionalnosti opisnikov 26

2.7 Model vreče besed 29

2.8 Klasifikacija 31

3 Implementacija sistema **35**

3.1 Uporabljene tehnologije 36

3.2 Prepoznavanje objektov v realnem času 39

4 Rezultati **43**

4.1 Zbirka oblakov točk 43

4.2 Prepoznavanje objektov v realnem času 56

5 Zaključek **63**

Seznam uporabljenih kratic

kratica	angleško	slovensko
PCL	Point Cloud Library	knjižnica za obdelavo oblakov točk
PCA	Principal Component Analysis	analiza glavnih komponent
SVM	Support Vector Machine	metoda podpornih vektorjev
ROS	Robot Operating System	programsko ogrodje za robotske sisteme
PFH	Point Feature Histogram	histogram opisnikov točk
FPFH	Fast Point Feature Histogram	hitri histogram opisnikov točk
VFH	Viewpoint Feature Histogram	histogram opisnikov in lege točke

Povzetek

V diplomskem delu se posvečamo prepoznavanju objektov na osnovi oblakov točk. Sistem najprej implementiramo z različnimi 3D opisniki ter naredimo temeljito analizo s preučevanjem rezultatov dobljenih na uveljavljeni zbirki oblakov točk. Ovrednotimo tudi, kako na klasifikacijsko točnost vpliva redukcija dimenzionalnosti opisnika ter predstavimo različne načine selekcije ključnih točk. Rezultate primerjamo z rezultati sistema, ki prepoznava kategorijo objektov iz slik. Sistem ovrednotimo na različno velikih množicah objektov ter na primeru zelo podobnih objektov, kjer pokažemo slabosti opisnikov, ki temeljijo zgolj na 3D informaciji.

Na podlagi rezultatov, doseženih na zbirki, implementiramo sistem za prepoznavanje objektov na osnovi oblakov točk v realnem času. Sistem implementiramo tako, da omogoča gradnjo baze objektov, ki jih želimo prepoznavati. Sistem ovrednotimo na bazi 18 objektov, ki jo zgradimo v ta namen. Rezultati pokažejo, da lahko zgolj z informacijo o obliki objekta dosežemo dobre rezultate.

Ključne besede: prepoznavanje objektov, 3D, vreča besed, PFH, FPFH, VFH, spin slike, SVM.

Abstract

In this thesis we address the problem of point cloud based 3D object recognition. First we develop the system with different 3D shape descriptors and then make a thorough analysis of results gathered from testing the system on a publicly available benchmark database. We also evaluate how dimensionality reduction and different methods for feature point selection influence on classification accuracy. The system is compared with the system for object recognition in 2D. We evaluate the system on different number of objects, including on objects that are very similar by shape.

Based on results from the dataset, we develop a real-time system for object recognition. The system allows us to build our own database of objects that we want to recognize. We constructed a database of 18 objects on which we evaluated the system. Results presented in this thesis have shown that we can recognize objects sufficiently good only with 3D shape information.

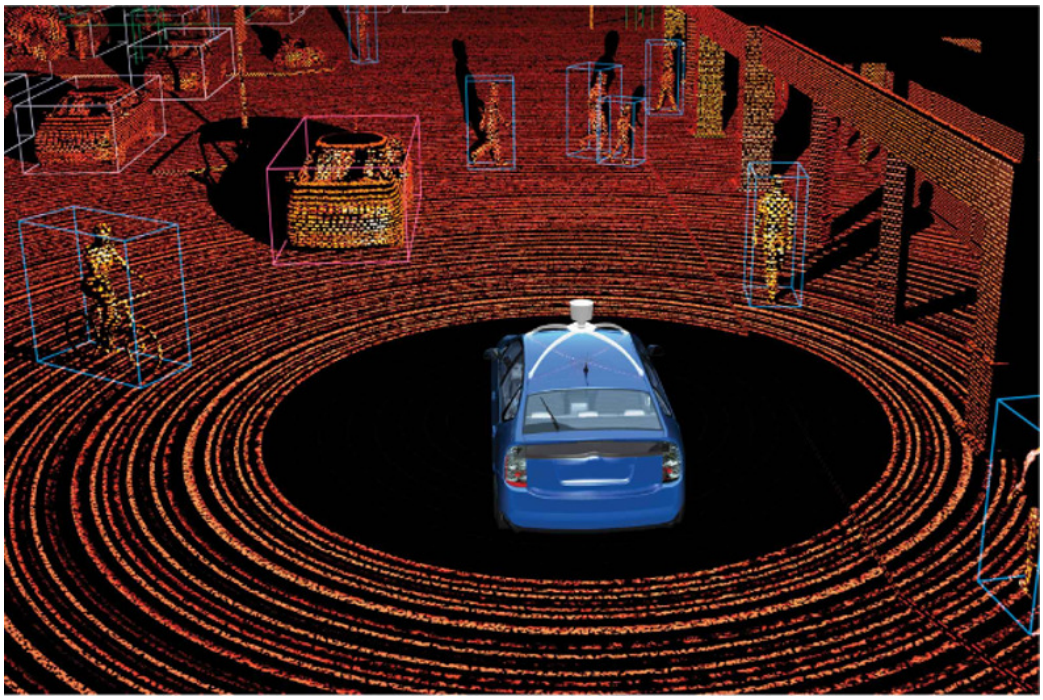
Keywords: object recognition, 3D, bag of words, PFH, FPFH, VFH, Spin Images, SVM.

Poglavje 1

Uvod

1.1 Motivacija

Prepoznavanje objektov je zelo pomembno opravilo na več različnih področjih. Ljudje pri prepoznavanju objektov nimamo težav in si sprva težko predstavljamo težavnost problema. Pri strojnem prepoznavanju objektov je sprva prevladovala slikovna informacija, v zadnjem času pa se poleg slikovne informacije veliko uporablja tudi globinska. 3D oblak točk lahko danes zajamemo na zelo enostaven in poceni način. Še ne dolgo nazaj to ni bilo tako. Prepoznavanje objektov na osnovi oblakov točk je zelo napredovalo s pojavitvijo poceni naprav za zajemanje 3D oblakov točk, kot je naprava Kinect, pa tudi zaradi razvoja avtonomno vozečih avtomobilov, kakršnega razvija Google [4]. Večina avtonomno vozečih avtomobilov za zajem okolice uporablja 3D laserske skenerje. Primer tako zajetega oblaka točk prikazuje slika 1.1. Danes veliko robotskih sistemov omogoča zajem oblaka točk in tako omogoča bistveno boljšo manipulacijo z objekti. Prepoznavanje objektov je pomembno predvsem pri kognitivnih robotih, s katerimi želimo posnemati človeško inteligenco. Z razširitvijo avtonomnih letalnikov se razvijajo nova področja uporabe, saj lahko z zajetimi slikami generiramo oblak točk površja. Primer tako zajetega oblaka točk podjetja Modri planet [2] prikazuje slika 1.2. Prepoznavanje terena, vegetacije in objektov je tukaj pomembno opravilo.



Slika 1.1: Primer oblaka točk, ki ga z avtonomnim vozilom zajame Google [6].

1.2 Sorodna dela

Prepoznavanje objektov na osnovi oblakov točk je v zadnjih letih postalo zelo aktualno raziskovalno področje predvsem zaradi pojava poceni naprav, ki omogočajo zajemanje oblakov točk. K temu je v zadnjih letih s svojimi raziskovalnimi deli in odprtokodno programsko opremo največ prispeval raziskovalni laboratorij Willow Garage. V diplomskem delu uporabljamo njihovo odprtokodno knjižnico Point Cloud Library (PCL) [31]. Ta namreč vsebuje vse opisnike in algoritme, ki smo jih uporabili pri obdelavi oblaka točk. Pri implementaciji sistema za prepoznavanje objektov v realnem času uporabljamo tudi njihovo odprtokodno programsko ogrodje Robot Operating System (ROS) [27], s katerim smo logično povezali komponente sistema. Podoben sistem za prepoznavanje objektov na osnovi oblakov točk v realnem času z globalnim opisnikom VFH je predstavljen v [35]. V delu [13] je predstavljena uporaba PCA za redukcijo dimenzionalnosti opisnikov. Testiranje

Slika 1.2: Primer oblaka točk zajetega z avtonomnim letalnikom.

je bilo izvedeno na isti zbirki oblakov točk, tako da smo predvsem za opisnik VFH imeli direktno primerjavo. V delu [24] so za prepoznavanje na zbirki oblakov točk živali uporabili opisnik FPFH in model vreče besed, tako da smo lahko dobili občutek o vplivu velikosti slovarja na klasifikacijsko točnost. V delu [19] sta predstavljena opisnik spin slike ter sistem za prepoznavanje 3D objektov, ki temelji na tem opisniku. V delu [21] so predstavljeni zbirka oblakov točk, ki smo jo uporabili v diplomskem delu, in rezultati prepoznavanja objektov v 3D s pomočjo opisnika spin slike. Podana je tudi primerjava med prepoznavanjem v 2D in 3D. V delu [33] so predstavljeni različni detektorji ključnih točk v 3D in rezultati testiranj, ki kažejo robustnost posameznih detektorjev. V delu [23] so predstavljeni opisnik SIFT ter rezultati prepoznavanja objektov v 2D s tem opisnikom. V diplomskem delu smo implementirali podoben sistem ter ga primerjali s sistemom za prepoznavanje objektov v 3D.

1.3 Cilj diplomskega dela

Cilj diplomskega dela je implementirati sistem, ki bo omogočal prepoznavanje manjših objektov na podlagi oblakov točk. V diplomskem delu bomo najprej ovrednotili različne principe prepoznavanja 3D objektov na uveljavljeni zbirki oblakov točk, tako da bomo ovrednotili različne lokalne ter globalne opisnike in različne pristope selekcije značilnih točk. Ovrednotili bomo, kako na klasifikacijsko točnost vpliva redukcija dimenzionalnosti opisnikov, in dosežene rezulatate primerjali z rezultati, do katerih pridemo samo s slikovno informacijo.

Cilj diplomskega dela je implementirati tudi sistem, ki omogoča prepoznavanje objektov v realnem času in sicer tako, da bo omogočal gradnjo baze oblakov točk, ki jih bomo nato želeli prepoznati. Sistem bomo implementirali na podlagi izkušenj in rezultatov prepoznavanja objektov na zbirki oblakov točk.

1.4 Zgradba diplomskega dela

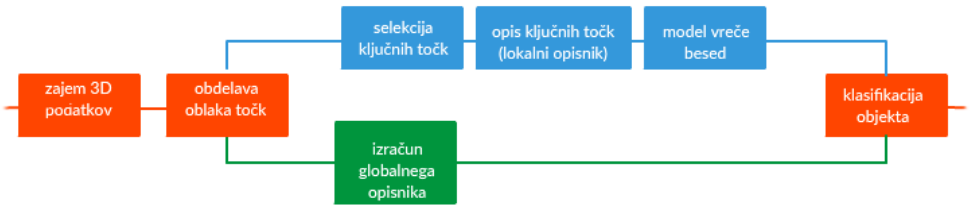
Diplomsko delo je v nadaljevanju razdeljeno na tri poglavja. V drugem poglavju so predstavljene teoretične osnove. Poglavje smo razdelili po logičnih sklopih, ki jih potrebujemo za implementacijo sistema od zajema oblaka točk do klasifikacije z metodami strojnega učenja. V tretjem poglavju je predstavljena implementacija sistema. Najprej so predstavljene uporabljene tehnologije, nato pa zbirka oblakov točk in sistem, ki smo ga razvili za prepoznavanje objektov v realnem času. V četrtem poglavju so predstavljeni rezultati. Na koncu v zaključku so podane sklepne ugotovitve in obravnavane možne izboljšave.

Poglavje 2

Teoretični del

2.1 Opis sistema

V tem poglavju so opisane teoretične osnove, ki smo jih potrebovali za izdelavo diplomske naloge. Na sliki 2.1 je predstavljen poenostavljen model sistema, ki smo ga uporabili za prepoznavanje objektov. Sistem je razdeljen na dva dela. Modra barva predstavlja uporabo lokalnih opisnikov, zelena barva pa uporabo globalnega opisnika. Oranžna barva predstavlja korake, ki so neodvisni od vrste opisnikov. Vhod v sistem je oblak točk, izhod iz sistema pa izpis prepoznanega objekta. Sistem je v nadaljevanju poglavja podrobneje razložen.



Slika 2.1: Poenostavljena predstavitev sistema za prepoznavanje objektov.

Za zajem 3D oblaka točk smo uporabili napravo Kinect, ki deluje na

principu strukturirane svetlobe. Za to napravo smo se odločili zaradi zajemanja gostega oblaka točk v realnem času in dobre integracije v programskem ogrodju ROS. Lahko bi uporabili tudi stereo sistem kamer, s katerim bi dobili redek oblak točk. Potreben bi bil postopek goste rekonstrukcije, ki je lahko časovno zelo zahteven.

Iz pridobljenega oblaka točk je potrebno izločiti objekte, kar storimo s postopkom segmentacije in gručenja.

Segmentiran objekt opišemo z opisniki. Opisnike ločimo na lokalne in globalne. Zaradi računske zahtevnosti izračunamo opisnike zgolj na ključnih točkah, ki jih detektira detektor ključnih točk. Lahko uporabimo tudi naključno izbrane točke. Lokalne opisnike združimo v en fiksni opisnik s pomočjo modela vreče besed.

Anotirani opisniki točk tvorijo vhod v učni algoritem, za katerega smo uporabili linearni SVM klasifikator, saj ga članki, ki smo jih prebrali, navajajo kot najustreznejšega. Linearni SVM klasifikator na podlagi učne množice opisnikov tvori model, ki vhodni opisnik klasificira v najbližji razred objektov.

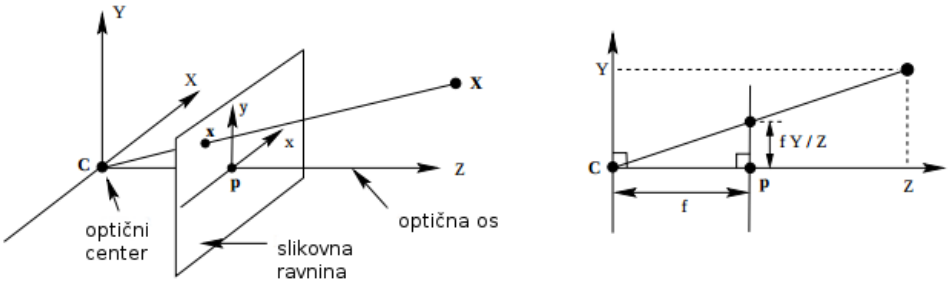
2.2 Zajem 3D podatkov

Za zajem slike uporabljamo kamero. Preprost geometrijski model kamere, ki točke iz prostora preslika v sliko, je centralno projekcijski model (ang. pinhole camera model). Opis centralno projekcijskega modela in epipolarne geometrije je v nadaljevanju povzet po [18].

2.2.1 Centralno projekcijski model

Centralno projekcijski model definira geometrijske relacije med 3D točko v prostoru in njeno 2D projekcijo na slikovni ravnini, kar imenujemo tudi perspektivna projekcija. Center projekcije, kjer se sekajo vsi žarki, imenujemo optični center, os, ki seka slikovno ravnino pod pravim kotom in gre skozi optični center, pa optično os. Točko, kjer se sekata slikovna ravnina in optična

os, imenujemo glavna točka. Razdaljo med optičnim centrom in slikovno ravnino imenujemo goriščna razdalja. Model je predstavljen na sliki 2.2.



Slika 2.2: Centralno projekcijski model [18].

Na sliki 2.2 je optični center izhodišče kooordinatnega sistema. 3D točka v prostoru $\boldsymbol{X_{cam}} = (X, Y, Z)^\intercal$ se preslika v točko $\mathbf{x}$, kjer se sekata daljica $\boldsymbol{CX}$ in slikovna ravnina. Projekcijo lahko s pomočjo podobnih trikotnikov zapišemo z izrazom (2.1). Poudariti je potrebno, da je točka v prostoru podana v koordinatnem sistemu kamere.

$$(X, Y, Z)^\intercal \mapsto (fX/Z, fY/Z, f)^\intercal \tag{2.1}$$

Če so koordinate podane v homogenem sistemu, lahko izraz (2.1) zapišemo z izrazom (2.2).

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \mapsto \begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & f & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.2}$$

Izraz (2.2) predpostavlja, da je izhodišče koordinatnega sistema slikovne ravnine v glavni točki. V praksi seveda to ni nujno, zato upoštevamo še odmik. Projekcijo zapišemo z izrazom (2.3).

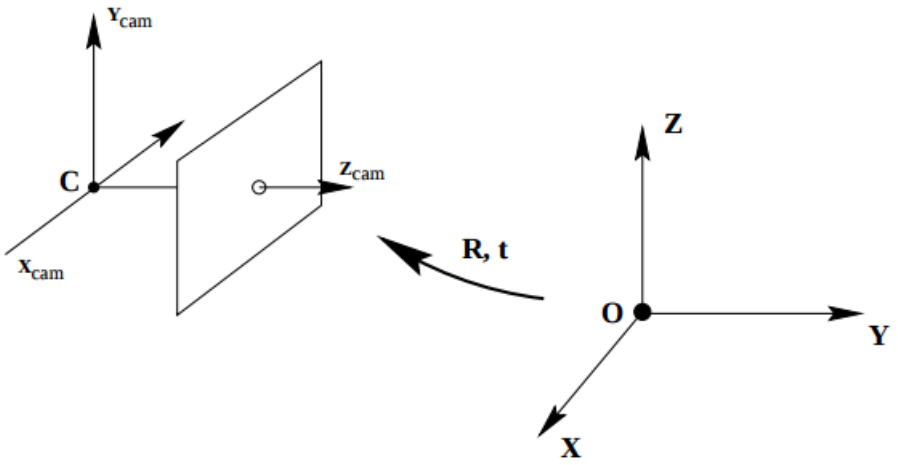
$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \mapsto \begin{bmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{bmatrix} = \begin{bmatrix} f & & p_x & 0 \\ & f & p_y & 0 \\ & & f & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.3}$$

Iz izraza (2.3) lahko izrazimo kalibracijsko matriko (2.4) in vse skupaj zapišemo krajše z izrazom (2.5).

$$K = \begin{bmatrix} f & & p_x & 0 \\ & f & p_y & 0 \\ & & f & 0 \end{bmatrix} \tag{2.4}$$

$$\boldsymbol{x} = K[I|0]\boldsymbol{X_{cam}} \tag{2.5}$$

V splošnem bodo koordinate podane v svetovnem koordinatnem sistemu in ne v kooordinatnem sistemu kamere. Koordinatna sistema sta povezana z rotacijo in translacijo, kar vidimo na sliki (2.3).



Slika 2.3: Evklidska transformacija med koordinatnima sistemoma [18].

Pretvorbo lahko zapišemo z izrazom (2.6), kjer $\tilde{C}$ predstavlja koordinate optičnega centra v svetovnem koordinatnem sistemu, R pa predstavlja 3×3 rotacijsko matriko koordinatnega sistema kamere.

$$\boldsymbol{X_{cam}} = \begin{bmatrix} R & -R\tilde{C} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.6}$$

Z izrazom (2.7) lahko sedaj zapišemo projekcijsko matriko P, z izrazom (2.8) pa preslikavo točke $\boldsymbol{X}$ v svetovnem koordinatnem sistemu na sliko.

$$P = K[R|t] \quad t = -R\tilde{C} \tag{2.7}$$

$$\boldsymbol{x} = P\boldsymbol{X} \tag{2.8}$$

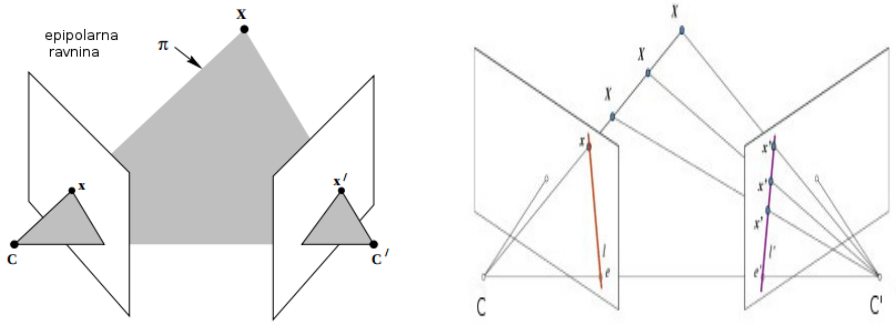
2.2.2 Epipolarna geometrija

Če objekt slikamo s pomočjo dveh ali več kamer, lahko geometrijske relacije med točkami na slikovnih ravninah opišemo z epipolarno geometrijo. Največkrat se jo uporablja pri iskanju korespondenčnih točk pri stereo ujemanju.

Če predpostavimo, da točko $\boldsymbol{X} = (X, Y, Z)$ vidimo iz dveh kamer, tako da je preslikana v točko $\boldsymbol{x}$ na prvi slikovni ravnini in točko $\boldsymbol{x'}$ na drugi slikovni ravnini, potem lahko z epipolarno geometrijo opišemo, kako sta povezani. Točke $\boldsymbol{X}$, $\boldsymbol{x}$, $\boldsymbol{x'}$ in optična centra kamer so koplanarne in tvorijo ravnino, na sliki 2.4a označeno s π.

Ravnina π je definirana s premico, ki povezuje optična centra, in žarkom, ki ga definira točka $\boldsymbol{x}$. Ugotovimo, da žarek, ki definira točko $\boldsymbol{x'}$, leži v ravnini π. Iz tega sledi, da točka $\boldsymbol{x'}$ leži na presečišču ravnine π z drugo slikovno ravnino (glej sliko 2.4b). Ob podanem paru stereo slik obstaja za vsako točko na prvi sliki, epipolarna premica l' na drugi sliki. Vsaka točka $\boldsymbol{x'}$ na drugi sliki, ki je korespondenčna točki $\boldsymbol{x}$ na prvi sliki, mora ležati na pripadajoči epipolarni premici l'. Velja izraz (2.9).

$$\boldsymbol{x'}^{\intercal} F \boldsymbol{x} = 0 \tag{2.9}$$



(a) (b)

Slika 2.4: Epipolarna geometrija. (a) Optična centra kamer sta označena s C in C'. $\boldsymbol{X}$ je točka v prostoru, $\boldsymbol{x}$ in $\boldsymbol{x}'$ pa sta njuni projekciji na slikovni ravnini. Vse točke ležijo na ravnini π. (b) Žarek, ki gre skozi točko $\boldsymbol{x}$ in optični center C, se na drugi slikovni ravnini preslika v l' [18].

F imenujemo fundamentalna matrika in opisuje geometrijske relacije med korespondenčnimi točkami v paru stereo slik. Epipolarni premici lahko izračunamo z izrazoma (2.10) in (2.11).

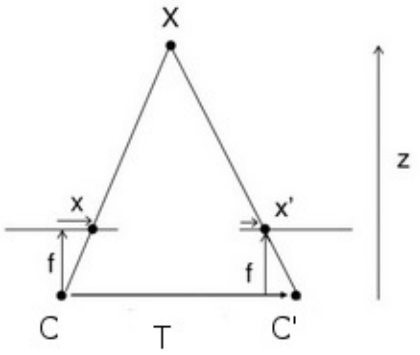
$$l' = F\boldsymbol{x} \tag{2.10}$$

$$l = F^{\intercal}\boldsymbol{x}' \tag{2.11}$$

2.2.3 Globinska informacija iz stereo slik

Pri klasičnem zajemanju slik z eno kamero izgubimo podatek o globini, saj se vse točke, ki ležijo na istem žarku, preslikajo v isto točko na slikovni ravnini (glej sliko 2.4b). Če uporabljamo stereo sistem, se te točke preslikajo v različne točke na drugi slikovni ravnini. S stereo sistemom tako lahko dobimo globinsko informacijo točke.

S pomočjo podobnih trikotnikov (slika 2.5) lahko izpeljemo enačbo za globino točke (2.12). Izraz $\boldsymbol{x} - \boldsymbol{x}'$, ki ga pišemo tudi kot $\boldsymbol{d}(\boldsymbol{x}, \boldsymbol{y})$, imenujemo disperzija. Za izračun disperzije moramo za vsako točko v prvi slikovni



Slika 2.5: Stereo sistem

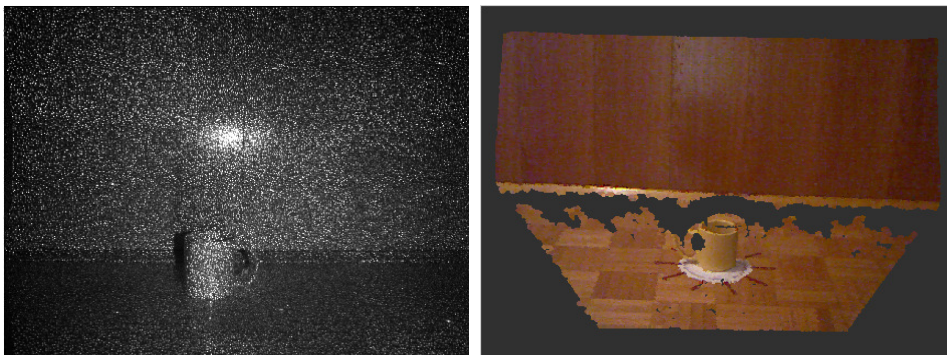
ravnini ($\boldsymbol{x}$) najti njeno korespondenčno točko v drugi slikovni ravnini ($\boldsymbol{x}'$). Pomagamo si z epipolarno geometrijo.

$$Z = \frac{Tf}{\boldsymbol{x} - \boldsymbol{x}'} \tag{2.12}$$

Namesto dveh kamer lahko uporabimo tudi kombinacijo kamere in projektorja. Projektor projicira strukturirano svetlobo. Strukturirana svetloba je urejen konstanten vzorec žarkov, ki ga projiciramo v prostor. V kombinaciji s kamero - v primeru laserskih žarkov je to infrardeča kamera - lahko zajamemo globinsko informacijo. V nadaljevanju opišemo postopek na primeru naprave Kinect.

2.2.4 Senzor Kinect

Senzor Kinect je sestavljen iz infrardečega projektorja, infrardeče kamere in RGB kamere. Projektor oddaja laserski žarek, ki se s pomočjo uklonske mrežice razprši v konstanten vzorec laserskih žarkov. Projicirani vzorec žarkov zajamemo z infrardečo kamero, kot je prikazano na sliki 2.6a. Globinsko mapo izračunamo na podoben način kot pri stereo slikah. V pomnilniku naprave je shranjen referenčni vzorec žarkov na virtualni ravnini, s katerim primerjamo projicirani vzorec. Iskanje korespondenc je v tem primeru mnogo lažje.



(a) Laserski žarki, zajeti z infrardečo kamero.

(b) Oblak točk, zajet z napravo Kinect.

Slika 2.6: Zajemanje oblaka točk s strukturirano svetlobo.

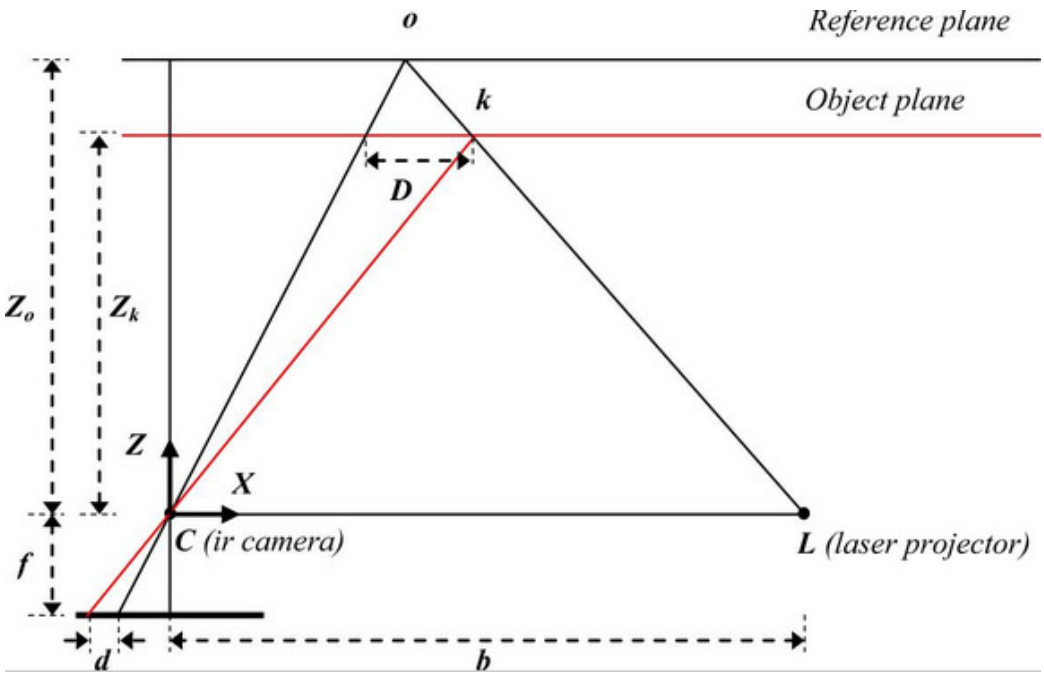
Ko najdemo korespondenčna vzorca žarkov na referenčni ravnini in na ravnini objekta, izračunamo disperzijo. Bolj kot se ravnini razlikujeta v oddaljenosti, večja je disperzija. Postopek prikazuje slika 2.7. S pomočjo podobnih trikotnikov z izrazoma (2.13) in (2.14) izrazimo globino točke, kar predstavlja izraz (2.15). Izpeljava enačb je povzeta po [20]. Z_k predstavlja globino objekta, b je razdalja med infrardečo kamero in projektorjem, f pa goriščna razdalja infrardeče kamere. D predstavlja odmik v prostoru objekta, d pa disperzijo v prostoru slike.

$$\frac{D}{b} = \frac{Z_o - Z_k}{Z_o} \tag{2.13}$$

$$\frac{d}{f} = \frac{D}{Z_k} \tag{2.14}$$

$$Z_k = \frac{Z_o}{1 + \frac{Z_o}{fb} d} \tag{2.15}$$

Preostali dve koordinati izračunamo z izrazoma (2.16) in (2.17), kjer sta x_k in y_k koordinati na sliki, x_o in y_o pa koordinati glavne točke. δx in δy predstavljata korekciji za napako leč. Rezultat je oblak točk na sliki 2.6b.



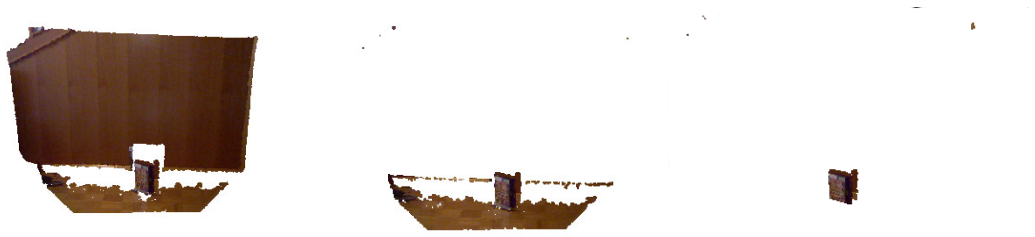
Slika 2.7: Geometrijski model naprave Kinect [20].

$$X_k = -\frac{Z_k}{f}(x_k - x_o + \delta x) \tag{2.16}$$

$$Y_k = -\frac{Z_k}{f}(y_k - y_o + \delta y) \tag{2.17}$$

2.3 Obdelava oblaka 3D točk

V oblaku točk, ki smo ga zajeli, je potrebno prepoznati objekte. Oblak točk lahko vsebuje mnogo objektov, te je potrebno segmentirati in v nadaljevanju prepoznati. Največji del oblaka točk po navadi predstavljajo ravnine, ki so lahko del sten ali tal. Za identifikacijo in odstranitev ravnin obstaja veliko algoritmov. Najbolj znan algoritem je RANSAC [16], ki temelji na iskanju najboljše ravnine. Izvaja se N iteracij, kjer vsakič naključno izbere tri točke. Z njimi lahko izračunamo ravnino, na kateri ležijo te tri točke. Ko izračunamo ravnino, najdemo vse točke z neko napako, ki ležijo na tej ravnini. Cilj algoritma je najti ravnino, ki ji pripada največ točk. Primer odstranjevanja ravnin pri 1000 iteracijah in 3 cm dovoljene napake prikazuje slika 2.8, kjer odstranimo tla in steno.



(a) (b) (c)

Slika 2.8: Odstranjevanje ravnin z algoritmom RANSAC.

Po odstranitvi ravnin je potrebno segmentirati objekte. V oblaku se lahko nahaja več objektov, poleg tega pa je lahko prisotno tudi veliko šuma. Za segmentacijo lahko uporabimo metodo evklidskega gručenja [3]. Postopek gručenja je predstavljen v algoritmu 2.

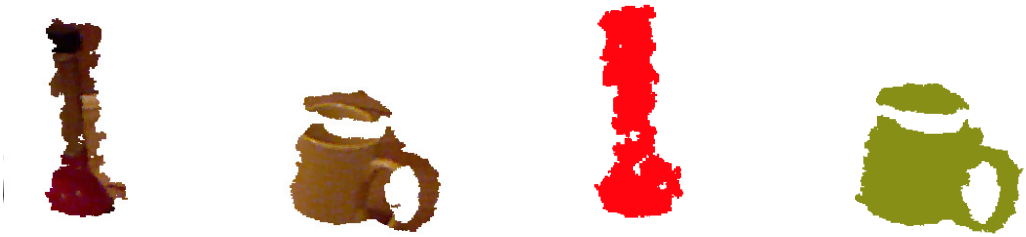
Algoritem 2 Evklidsko gručenje

C = množica gruč, Q = vrsta neobdelanih točk

Za vsako točko p_i iz oblaka točk P naredimo naslednje korake:

- točko p_i dodamo v vrsto Q,
- za vsako točko p_i v Q najdemo množico sosedov P_k^i v radiju r,
- za vsako sosedo p_k^i preverimo, ali smo jo že sprocesirali; v nasprotnem primeru jo dodamo v Q,
- ko sprocesiramo vse točke v Q, dodamo Q v seznam gruč C in resetiramo Q,
- algoritem ustavimo, ko sprocesiramo vse točke v P, ki so sedaj del C.

Ko imamo gruče, se lahko odločimo, da obdržimo le tiste, ki imajo dovolj točk. Postopek gručenja je predstavljen na sliki 2.28, kjer smo najdene gruče obarvali z različnimi barvami. Predhodno smo že odstranili ravnine.



(a) Vhodni oblak točk. (b) Najdene gruče objektov.

Slika 2.9: Postopek gručenja na oblaku točk.

2.4 Selekcija značilnih točk

Ko imamo sprocesiran oblak točk, je potrebno izračunati opisnike. Opisnike ločimo na globalne in lokalne. Globalni opisnik opisuje oblak točk kot celoto, lokalni pa okolico posamezne točke. Opisniki so predstavljeni v naslednjem poglavju. Ker je računanje opisnikov računsko zahtevna operacija, je potrebno izbrati podmnožico točk, ki so pomembne po neki metriki. Dober detektor značilnih točk naj bi imel naslednje lastnosti:

- izbrana mora biti majhna podmnožica vseh točk,
- zagotovljena je ponovljivost, isto značilnico najdemo tudi v drugem oblaku točk na isti lokaciji,
- okolica značilnice je diskriminativna.

Iskanje značilnih točk je pomembno pri lokalnih opisnikih. Pri globalnih se večinoma poslužujemo metode za redčenje oblaka točk. Točke lahko izberemo tudi naključno. V nadaljevanju predstavimo detektor značilnih točk Harris3D [36]. V diplomski nalogi smo uporabili omenjeni detektor in naključno izbiranje točk.

2.4.1 Detektor značilnih točk Harris3D

Prvotni detektor značilnih točk Harris je namenjen iskanju značilnic v slikah [17]. Priljubljen je predvsem zaradi invariantnosti na rotacijo, osvetlitev in šum v slikah. Detektor temelji na detektiranju oglišč (ang. corners) v slikah. Oglišča predstavljajo veliko varianco gradienta v sliki. Majhen premik v oglišču pomeni veliko spremembo intenzitete, velikost te spremembe opišemo z uteženo avtokorelacijsko funkcijo (2.18), kjer vektor (u,v) predstavlja premik. Za utežitveno funkcijo w se uporablja gausovo porazdelitev. $I(x, y)$ predstavlja intenziteto slikovnega elementa.

$$\mathbf{E(u,v)} = \sum_{x_i, y_i} w(x_i, y_i)[I(x_i + u, y_i + v - I(x_i, y_i)]^2 \qquad (2.18)$$

Za majhne premike lahko izraz (2.18) linearno aproksimiramo z izrazom (2.19), kjer je M kovariančna matrika gradientov (glej izraz 2.20). Vsoto smo nadomestili s konvolucijo.

$$E(u,v) = \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix} \tag{2.19}$$

$$M = G(\sigma) \star \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \tag{2.20}$$

Oglišče detektiramo z analizo kovariančne matrike, tako da poiščemo lastne vrednosti matrike. Oglišče se nahaja v točki, kjer sta obe lastni vrednosti veliki. Ker je računanje lastnih vrednosti v vsaki točki računsko prezahtevno, uporabljamo odzivno funkcijo (2.21). Za oglišča vzamemo točke, kjer odzivna funkcija preseže neki prag.

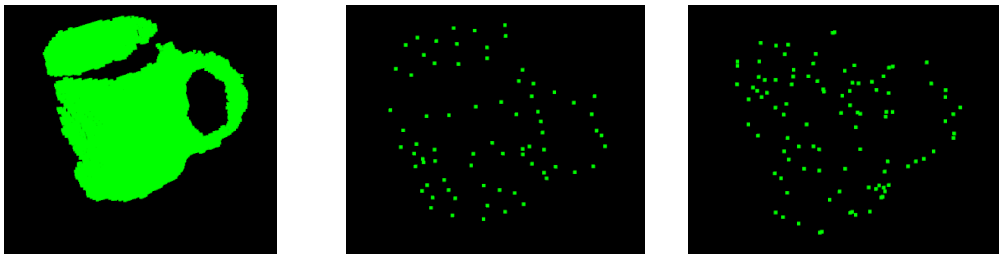
$$r(x,y) = det(M) - \alpha trace^2(M) \tag{2.21}$$

V primeru Harris3D gradiente nadomestimo s površinskimi normalami. Obstajata tudi dve dodatni verziji Harris3D detektorja, ki pa se razlikujeta samo po odzivni funkciji. Odzivno funkcijo za Lowe Harris3D izračunamo z izrazom (2.22), Noble Harris3D pa z izrazom (2.23).

$$r(x,y,z) = \frac{det(M)}{trace^2(M)} \tag{2.22}$$

$$r(x,y,z) = \frac{det(M)}{trace(M)} \tag{2.23}$$

Slika 2.10 prikazuje detektirane značilnice na skodelici. Pri Harris3D smo uporabili odzivno funkcijo (2.22). Upoštevali smo 100 točk z največjo vrednostjo odzivne funkcije. Za naključno izbiranje smo izbrali 100 naključnih točk z enakomerno porazdelitvijo.



(a) (b) (c)

Slika 2.10: (a) Začetni oblak točk. (b) Značilnice, detektirane s Harris3D. (c) Značilnice, detektirane z naključnim izbiranjem.

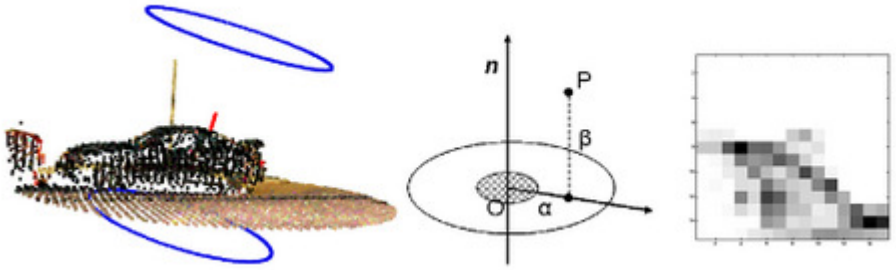
2.5 Opis značilnih točk

Koordinate točk nosijo premalo infomacij, da bi lahko neposredno iz njih sklepali o objektu, ki ga predstavljajo. Točke je potrebno opisati z lokalnimi ali globalnimi opisniki. Obstaja veliko 3D opisnikov. Sprva so uporabljali preproste opisnike, ki so bili zmožni iz površin zaznati ravnine, cilindre in sferične objekte. Kasneje se je pojavila nova vrsta opisnikov, ki temeljijo na transformacijah iz 3D v 2D informacijo. Najbolj znan opisnik iz te družine so spin slike (ang. Spin Images) [19]. S pojavom poceni RGBD naprav, kot je Kinect, so se začeli pojavljati tudi novi opisniki. 3D informacija, pridobljena s Kinectom, vsebuje veliko šuma, zato so rezultati prej omenjenih opisnikov slabi. Problem je tudi njihova računska zahtevnost. Pojavila se je nova družina opisnikov, ki daje precej boljše rezultate [10]. V nadaljevanju so predstavljeni glavni opisniki.

2.5.1 Spin slike (Spin images)

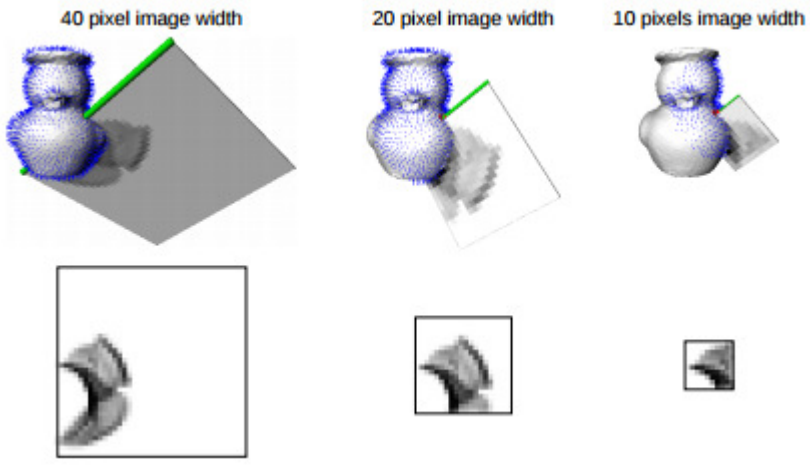
Spin sliko, ki je najstarejši izmed predstavljenih opisnikov, je leta 1999 predstavil Andrew E. Johnson v [19]. Prvotno je bil razvit za opis poligonskih mrež. Obstaja tudi različica, ki se izračuna neposredno na oblaku točk [9]. Izračun spin slike poteka v cilindričnem koordinatnem sistemu, ki ga določata referenčna točka, v kateri računamo opisnik, in njena normala. Točke, ki jih

definira cilindrični koordinatni sistem, so preslikane z izračunom α, ki predstavlja oddaljenost od normale, in β, ki predstavlja oddaljenost od ravnine, ki jo definirata točka in pripadujoča normala, v kateri računamo opisnik 2.11.



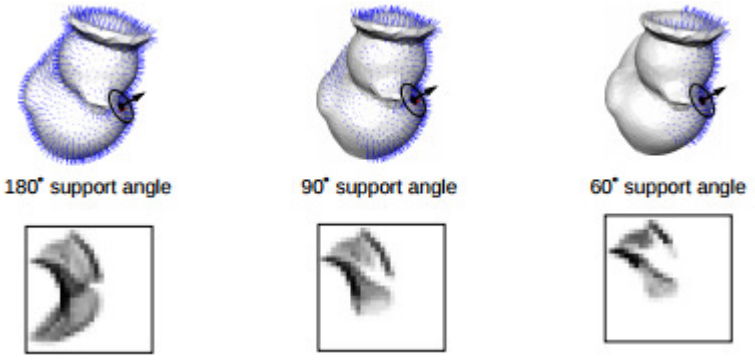
Slika 2.11: Računanje spin slike: Leva slika predstavlja cilindrični model z normalo točke, ki ga definira. Desni del slike predstavlja izračun, spin slika je histogram točk, ki padejo v določen α, β razdelek [19].

Pomemben parameter je velikost spin slike, ki pove, kolikšen del površine opišemo. Vpliv parametra je prikazan na sliki 2.12. Večja je širina slika, večji del modela opišemo.



Slika 2.12: Večja je velikost slike, večji volumen opišemo [19].

Pomemben parameter je tudi podporni kot, ki predstavlja maksimalni kot med normalo referenčne točke in normalami preostalih točk. Vpliv parametra opisuje slika 2.13.



Slika 2.13: Vpliv velikosti podpornega kota [19].

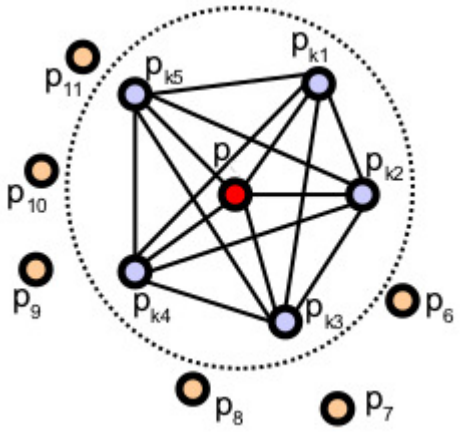
Implementacija v knjižnici PCL, ki jo uporabljamo v diplomski nalogi, predstavlja 2D histogram s 153 dimenzionalnim opisnikom.

2.5.2 Point Feature Histograms (PFH)

Point Feature Histograms (PFH) [32] je lokalni opisnik, ki opisuje lastnosti površine v okolice točke 3D objekta. Cilj PFH je zakodirati geometrijske lastnosti v okolici točke z visoko dimenzionalnim histogramom. Visoko dimenzionalni histogram omogoča bogat opis množice značilk, invariantno glede na pozicijo in orientacijo pripadajoče površine. Zelo dobro se obnese tudi pri različnih gostotah vzorčenja in prisotnosti šuma. Temelji na relacijah med točkami v bližnji okolici in njihovimi normalami.

Naj bo p neka točka v oblaku točk. Izračun opisnika poteka tako, da najprej poiščemo vse točke, ki so od točke p oddaljene manj kot radij r. Ta proces ponazarja slika 2.14.

Za vsak par točk p_i, p_j (i $\neq$ j) in pripadajoči normali n_i in n_j tvorimo fiksni koordinatni sistem uvw, kot to prikazuje slika 2.15. Označimo točko, ki ima manjši kot med pripadajočo normalo in premico, ki povezuje par točk



Slika 2.14: Soseščina točke p in polno povezani pari za izračun PFH [7].

s p_s in preostalo točko s p_t. Pripadajoči normali označimo z n_s in n_t. S pomočjo fiksnega koordinatnega okvira izračunamo kote α, ϕ in θ.

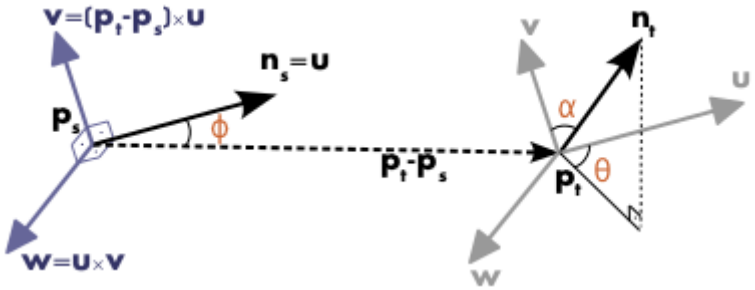
$$\alpha = v \cdot n_s \tag{2.24}$$

$$\phi = u \cdot \frac{p_t - p_s}{||p_t - p_s||} \tag{2.25}$$

$$\theta = arctan(w \cdot n_t, u \cdot n_t) \tag{2.26}$$

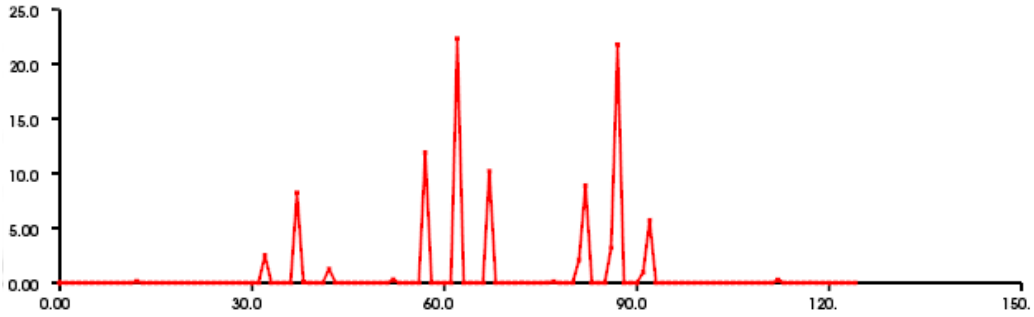
Koti (2.24, 2.25, 2.26) so predstavljeni na sliki 2.15. Trojček $\langle \alpha, \phi, \theta \rangle$ je izračunan za vsak par točk v okolici točke p. Dvanajst vrednosti (kooordinate in normale) smo tako predstavili s tremi lastnostmi. Da ustvarimo končni PFH, združimo vse trojčke v histogram. Interval vsake lastnosti razdelimo na b podintervalov. Ker morajo biti intervali za vsako lastnost razdeljeni v enako število podintervalov, dobimo končni histogram, ki ima b^3 dimenzij. Primer PFH prikazuje slika 2.16.

V diplomski nalogi smo PFH opisniku dodali tudi barvno informacijo. Izkoristili smo dejstvo, da lahko oblaki točk poleg koordinat nosijo tudi RGB informacijo in v vsaki točki, kjer se je računal opisnik PFH, izdelali barvni



Slika 2.15: Fiksni koordinatni sistem in lastnosti PFH [7].

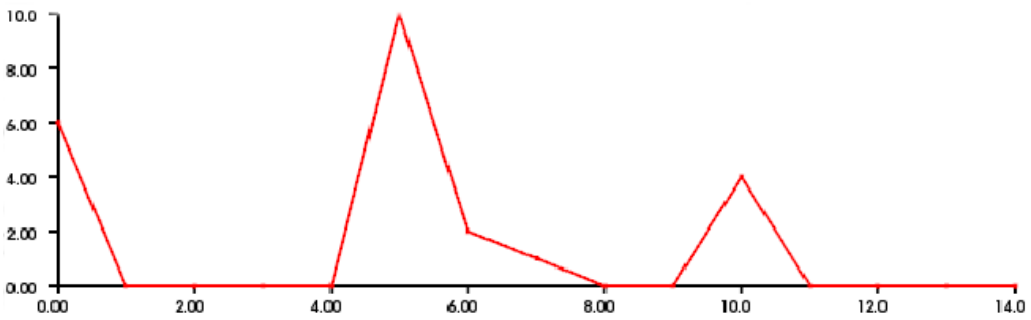
HSV histogram. Za izdelavo histograma smo uporabili barvno informacijo točk v soseščini točke p, na podlagi katerih se računa tudi sam opisnik. RGB barvni prostor smo pretvorili v HSV in za gradnjo histograma uporabili vrednosti odtenkov (10 dimenzij) in intenzivnosti barve (5 dimenzij). Rezultati so predstavljeni v praktičnem delu diplomske naloge. Primer HSV histograma skodelice 2.10a prikazuje slika 2.17. Skodelica vsebuje tri različne barve, kar se opazi tudi na histogramu.



Slika 2.16: Primer PFH opisnika skodelice 2.10a.

2.5.3 Fast Point Feature Histograms (FPFH)

Teoretična računska zahtevnost PFH za dani oblak točk P z n točkami je $O(nk^2)$, kjer je k število sosedov za vsako točko p v P. Računska zahtevnost FPFH [29] je zmanjšana na $O(nk)$, kljub temu pa se diskriminativna moč



Slika 2.17: Primer HSV opisnika skodelice 2.10a.

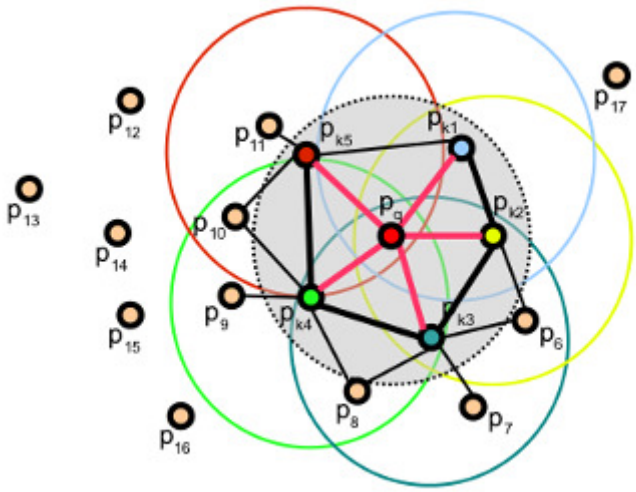
opisnika ni bistveno zmanjšala, kar lahko vidimo v praktičnem delu diplomske naloge.

Izračun opisnika poteka tako, da tako kot pri PFH za vsako točko p najprej izračunamo lastnosti (2.24, 2.25, 2.26). Razlika je v tem, da histogram izračunamo le med točko p in njenimi k najbližjimi neposrednimi sosedami in ne med vsemi pari sosedov 2.18. Ta histogram imenujemo poenostavljeni PFH (SPFH). V drugem delu nato FPFH izračunamo kot uteženo vsoto SPFH k sosed (2.27). Vrednosti so utežene glede na oddaljenost od točke p, kar označimo z w.

$$FPFH(p) = SPFH(p) + \frac{1}{k}\sum_{i=1}^{k}\frac{1}{w^k}SPFH(p_i) \tag{2.27}$$

Glavne razlike med PFH in FPFH so:

- pri FPFH sosede niso polno povezane, kar pomeni, da se izgubi nekaj informacij o geometriji okoli točke p,
- pri PFH zajamemo geometrijske informacije le znotraj radija r, pri FPFH jih lahko zajamemo znotraj radija $2r$,
- zmanjšana računska zahtevnost.



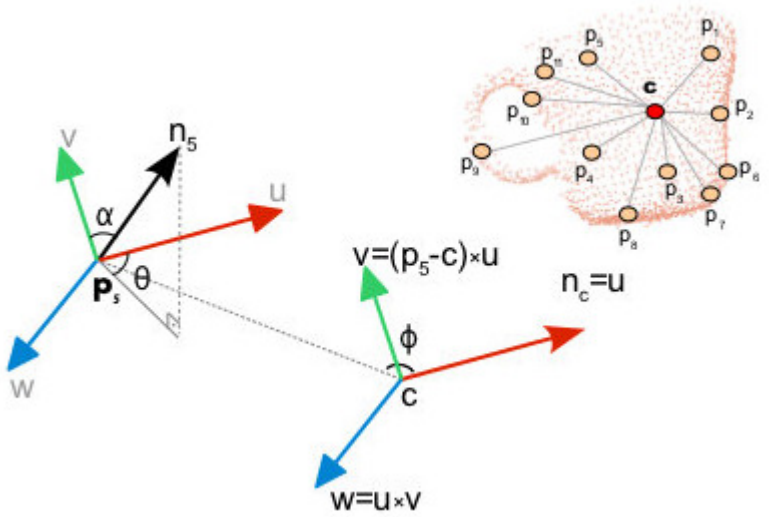
Slika 2.18: FPFH, vsaka točka je povezana le z njenimi neposrednimi sosedami. Vsak neposredni sosed je povezan s svojimi neposrednimi sosedami, končni FPFH predstavlja utežena vsota SPFH. Odebeljene povezave prispevajo dvakrat [29].

2.5.4 Viewpoint Feature Histogram (VFH)

Viewpoint Feature Histogram (VFH) [30] je globalni opisnik. Ta tip opisnikov opisuje globalne lastnosti oblaka točk tako, da za izračun uporablja vse točke. VFH je izpeljan iz FPFH opisnika, dodana mu je varianca na orientacijo in točko pogleda, da lahko z njim prepoznamo tudi orientacijo in kot, pod katerim gledamo objekt (6D lega). PFH in FPFH sta invariantna na orientacijo. 6D pozicija je pomembna predvsem pri robotiki, kjer je dostikrat potrebna manipulacija z objektom.

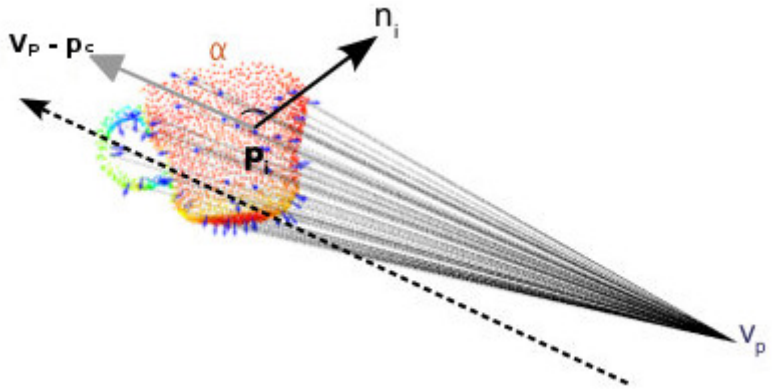
Opisnik je razdeljen na dva dela. Prvi del predstavlja izračun razširjenega FPFH tako, da izračunamo lastnosti (2.24, 2.25, 2.26) med centroidno točko oblaka in ostalimi točkami, kot to prikazuje slika 2.19. Za vsako lastnost naredimo 45-dimenzionalni histogram. Dodatnih 45 dimenzij razširjenega FPFH predstavlja še SPFH centroidne točke z dodanim histogramom oddaljenosti točk od centroidne točke.

Drugi del je izračun komponente točke pogleda. Izračunamo kot med



Slika 2.19: Izračun razširjenega FPFH na primeru točke 5 [30].

centralno smerjo pogleda in vsako izmed normal, kot to prikazuje slika 2.20. Naredimo 128-dimenzionalni histogram. Rezultat je 308-dimenzionalni opisnik.



Slika 2.20: Izračun komponente točke pogleda [30].

2.6 Redukcija dimenzionalnosti opisnikov

Opisniki, ki smo jih opisali, so visokodimenzionalni: PFH je 125-dimenzionalen, FPFH je 33-dimenzionalen, VFH pa 308-dimenzionalen. Proces učenja, opisan v naslednjih poglavjih, je tako lahko časovno zelo zahteven. V podatkih se nahaja tudi precej šuma, vendar ga z metodami redukcije lahko odstranimo. Obstaja veliko tehnik zmanjševanja dimenzionalnosti podatkov. V splošnem jih ločimo na metode, ki projicirajo podatke z zmanjšano dimenzionalnostjo v nov prostor, kjer so novi podatki po navadi zgrajeni iz kombinacije starih, in metode, ki iz množice podatkov izberejo manjšo podmnožico, in ta maksimizira relevantnost pri klasifikaciji. Najbolj znani metodi prvega tipa sta analiza glavnih komponent (PCA) [34] in linearna diskriminativna analiza (LDA) [37], drugega tipa pa informacijski prispevek [15] in Relief [28]. V nadaljevanju je opisana metoda PCA, ki smo jo uporabili v diplomski nalogi.

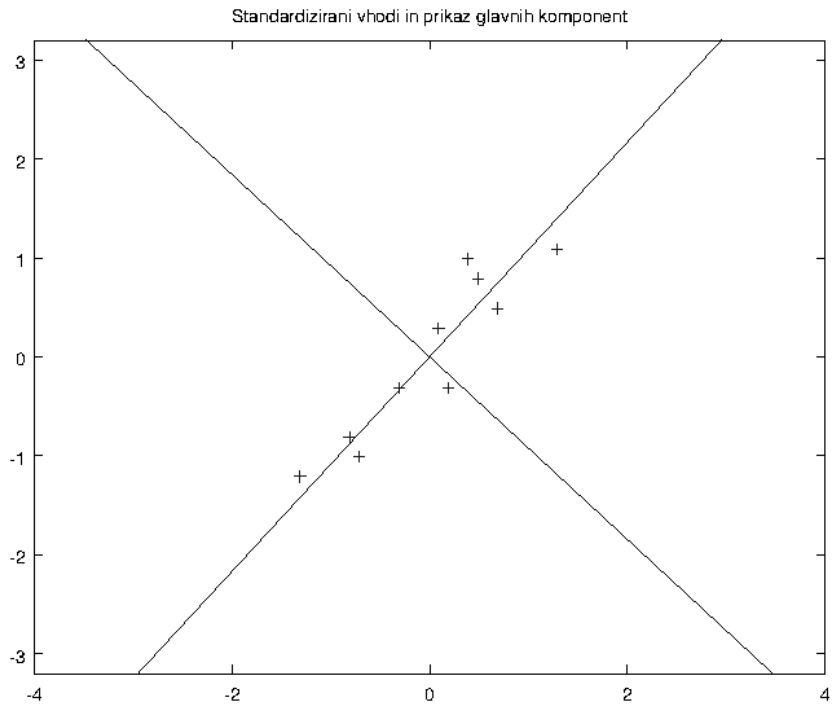
2.6.1 Analiza glavnih komponent (PCA)

Analizo glavnih komponent [34] bomo razložili na enostavnem primeru dveh dimenzij. Predpostavimo, da smo z meritvami zbrali vrednosti dveh spremenljivk. Podatke je potrebno najprej standardizirati tako, da vsaki dimenziji odštejemo povprečno vrednost podatkov v tej dimenziji, s čimer dobimo podatke, katerih povprečna vrednost je nič. Cilj analize glavnih komponent je najti komponente koordinatnega sistema, ki maksimizirajo varianco podatkov in minimizirajo rekonstrukcijsko napako, do katere pride pri neupoštevanju nekaterih dimenzij (redukcija). Kovariančno matriko podatkov izračunamo z izrazom (2.28), kjer $\hat{X}$ predstavlja standardizirano matriko podatkov.

$$C = \frac{1}{N}\hat{X}\hat{X}^T \tag{2.28}$$

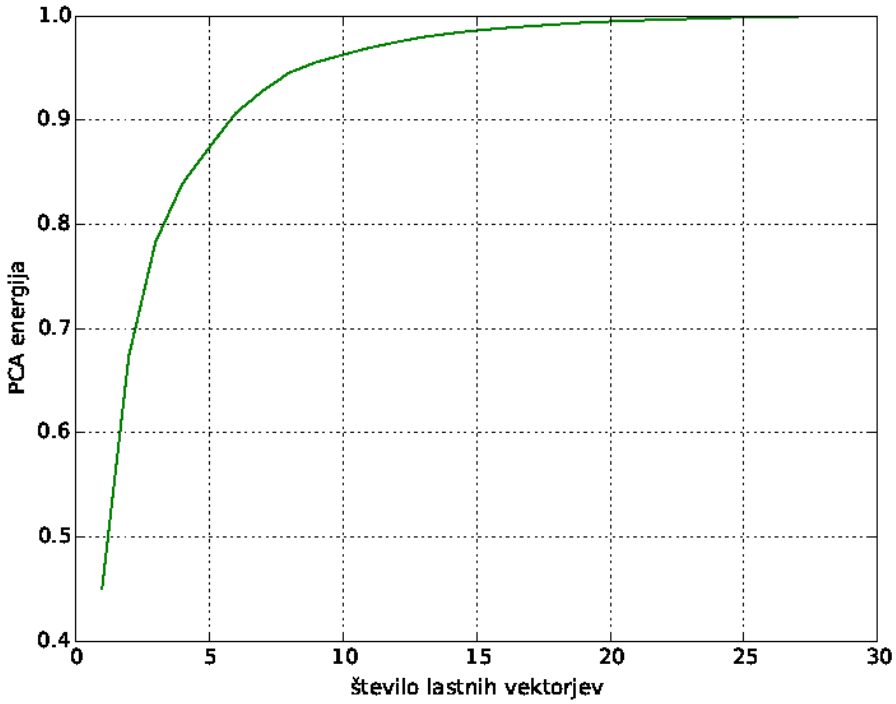
Z lastnimi vektorji in lastnimi vrednostmi kovariančne matrike C pridemo do komponent, ki maksimizirajo varianco v posameznih dimenzijah. Smerni vektor, ki maksimizira varianco, je lastni vektor z največjo lastno vredno-

stjo. Lastna vrednost predstavlja varianco podatkov v smeri pripadajočega lastnega vektorja. Lastna vektorja lahko vidimo na sliki 2.21. Lastne vrednosti in vektorje najhitreje dobimo z SVD razcepom, ki ga podpirajo vsa glavna matematična orodja. Tukaj lahko sedaj izpustimo komponente, ki imajo manjše lastne vrednosti, in tako zmanjšamo dimenzijo podatkov. Pri odločitvi, koliko lastnih vektorjev naj obdržimo, nam pomaga kumulativni graf lastnih vrednosti, ki nam pove, kolikšen delež variance opišemo z izbrano podmnožico lastnih vektorjev. Primer grafa je prikazan na sliki 2.22, kjer 90 odstotkov variance zajamemo že pri prvih šestih komponentah od triintrideset komponent opisnika FPFH.



Slika 2.21: Normalizirani podatki z lastnimi vektorji.

Ko smo izbrali komponente, preslikamo podatke v nov dimenzijski prostor z izrazom (2.29), kjer $\hat{U}$ predstavlja matriko izbranih lastnih vektorjev.



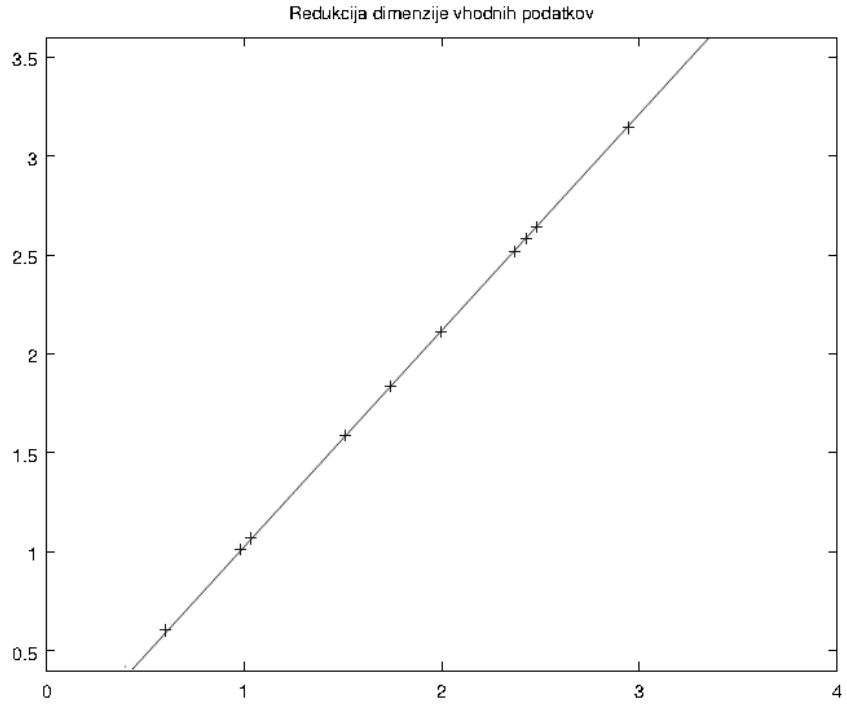
Slika 2.22: Kumulativni graf lastnih vrednosti na primeru opisnika FPFH.

Rezultat preslikave podatkov iz 2D v 1D lahko vidimo na sliki 2.23.

$$Y = \hat{U}^T \hat{X} \tag{2.29}$$

Podatke rekonstruiramo z izrazom (2.30), kar je uporabno pri njihovi kompresiji. V praktičnem delu diplomske naloge so predstavljeni rezultati uporabe PCA metode na opisnikih, opisanih v prejšnjem poglavju.

$$\tilde{X} = \hat{U}y + \mu \tag{2.30}$$



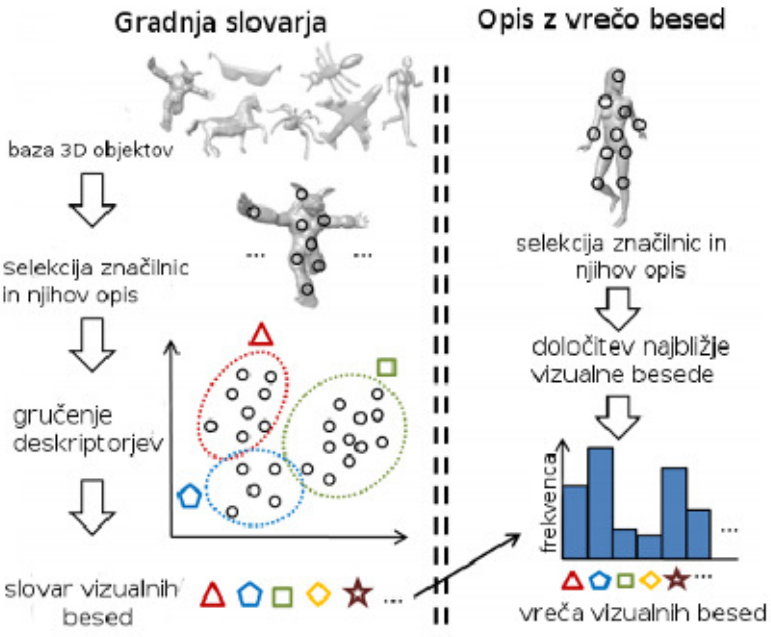
Slika 2.23: Redukcija dimenzije podatkov iz 2D v 1D.

2.7 Model vreče besed

Pri uporabi lokalnih opisnikov opišemo oblak točk z množico opisnikov, izračunanih v značilnih točkah oblaka. Za oblak točk tako dobimo množico opisnikov, ki jih je za klasifikacijo z metodami strojnega učenja potrebno združiti v en opisnik. Na področju računalniškega vida je za ta namen najbolj priljubljena metoda vreče vizualnih besed [25]. Model predstavlja oblak točk kot neurejeno zbirko lokalnih opisnikov. Metoda izhaja iz predstavitve tekstualnega dokumenta kot normaliziranega histograma števila posameznih besed, ki jih imamo v slovarju besed. Model, ki se uporablja v računalniškem vidu, je analogen temu, tako da slovar dobimo z gručenjem lokalnih opisnikov, ki jih dobimo iz učne množice oblakov točk. Lokalni opisniki so analogni besedam v dokumentu. Gručenje je potrebno, ker je lokalnih opisnikov

preveč in je množico potrebno diskretizirati. Vsaka gruča je samostojna vizualna beseda. Iz novega oblaka točk nato izračunamo lokalne opisnike in najdemo njihove najbližje vizualne besede. Končni opisnik nato predstavlja normaliziran histogram frekvence posameznih vizualnih besed. Proces na primeru oblakov točk ponazarja slika 2.24 in ga lahko strnemo v naslednjih treh točkah:

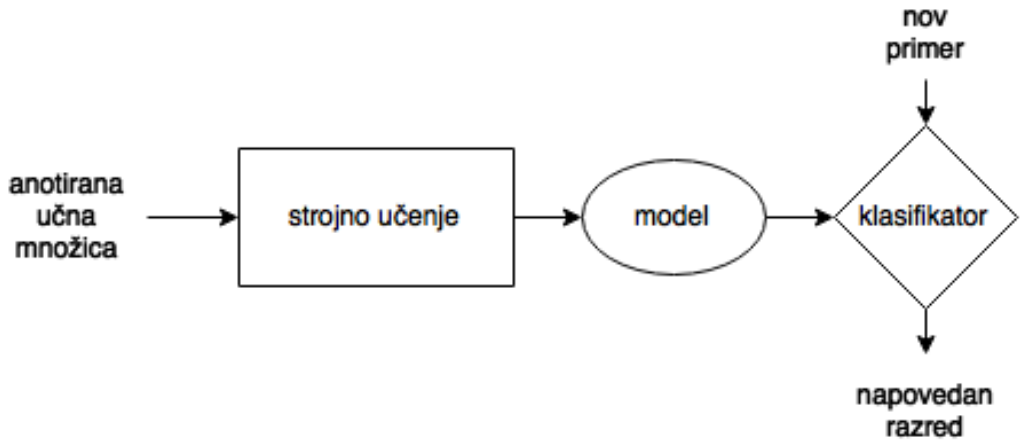
- *Gradnja slovarja:* Lokalne opisnike, pridobljene iz učne množice, gručimo. Vsaka gruča predstavlja vizualno besedo v slovarju.
- *Določitev besede:* Izračunamo lokalne opisnike in vsakemu določimo najbližjo besedo z npr. metodo najbližjih sosedov.
- *Histogram besed:* Za vsako vizualno besedo v slovarju štejemo število pojavitev in naredimo normaliziran histogram frekvenc vizualnih besed v slovarju. Dobimo predstavitev z vrečo besed.



Slika 2.24: Model vreče besed [22].

2.8 Klasifikacija

Iz zbranih predstavitev oblaka je potrebno določiti, katero kategorijo objekta predstavlja. Vsak oblak, ki ga opišemo z globalnim opisnikom, je predstavljen z enim večdimenzionalnim vektorjem. Tudi lokalne opisnike znamo sedaj združiti v en vektor z modelom vreče besed; na podlagi teh vektorjev določimo kategorijo objekta. Uporabimo metode strojnega učenja, ki v splošnem delujejo tako, kot prikazuje slika 2.25.



Slika 2.25: Postopek strojnega učenja.

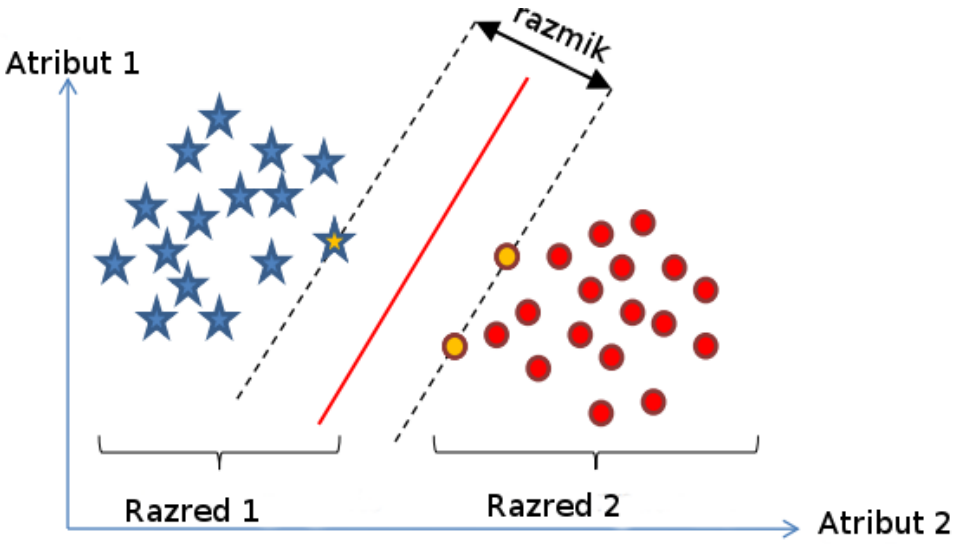
Slika 2.25 predstavlja nadzorovano strojno učenje, saj mora učna množica vsebovati tudi razrede, katerim pripada posamezen učni primer. Obstaja veliko metod strojnega učenja, med najbolj znanimi so:

- odločitvena drevesa,
- Bayesov klasifikator,
- metoda podpornih vektorjev,
- nevronske mreže.

V nadaljevanju je opisana metoda podpornih vektorjev, ki smo jo uporabili v diplomski nalogi.

2.8.1 Metoda podpornih vektorjev (SVM)

Metodo podpornih vektorjev [11] bomo opisali na primeru dveh atributov, kot to predstavlja slika 2.26. Cilj metode podpornih vektorjev je najti hiperravnino, ki ločuje razreda in ima največjo razdaljo do mejnih primerov razreda, ki jih imenujemo podporni vektorji. Iskanje ravnine se zastavi kot optimizacijski problem, minimizirati je potrebno izraz (2.31), in sicer pri pogoju (2.32). Levi del izraza (2.31) predstavlja razdaljo med podpornima vektorjema. Vpeljemo tudi kazensko spremenljivko ξ, saj ne moremo vedno idealno razmejiti pozitivnih in negativnih primerkov. C imenujemo posplošitveni faktor, saj z njim določamo, kako dobro želimo klasificirati učno množico. S preveliko vrednostjo C pride do prevelikega prilagajanja (ang. overfitting), s čimer izgubimo na splošnosti modela.

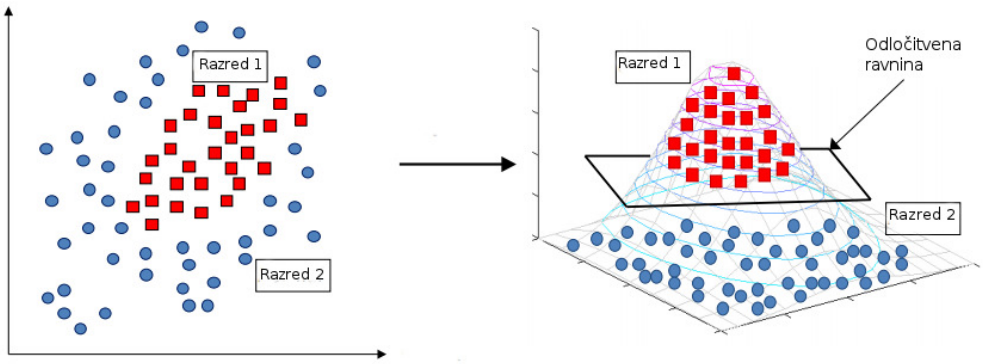


Slika 2.26: Metoda podpornih vektorjev [1].

$$\frac{1}{2}||w||^2 + C\sum_{i}^{n}\xi_i \tag{2.31}$$

$$y_i(w^T x + b) \geq 1 - \xi_i \tag{2.32}$$

Pomembna ideja metode podpornih vektorjev je transformacija atributnega prostora v kompleksnejši prostor. Dostikrat so primeri v originalnem prostoru linearno neločljivi in takrat se podatke preslika v nov, kompleksnejši atributni prostor, za katerega upamo, da postanejo podatki v njem linearno ločljivi. Postopek prikazuje slika 2.27. Eksplicitne transformacije podatkov iz prostora $\mathbb{R}^{\mathbb{N}}$ v prostor $\mathbb{R}^{\mathbb{M}}$, kjer velja , $M >> N$, niso praktične, saj je računska zahtevnost in poraba pomnilnika prevelika.



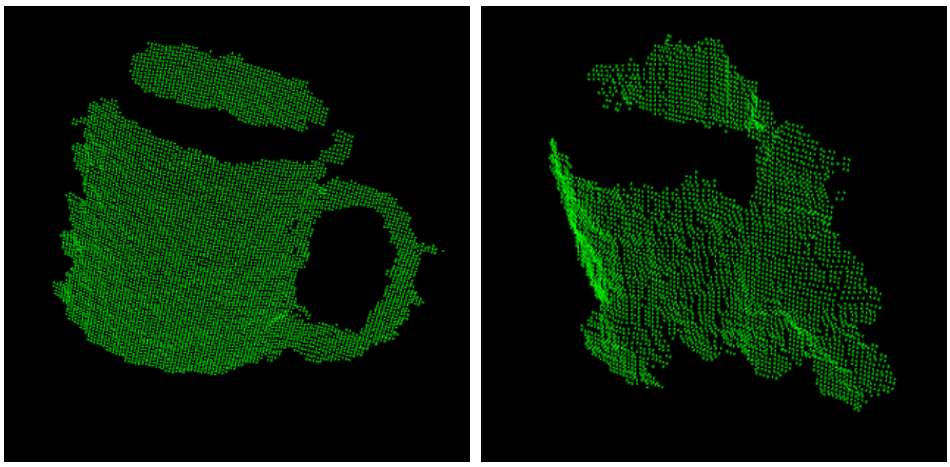
Slika 2.27: Primer preslikave linearno neločljivega primera v nov prostor, kjer postane linearno ločljiv [1].

Izkaže se, da eksplicitne transformacije podatkov v nov kompleksnejši prostor sploh ne potrebujemo. Pokažemo lahko [8], da potrebujemo zgolj skalarne produkte. Izkaže se tudi, da obstajajo posebne funkcije, imenovane jedrne funkcije, ki na podlagi danih vektorjev v $\mathbb{R}^{\mathbb{N}}$, izračunajo njune skalarne produkte v $\mathbb{R}^{\mathbb{M}}$, in sicer brez eksplicitne transformacije podatkov. Ta postopek imenujemo jedrni trik. S tem se izognemo tudi prekletstvu dimenzionalnosti.

Glavne jedrne funkcije, ki preslikajo atributni prostor v kompleksnjšega, so:

- linearna,
- polinomska,
- radialna,
- sigmoidna.

Na primeru skodelice 2.28a poteka klasifikacija tako, da zajamemo vse orientacije skodelice, s čimer zagotovimo invariantnost na orientacijo. Opisniki zajetih oblakov točk predstavljajo učno množico. Rezultat učenja je model, ki na novo zajeti in opisani objekt klasificira. Učno množico je sestavljalo 18 različnih objektov. Klasifikacijska točnost na učni množici je bila 99,3 %. Skodelico je klasificiralo pravilno, z verjetnostjo 72,7%. Drugi najbližji razred z verjetnostjo 19,3% je objekt na sliki 2.28b, ki je podoben skodelici.



(a) (b)

Slika 2.28: Skodelica in lonček za kaktus.

Poglavje 3

Implementacija sistema

Razvoj sistema za prepoznavanje objektov na osnovi oblakov točk smo razdelili na dva dela. V prvem delu smo prepoznavanje 3D objektov preizkusili na uveljavljeni bazi podatkov, v drugem delu pa smo implementirali realni sistem z napravo Kinect. V prvem delu smo testirali vse opisnike, ki smo jih predstavili v teoretičnem delu. Testirali smo tudi, kako na klasifikacijsko točnost vpliva redukcija dimenzionalnosti s PCA. Prepoznavanje smo realizirali na nivoju kategorije objektov. Uspešnost prepoznavanja kategorije objektov s 3D podatki smo primerjali s klasičnim prepoznavanjem v 2D in v ta namen razvili sistem za prepoznavanje v 2D. V nadaljevanju najprej predstavljamo uporabljene tehnologije, nato opišemo implementacijo sistema na bazi podatkov ter implementacijo sistema z napravo Kinect.

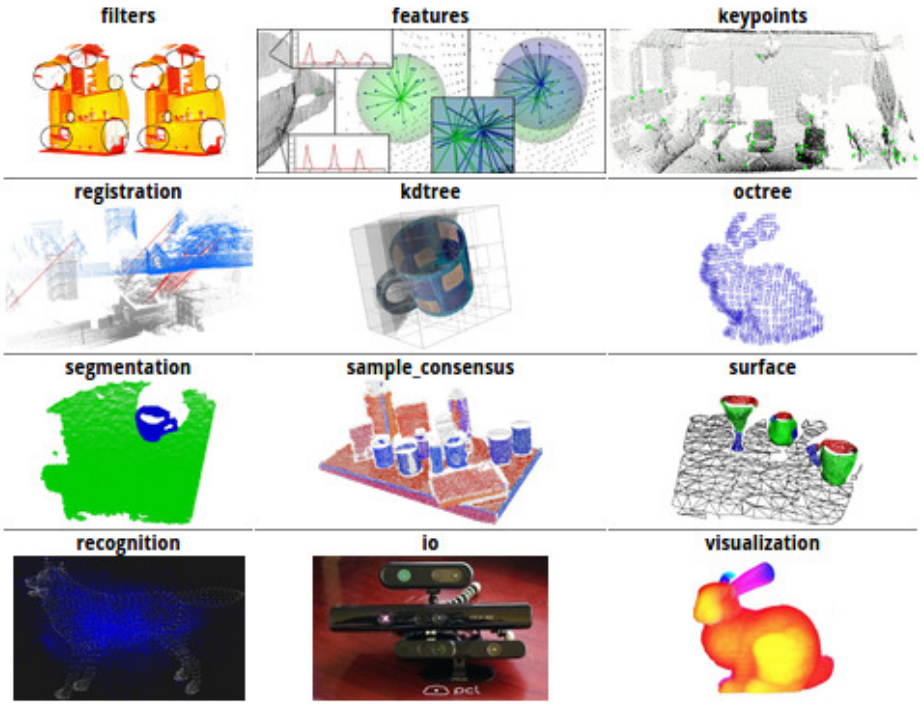
3.1 Uporabljene tehnologije

3.1.1 Point Cloud Library (PCL)

S prihodom poceni 3D zaznavnih naprav, kakršna je na primer Kinect, ki nam omogoča pridobivanje oblaka točk v realnem času že za 150 €, se je pričela razvijati skupnost, katere namen je bil razviti programsko ogrodje za enostavno in učinkovito upravljanje oblakov točk. Razvita je bila odprtokodna knjižnica Point Cloud Library (PCL) [31], ki je bila sprva del ROS (Robot Operating System), danes pa je povsem samostojna, razvita v C++. Od verzije 0.6 je PCL knjižnica podprta na vseh operacijskih sistemih, v delu pa je tudi verzija za mobilni operacijski sistem Android. Knjižnica podpira veliko operacij nad oblakom točk, npr. filtriranje, računanje značilk, računanje opisnikov, segmentacijo, registracijo, vizualizacijo.

Knjižnica je razdeljena na več osnovnih sklopov, ki jih vidimo na sliki 3.1. V diplomski nalogi smo jo potrebovali za obdelavo oblakov točk. Potrebovali smo predvsem naslednje module:

- *filtriranje (filters):* redukcija oblaka točk, odstranjevanje osamelcev;
- *značilke (keypoints):* iskanje značilk Harris3D;
- *opisniki (features):* računanje opisnikov (PFH, FPFH, Spin Images, VFH);
- *vhod/izhod (I/O):* nalaganje oblakov točk iz zbirke oblakov;
- *segmentacija (segmentation):* odstranjevanje ravnin in gručenje.



Slika 3.1: Moduli knjižnice PCL [5].

3.1.2 OpenCV

OpenCV [12] knjižnica implementira veliko metod računalniškega vida v 2D. Namenjena je enostavnejšemu in hitrejšemu programiranju aplikacij, ki uporabljajo metode računalniškega vida. Knjižnica implementira več kot 2500 optimiziranih algoritmov. V diplomski nalogi smo jo potrebovali pri izdelavi HSV histograma in za implementacijo sistema za prepoznavanje kategorije objektov v 2D, kjer smo s slik objektov našli značilne točke in jih opisali s SIFT opisnikom. Podobno kot pri 3D smo tudi tukaj uporabili model vreče besed, ki je že implementiran v OpenCV, ter linearni SVM klasifikator.

3.1.3 Programski jeziki in knjižnice za strojno učenje

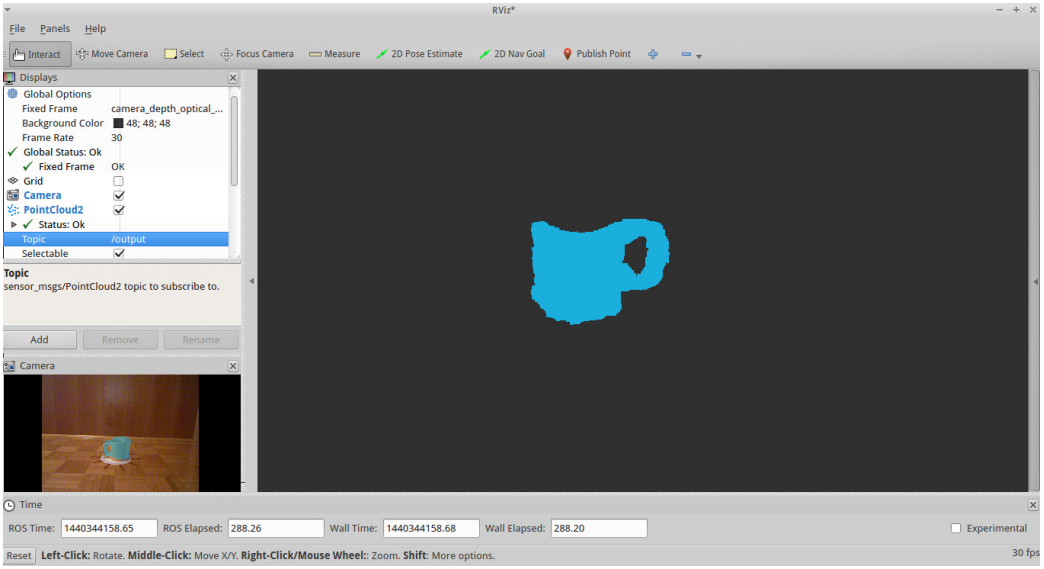
Strojno učenje smo implementirali v programskem jeziku Python, saj ima na voljo ogromno knjižnic, ki poenostavijo programiranje. Za izdelavo modela vreče besed smo uporabili gručenje K-means iz knjižnice scikit-learn [26], za učenje in predikcijo pa linearni SVM klasifikator iz knjižnice LIBSVM [14]. Za zajem in obdelavo oblakov točk smo uporabljali C++, saj je knjižnica PCL na voljo le v tem programskem jeziku. Metodo PCA smo implementirali v programskem jeziku Octave.

3.1.4 Robot Operating System (ROS)

Robot Operating System (ROS) [27] je programsko ogrodje, ki se uporablja predvsem v robotiki. Namen ROS-a je poenostaviti programiranje robotov z dekompozicijo programa na funkcionalne enote, ki med seboj komunicirajo. ROS tudi poenostavlja dostop do strojne opreme. Vsebuje programske konstrukte, kot so:

- *sporočila (messages):* za komunikacijo med posameznimi moduli se uporablja sporočila, ki jih lahko sami tudi definiramo;
- *teme (topics):* namenjene objavljanju sporočil;
- *storitve (services):* namenjene proženju oddaljenih metod.

V diplomski nalogi smo ROS uporabili za zajemanje oblaka točk ter njegovo vizualizacije v programu Rviz (glej sliko 3.2), ki je del ROS-a. ROS omogoča enostavno zajemanje oblaka točk iz naprave Kinect, saj se sistem le prijavi na temo, kamor sistem ROS objavlja oblak točk. Uporabljali smo tudi koncepte sporočil, s katerimi sistem pošilja opisnike v modul, ki opravlja klasifikacijo. Modul klasifikacije se proži s storitvijo. Sistem je podrobneje opisan v nadaljevanju, kjer predstavimo sistem za prepoznavanje 3D objektov v realnem času.



Slika 3.2: Vizualizacija oblaka točk skodelice v programu Rviz.

3.2 Prepoznavanje objektov v realnem času

Prepoznavanje 3D objektov smo realizirali tudi tako, da smo razvili sistem, ki omogoča gradnjo baze objektov, ki jo nato uporabimo za prepoznavanje. Cilj je bil naučiti sistem z vnaprej danimi objekti, ki jih želimo nato prepoznavati. Zbirke oblakov točk, kakršno smo uporabljali pri vrednotenju sistema na zbirki oblakov 4.1, so majhne, predvsem z vidika števila instanc posameznega objekta. Da bi zagotovili robustno prepoznavo kategorije objektov, bi potrebovali ogromno število instanc, saj drugače posplošitev ni možna. Velikokrat posplošitve niti ne potrebujemo, saj želimo prepoznavati točno določene objekte, ki so vnaprej znani in tako želimo doseči le robustno prepoznavanje teh objektov.

Sistem je razdeljen na dva dela: prvi omogoča gradnjo baze objektov ter učenje, drugi prepoznavanje v realnem času. Sistem (prikazuje ga slika 3.3a) smo razvili z ogrodjem ROS, s katerim smo programsko rešitev razdelili na tri glavne module:

- zajem oblaka točk in segmentacija objekta,
- izračun opisnika,
- prepoznava objekta.

(a) Sistem (b) Segmentirani objekt

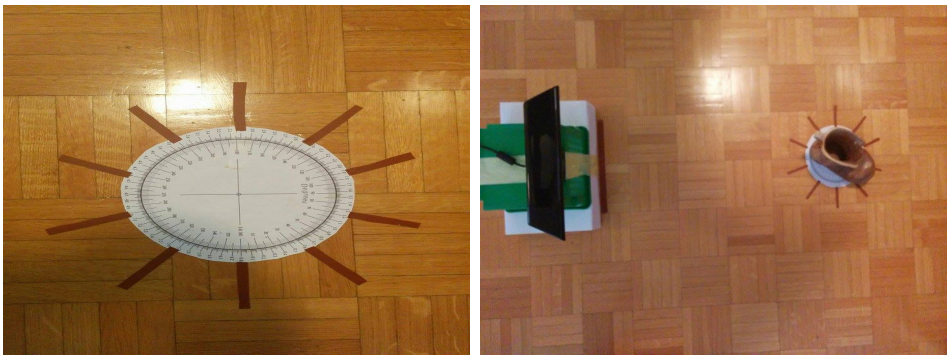
Slika 3.3: Sistem za prepoznavanje objektov.

Modul, ki omogoča zajem objekta, je dostopen prek storitve; ta je konstrukt ogrodja ROS, ki omogoča klic oddaljene metode. S klicem storitve modul zajame oblak točk in segmentira objekt, kot smo to opisali v 2.3. Rezultat prikazuje slika 3.3b

Uporabili smo globalni opisnik VFH, saj se je pri testiranju na zbirki izkazalo, da deluje podobno dobro kot lokalni opisniki. Izračun opisnika smo zopet izpostavili kot storitev. Segmentacija objekta je v realnem času vidna v okolju Rviz, ki omogoča prijavo na temo, kamor se objavlja segmentirani oblak točk. Ko je objekt dobro segmentiran, pokličemo storitev, ki izračuna globalni opisnik. Izračunani globalni opisnik se objavi v temo, na katero je prijavljen modul, ki omogoča prepoznavanje objektov. Ko se izračunani

opisnik objavi v temo, se v modulu za prepoznavo objekta sproži metoda, ki objavljeni opisnik klasificira.

Poseben modul je namenjen tudi shranjevanju segmentiranega oblaka točk, kar je potrebno pri gradnji baze. Učenje objekta smo realizirali tako, da smo objekt vrteli v krogu s korakom 36°, kot to prikazuje slika 3.4.



(a) (b)

Slika 3.4: Gradnja baze oblakov točk.

Bazo gradimo v enaki hierarhiji kot je sestavljena zbirka podatkov iz prejšnjega poglavja. Tako poteka učenje na isti način, potrebno je spremeniti le vhodno bazo oblakov točk.

Poglavje 4

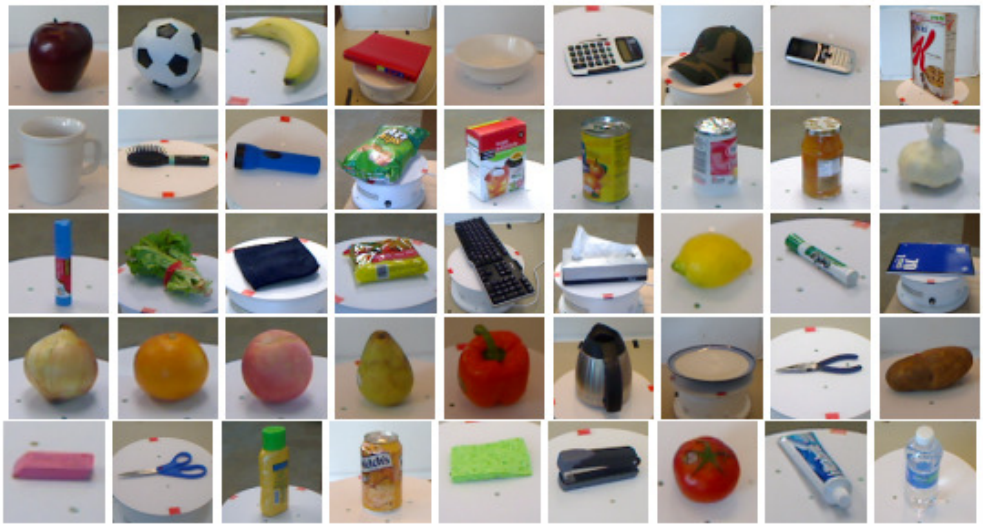
Rezultati

V tem poglavju so predstavljeni rezultati, ki smo jih dosegli na zbirki oblakov točk ter na sistemu, ki omogoča prepoznavanje objektov v realnem času. Predstavljena je tudi metodologija vrednotenja sistema.

4.1 Zbirka oblakov točk

Za vrednotenje sistema smo uporabili zbirko oblakov točk univerze v Washingtonu [21], ki vsebuje slikovne in globinske informacije o 300 različnih objektih, posnete pod različnimi pogledi. Zbirka je sestavljena iz predmetov, ki jih najdemo v vsakem domu in pisarni. Objekti so razvrščeni v 51 kategorij, ki jih v diplomski nalogi prepoznavamo. Vsako kategorijo sestavlja od 3 do 14 instanc objektov. Slika 4.1 prikazuje nekatere izmed kategorij objektov.

Zbirka je posneta z RGB-D napravo, podobno napravi Kinect, proizvajalca PrimeSense in klasično kamero proizvajalca Point Grey Research. RGB-D kamera simultano zajema tako barvno kot globinsko informacijo z ločljivostjo 640 x 480 slikovnih elementov. 3D lokacija posameznega slikovnega elementa se izračuna tako, kot smo to že opisali v teoretičnem delu 2.2.3, klasična kamera zajema RGB sliko z resolucijo 1600 x 1200 slikovnih elementov. Objekti so posneti na rotacijski mizici, ki se vrti s konstantno histrostjo na oddaljeno-



Slika 4.1: Nekaj izmed kategorij objektov v zbirki oblakov točk [21].

sti enega metra, in sicer pod različnimi koti: 30°, 45° in 60°. Zbirko sestavlja 250.000 RGB in RGBD slik. Na voljo ima tudi oblake točk v PCL formatu.

Vrednotenje sistema smo razdelili na tri dele. Oblake točk smo vedno vzorčili na približno 30°, in sicer na vseh pogledih, ki so v zbirki. Vsaka kategorija objektov vsebuje več instanc; pri vrednotenju smo eno instanco vsake kategorije vzeli za testno množico, preostale instance pa smo uporabili za učenje. Slika 4.2 prikazuje vse instance skodelice. Ovrednotili smo delovanje vseh opisnikov, ki smo jih predstavili v teoretičnem delu 2.5. Za učenje in klasifikacijo smo uporabili linearni SVM.

Slika 4.2: Vse instance skodelice v podatkovni zbirki [21].

4.1.1 Mali test

Pri malem testu smo uporabili naslednje kategorije objektov:

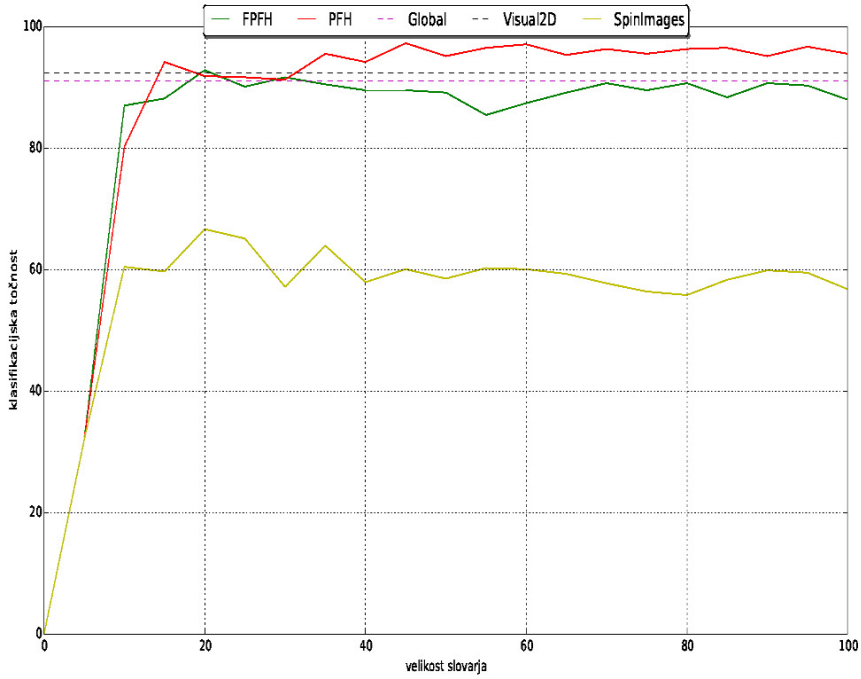
- žarnica,
- pločevinka,
- skodelica.

Slika 4.3: Objekti uporabljeni pri malem testu.

Objekti so prikazani na sliki 4.3. Pričakovali smo visoke klasifikacijske točnosti, saj so kategorije objektov zelo različne. Slika 4.4 prikazuje vpliv velikosti slovarja pri modelu vreče besed na klasifikacijsko točnost. Meritve smo opravili na intervalu [1,100], s korakom 5. Opravili smo tri meritve, rezultat smo povprečili. Klasifikacijske točnosti so zbrane v tabeli 4.1. Pri lokalnih opisnikih smo vzeli 100 naključnih točk, ki so enakomerno razporejene po celotnem objektu. Velikost slovarja v 2D smo nastavili na 1000, saj se je izkazalo, da so rezultati pri velikosti 100 bistveno slabši. Z vsake slike smo vzeli 100 najboljših značilnic, ki jih detektira SIFT, in jih z opisnikom SIFT tudi opisali.

Opisnik	Klasifikacijska točnost
PFH	97.3 %
FPFH	92.8 %
SpinImages	66.7 %
VFH	91 %
Visual2D	92.4 %

Tabela 4.1: Klasifikacijska točnost



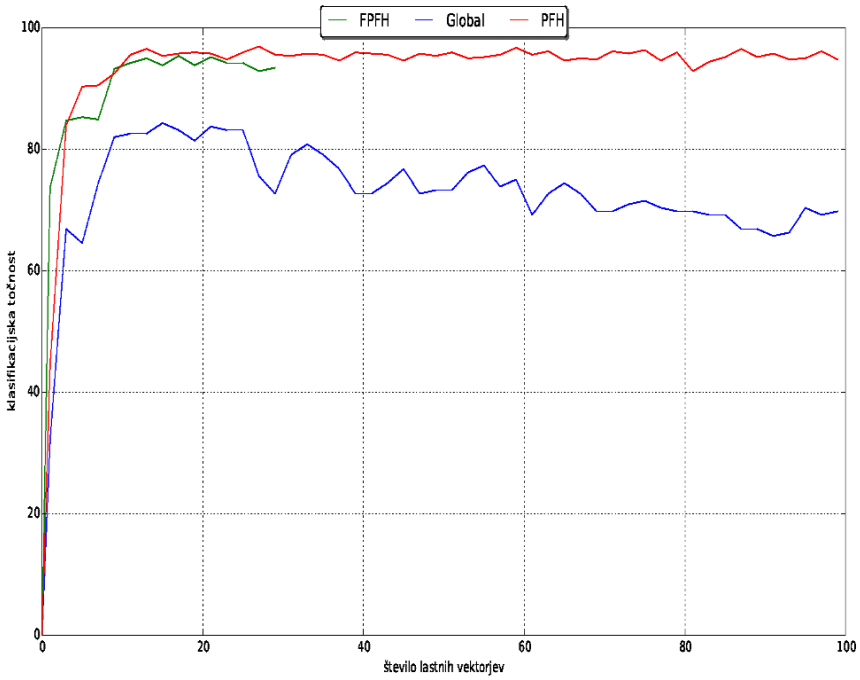
Slika 4.4: Vpliv velikosti slovarja vreče besed na klasifikacijsko točnost pri treh kategorijah.

Rezultati so pričakovani. Najbolje so se obnesli lokalni opisniki. Opazi se, da se FPFH obnese slabše od PFH. VFH globalni opisnik deluje podobno dobro kot FPFH, saj je, kot smo omenili v teoretičnem delu 2.5.4, iz njega tudi izpeljan. Spin slike se pričakovano obnesejo najslabše, kar je predvsem posledica šuma v podatkih. Prej omenjeni opisniki ta problem rešujejo. Pri prepoznavanju objektov na slikah v 2D nismo dosegli boljših rezultatov.

Pri velikosti slovarja vidimo, da se po velikosti 20 klasifikacijska točnost ne spreminja bistveno. Izjema je PFH, kjer se maksimalne vrednosti doseže pri velikosti okoli 45. Menimo, da je to posledica načina izračuna opisnika, saj moramo pri PFH lastnosti, ki smo jih omenili v teoretičnem delu 2.5.2 izračunati med vsemi pari točk v okolici točke, kjer računamo opisnik. Pri FPFH računamo te lastnosti le med neposrednimi sosedami. Vpliva tudi

velikost opisnika, saj ima FPFH 33, PFH pa 125 dimenzij.

Slika 4.5 prikazuje, kako na klasifikacijsko točnost vpliva redukcija dimenzionalnosti opisnikov. Meritve smo opravili na intervalu [1,100] s korakom 2, pri FPFH pa na intervalu [1,30] s korakom 2. Velikost slovarja pri lokalnih opisnikih je 50. Klasifikacijske točnosti, ki smo jih dosegli, so zbrane v tabeli 4.2.



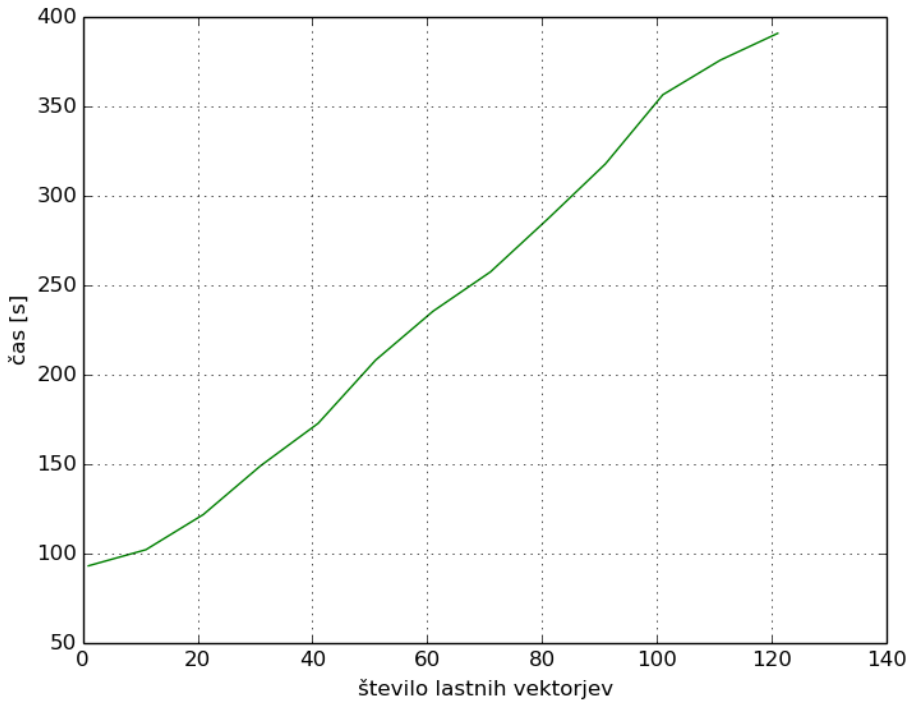
Slika 4.5: Vpliv števila lastnih vektorjev pri PCA na klasifikacijsko točnost pri treh kategorijah.

Na sliki 2.22 v teoretičnem delu je prikazan graf energije za FPFH, ki nam pove, da s prvimi šestimi lastnimi vektorji opišemo 90 % variance podatkov. Podobne vrednosti smo dobili tudi za PFH in VFH. Na grafu se to opazi tako, da klasifikacijska točnost najbolj raste ravno v tem intervalu, kasneje pa se klasifikacijska točnost ne dviguje tako hitro. Opazimo, da se redukcija dimenzionalnosti obnese, saj rezultati kljub zmanjšani dimenzionalnosti niso

Opisnik	Klasifikacijska točnost
PFH	96.9 %
FPFH	84.3 %
VFH	91 %

Tabela 4.2: Klasifikacijska točnost

dosti slabši. Redukcija dimenzionalnosti je posebej smiselna pri lokalnih opisnikih, saj tako bistveno pospešimo gradnjo slovarja pri modelu vreče besed. Graf 4.6 prikazuje vpliv števila lastnih vektorjev na čas učenja pri velikosti slovarja 50 in opisniku PFH.



Slika 4.6: Vpliv števila lastnih vektorjev pri PCA na čas učenja.

4.1.2 Veliki test

Za testiranje na večji množici objektov, smo vzeli naslednje kategorije objektov:

- banana
- česen,
- zobna pasta,
- skodelica,
- klešče,
- pločevinka,
- jogurt,
- šampon,
- ročna svetilka.

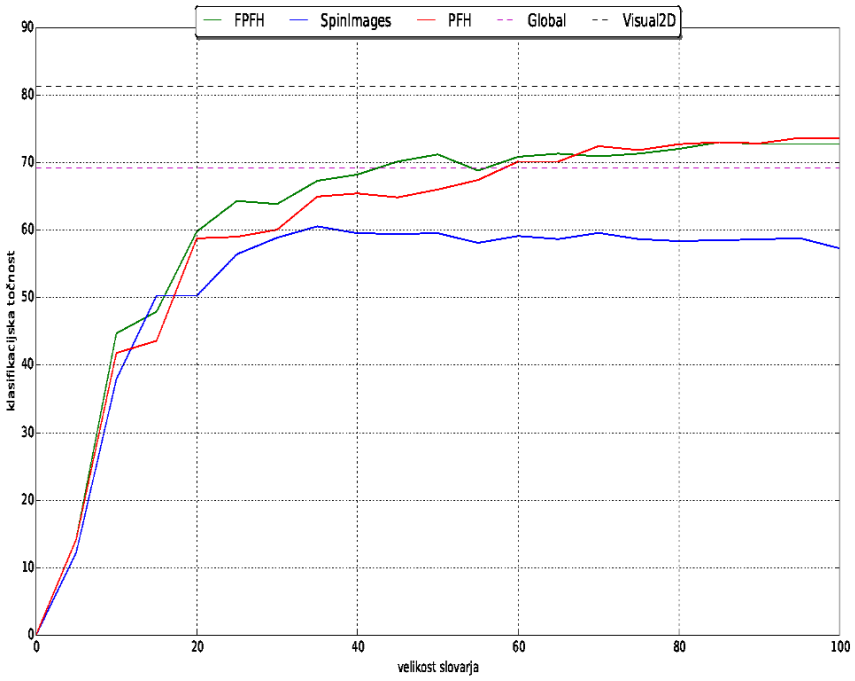
Slika 4.7: Objekti uporabljeni pri velikem testu.

Objekti so prikazani na sliki 4.7. Slika 4.8 prikazuje vpliv velikosti slovarja na klasifikacijsko točnost. Klasifikacijske točnosti so zbrane v tabeli 4.3. Metodologija vrednotenja je ista kot pri malem testu 4.1.1.

Rezultati so zopet pričakovani: podobno kot pri testiranju na treh kategorijah se tudi tu najbolje odrežejo lokalni opisniki; rezultati globalnega so nekoliko slabši. Pri prepoznavanju kategorije na slikah v 2D nam je uspelo doseči boljše rezultate, vendar pa je bilo potrebno velikost slovarja povečati na 1000, kar je zelo podaljšalo čas gradnje slovarja. Pri velikosti 100 so rezultati slabši.

Opisnik	Klasifikacijska točnost
PFH	73.7 %
FPFH	73 %
SpinImages	60.6 %
VFH	69.2 %
Visual2D	81.25 %

Tabela 4.3: Klasifikacijska točnost

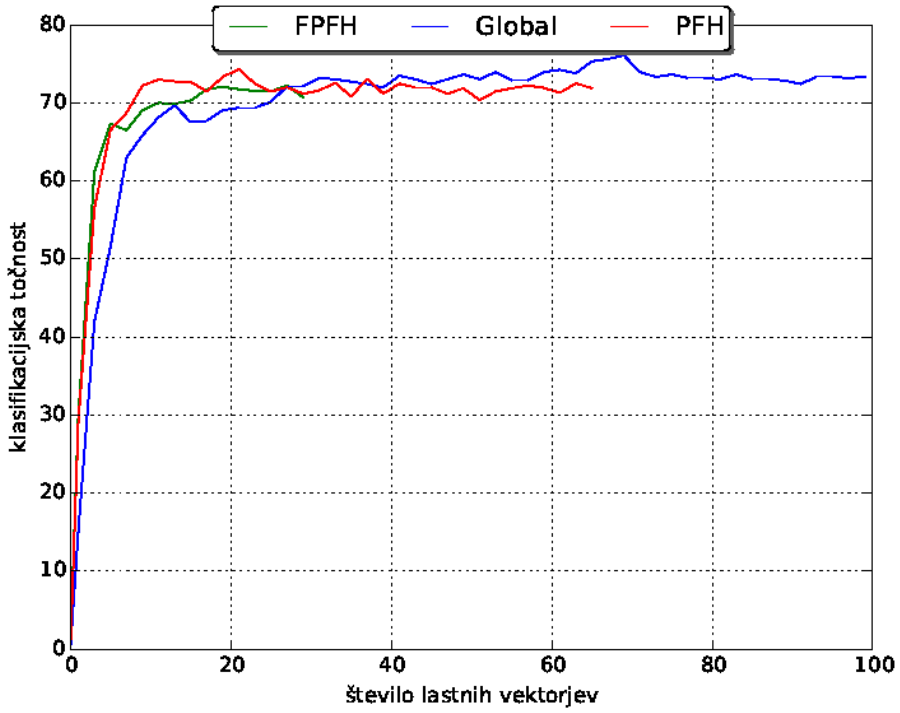


Slika 4.8: Vpliv velikosti slovarja vreče besed na klasifikacijsko točnost pri 9 kategorijah.

Opazi se, da mora biti velikost slovarja za doseganje najboljših rezultatov bistveno večja. Za FPFH mora biti velik okoli 45, pri PFH pa mora biti večji od 80.

Opisnik	Klasifikacijska točnost
PFH	74.3 %
FPFH	72.2 %
VFH	76.1 %

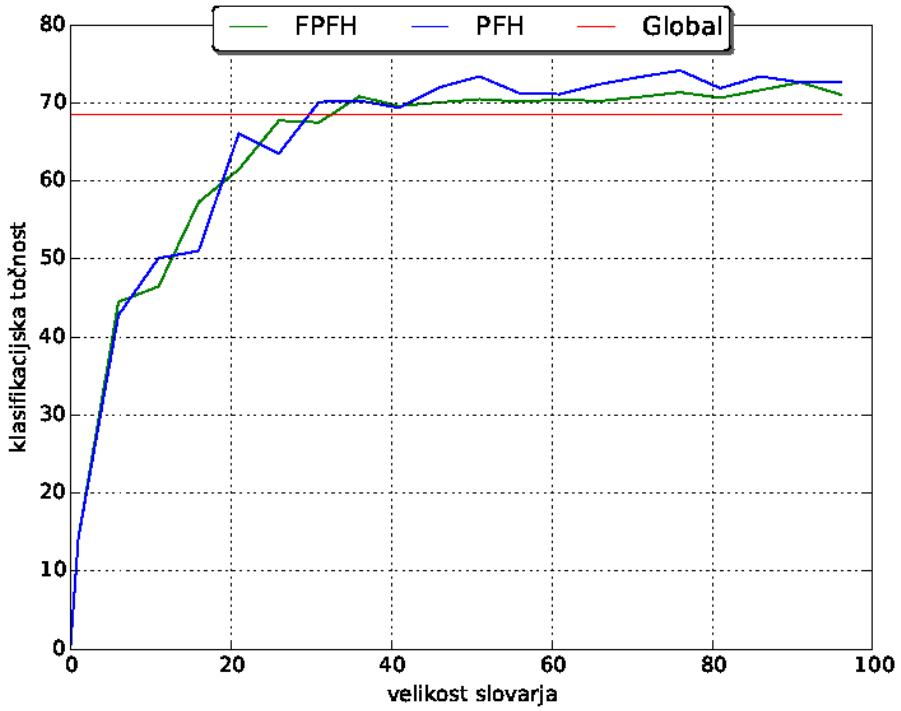
Tabela 4.4: Klasifikacijska točnost



Slika 4.9: Vpliv števila lastnih vektorjev pri PCA na klasifikacijsko točnost pri 9 kategorijah.

Slika 4.9 prikazuje, kako na klasifikacijsko točnost vpliva redukcija dimenzij. Klasifikacijske točnosti so zbrane v tabeli 4.4. Zanimiv je predvsem rezultat VFH opisnika, saj s PCA dosežemo precej boljše rezultate. Ugotovimo, da zaradi več kategorij potrebujemo tudi več lastnih vektorjev, da opišemo večino variance. Najmanj jih je potrebno pri PFH opisniku, za kar

je razlog predvsem v sestavi histograma opisnika. Pri PFH je interval za posamezno lastnost razdeljen na 5 delov. Ker imamo 3 lastnosti, je histogram velik 5^3. To si lahko predstavljamo kot $5 \times 5 \times 5$ kocko, kjer posamezna celica predstavlja točko z določenimi vrednostmi lastnosti. Prostor je tako polno koreliran in vsebuje veliko ničelnih vrednosti, kar prispeva k redundantnosti informacije. Pri FPFH je prostor dekoreliran tako, da je za vsako lastnost zgrajen poseben histogram, končni histogram pa je le konkatenacija teh histogramov.



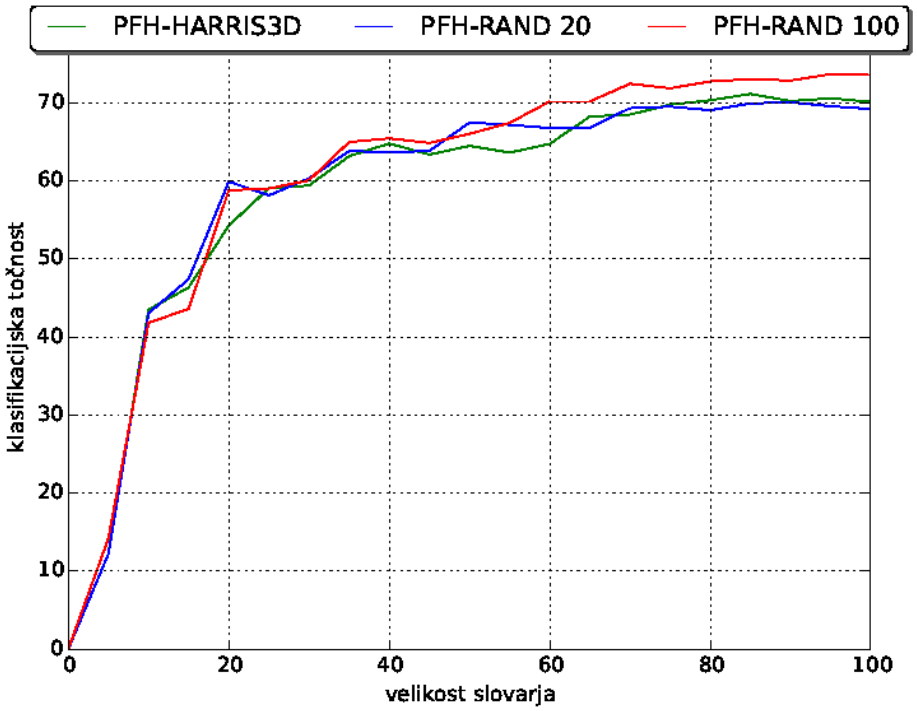
Slika 4.10: Vpliv velikosti slovarja vreče besed na klasifikacijsko točnost pri 9 kategorijah in PCA = 10.

Slika 4.10 prikazuje vpliv velikosti slovarja na klasifikacijsko točnost pri redukciji dimenzionalnosti opisnikov na 10. Rezultati, ki jih dosežemo, so zbrani v tabeli 4.5. Opazimo, da dosežemo praktično identične rezultate kot

Opisnik	Klasifikacijska točnost
PFH	74.1%
FPFH	72.6%
VFH	68.5%

Tabela 4.5: Klasifikacijska točnost

pri nereduciranih opisnikih. Opazimo tudi, da se je zmanjšala tudi potrebna velikost slovarja.



Slika 4.11: Primerjava značilnic Harris3D in naključnega izbiranja.

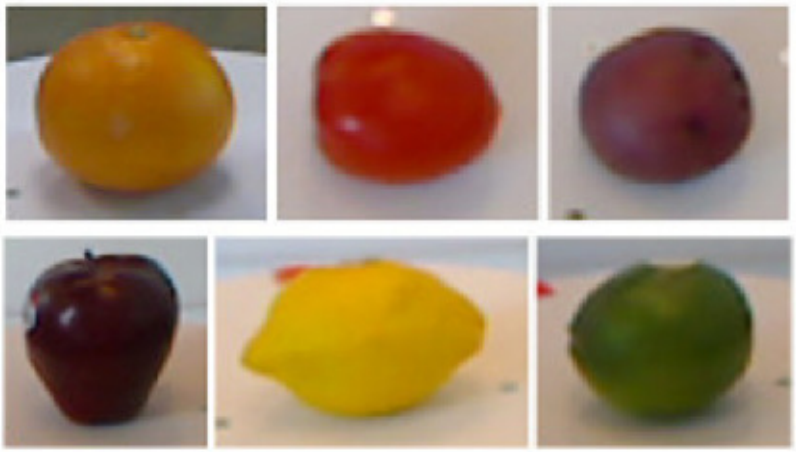
Na grafu 4.11 smo primerjali detektor značilnih točk Harris3D z enakomernim naključnim izbiranjem točk. Povprečno število, ki jih je detektor Harris3D izbral na oblaku točk, je bilo 20, zato smo detektor primerjali z naključnim izbiranjem 20 točk. Za primerjavo smo dodali tudi rezultate pri

naključnem izbiranju 100 točk. Opazimo, da so rezultati praktično identični in da značilnice, pridobljene s Harris3D, ne vplivajo bistveno na rezultat.

4.1.3 Podobni objekti

Delovanje smo testirali tudi na naslednji množici po obliki zelo podobnih objektov:

- jabolko,
- limona,
- grenivka,
- pomaranča,
- paradižnik,
- krompir.



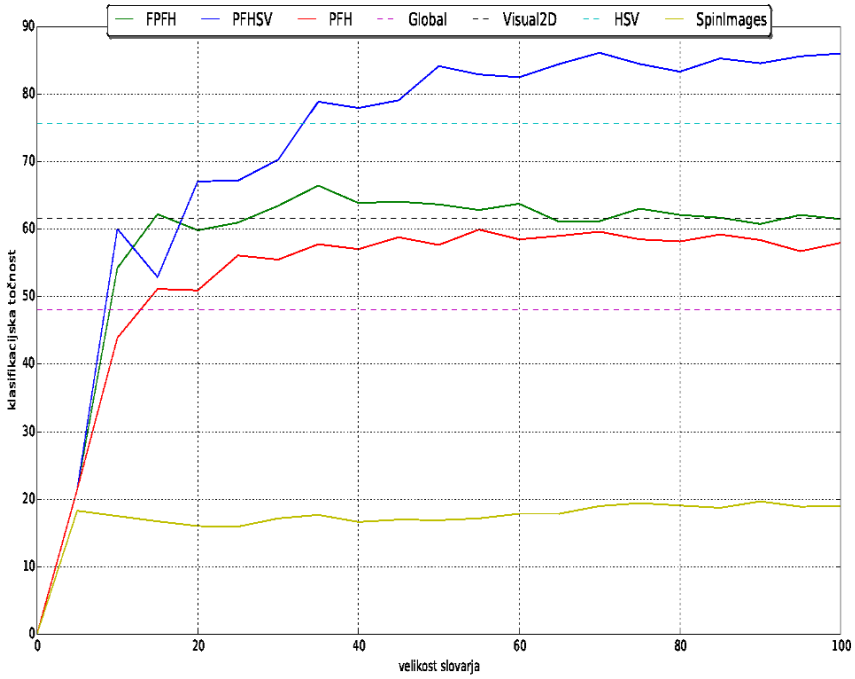
Slika 4.12: Objekti uporabljeni pri testu podobnih objektov.

Objekti, ki smo jih uporabili so prikazani na sliki 4.12. Zaradi oblikovnih podobnosti med objekti smo pričakovali slabe rezultate. Vpliv velikosti slovarja na klasifikacijsko točnost prikazuje slika 4.13. Metodologija vrednotenja je ista kot pri malem testu 4.1.1. Rezultati so zbrani v tabeli 4.6.

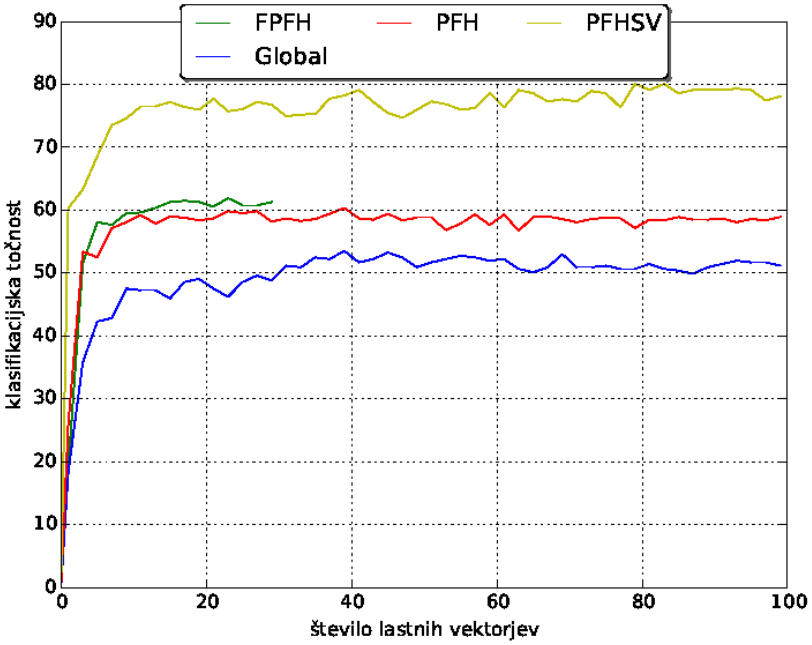
Slika 4.14 prikazuje vpliv števila lastnih vektorjev pri PCA na klasifikacijsko točnost. Potrebno je zelo malo lastnih vektorjev, saj je zaradi podobnosti oblakov v oblakih točk zelo malo variance. Pri večini opisnikov dosežemo 90 % variance že pri prvem lastnem vektorju.

Opisnik	Klasifikacijska točnost
PFH	59.9 %
FPFH	66.5 %
SpinImages	19.7 %
VFH	48 %
PFHSV	86.12 %
HSV	75.7 %
Visual2D	61.6 %

Tabela 4.6: Klasifikacijska točnost



Slika 4.13: Vpliv velikosti slovarja vreče besed na klasifikacijsko točnost pri podobnih objektih.

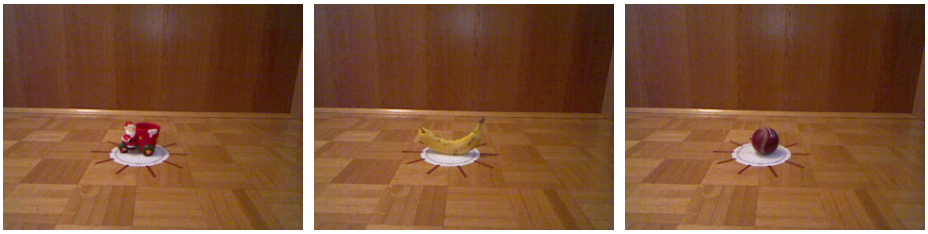


Slika 4.14: Vpliv števila lastnih vektorjev pri PCA na klasifikacijsko točnost pri podobnih objektih.

4.2 Prepoznavanje objektov v realnem času

Sistem smo najprej preizkusili na desetih objektih, ki jih prikazuje slika 4.15, njihove oblake točk pa slika 4.16. Zgradili smo bazo oblakov točk, nato pa smo vsakega izmed objektov poskušali ponovno prepoznati. Delovanje smo testirali tako, da smo objekt prepoznavali desetkrat, vsakič naključno orientiranega. Gradnja baze oblakov točk je opisana v 3.2.

Rezultate smo zbrali v prediktorski matriki na sliki 4.17. Iz matrike vidimo, da je dosežena klasifikacijska točnost 93 %. Lonček se je trikrat nepravilno klasificiral v skodelico, podobno tudi skodelica v lonček, kar je posledica njune podobnosti in premajhnega števila oblakov točk v bazi. Napačno se je klasificirala tudi mala svetilka, kar se je zgodilo, ko je bila posneta pod pravim kotom, tako da je bil ožji del obrnjen proti kameri. Tega položaja



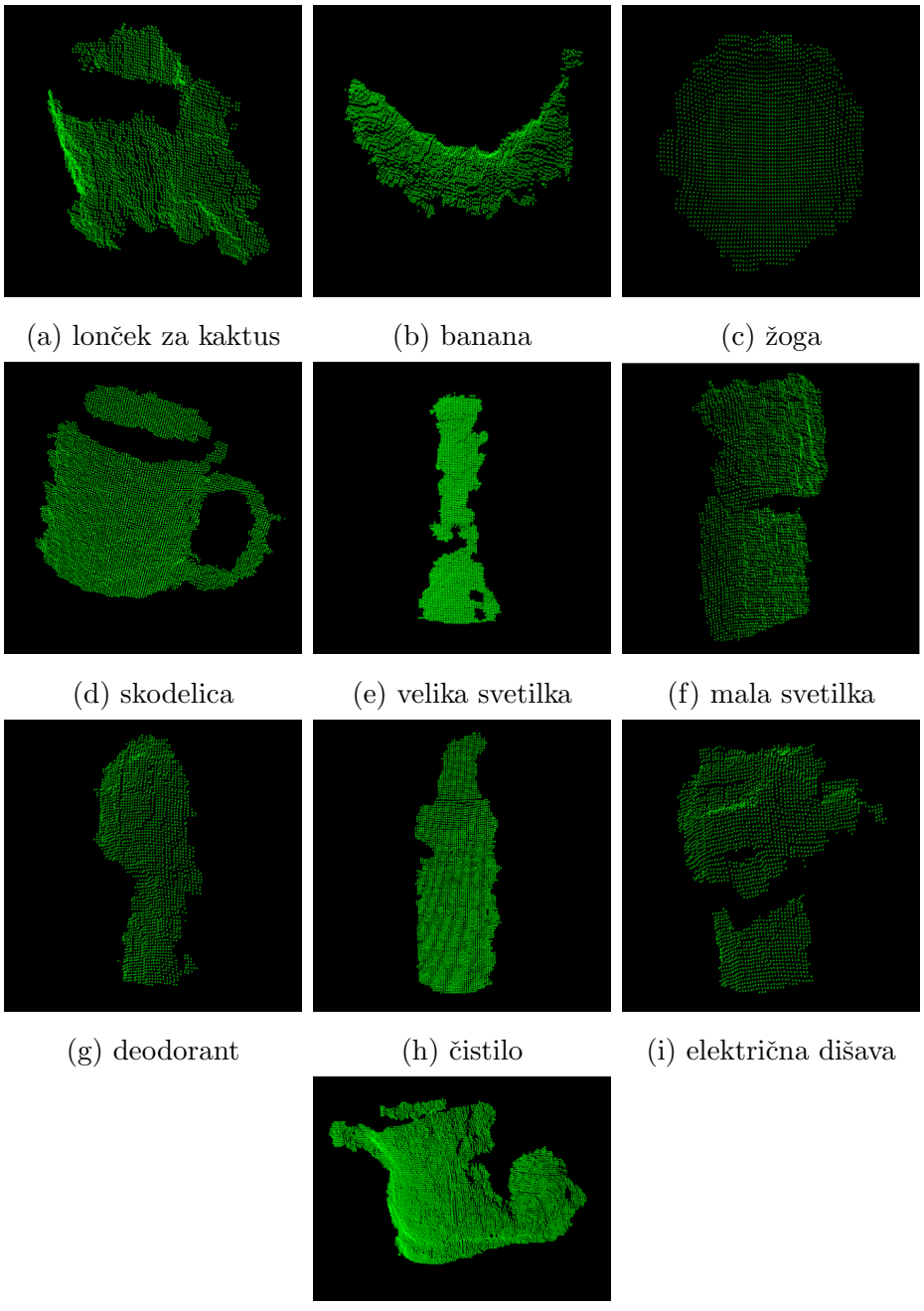
(a) lonček za kaktus (b) banana (c) žoga

(d) skodelica (e) velika svetilka (f) mala svetilka

(g) deodorant (h) čistilo (i) električna dišava

(j) okrasni čevelj

Slika 4.15: Prepoznavanje objektov (1).



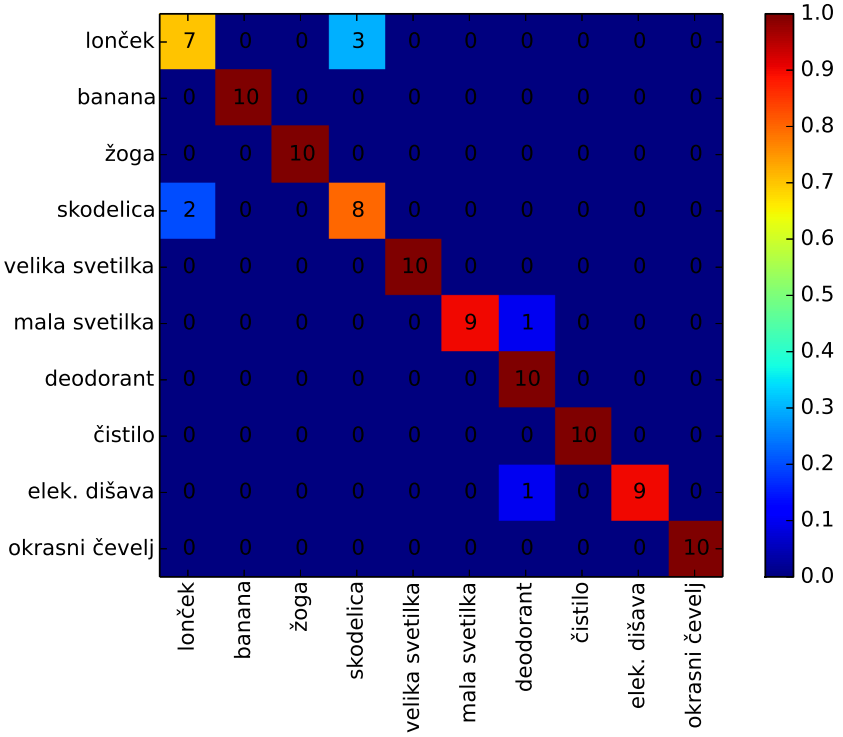
(a) lonček za kaktus (b) banana (c) žoga

(d) skodelica (e) velika svetilka (f) mala svetilka

(g) deodorant (h) čistilo (i) električna dišava

(j) okrasni čevelj

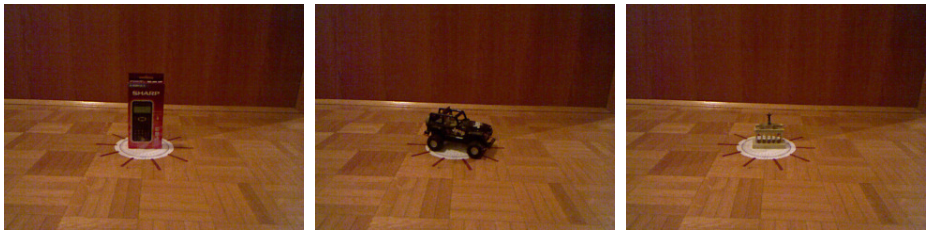
Slika 4.16: Prepoznavanje objektov - oblaki točk (1).



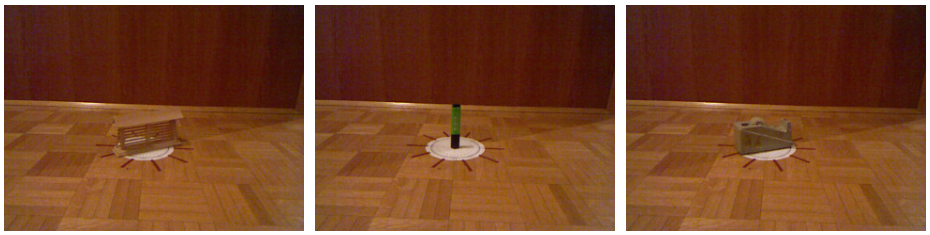
Slika 4.17: Prediktorska matrika (1).

nismo zajeli v bazi, zato se je klasificiral v najbližji objekt, to je deodorant. Višina deodoranta je popolnoma enaka mali svetilki.

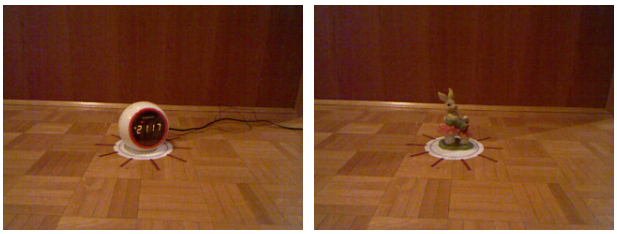
Pri zajemanju oblakov točk se pojavlja nekaj šuma (glej sliko 4.20), kar je posledica predvsem segmentacije objekta, zato je en oblak za posamezni pogled premalo za robustno prepoznavanje. Bazo objektov smo razširili z objekti na sliki 4.18 ter pri gradnji za vsak pogled naredili 100 oblakov točk, ki jih prikazuje slika 4.19. Korak zajemanja je ostal 36°. Rezultati so v prediktorski matriki 4.21. Klasifikacijska točnost je 97.2 %, kar je bistveno bolje. Napake se pojavljajo, če ima več objektov podoben pogled ali pa pogleda sploh nismo zajeli. Nekaterih napak se zaradi podobnosti in omejene informativnosti opisnikov ne da popraviti.



(a) embalaža (b) avtomobil (c) spominek

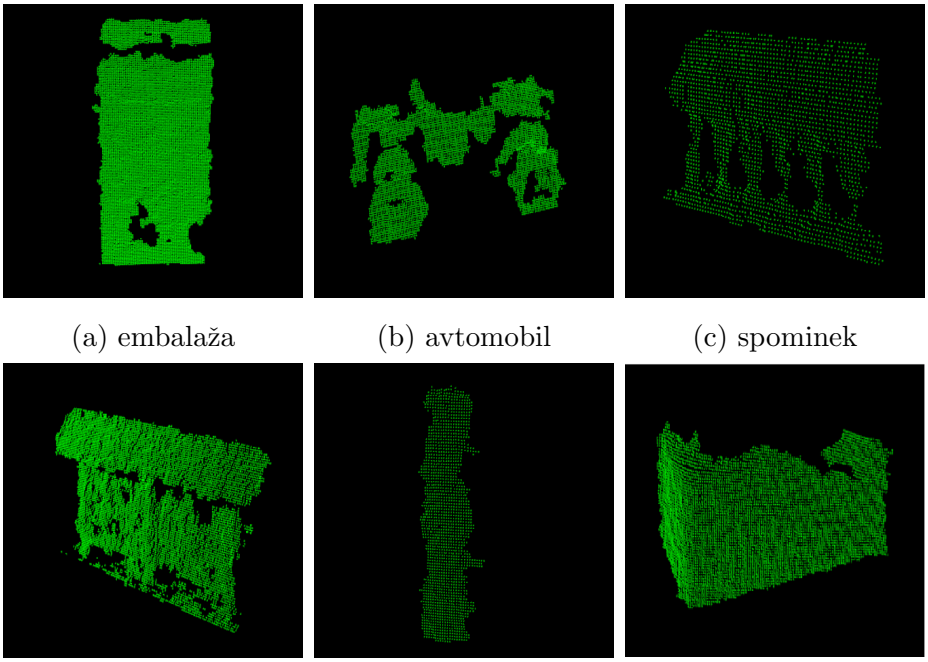


(d) kozolec (e) marker (f) lepilni trak



(g) budilka (h) okrasni zajec

Slika 4.18: Prepoznavanje objektov (2).



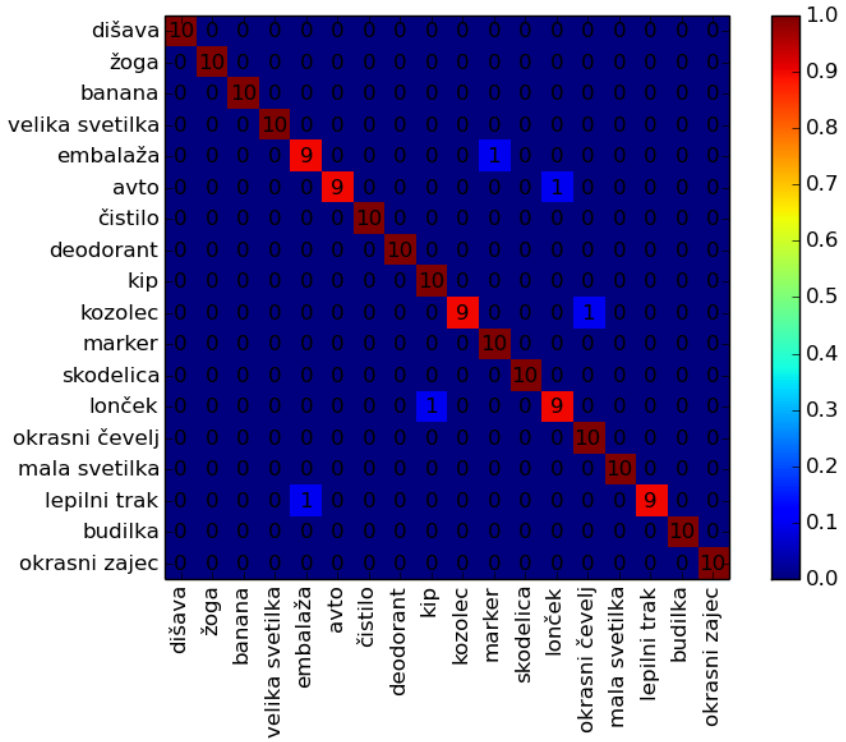
(a) embalaža (b) avtomobil (c) spominek

(d) kozolec (e) marker (f) lepilni trak

(g) budilka (h) okrasni zajec

Slika 4.19: Prepoznavanje objektov - oblaki točk (2).

Slika 4.20: Varianca pri zajemanju oblaka.



Slika 4.21: Prediktorska matrika (2).

Poglavje 5

Zaključek

V diplomskem delu smo se ukvarjali s prepoznavanjem objektov na osnovi oblakov točk. Sistem smo implementirali z različnimi najnovejšimi lokalnimi in globalnimi opisniki ter ga ovrednotili na uveljavljeni zbirki oblakov točk. Zaradi visoke dimenzionalnosti opisnikov nas je zanimalo, kako na klasifikacijsko točnost vpliva redukcija dimenzionalnosti z metodo glavnih komponent. Implementirali smo tudi prepoznavanje objektov iz slik ter tako dobili neposredno primerjavo. Na podlagi rezultatov, ki smo jih dobili pri testiranju na zbirki oblakov točk, smo izbrali najbolj optimalno metodo in jo tudi praktično implementirali. Razvili smo sistem za prepoznavo objektov na osnovi oblakov točk v realnem času. Ta omogoča gradnjo baze oblakov točk, ki jo nato uporabimo za učenje. Objekte smo klasificirali z metodo podpornih vektorjev.

Rezultati, ki smo jih dosegli, so pokazali, da je samo s 3D informacijo mogoče doseči dobre rezultate. Ugotovili smo, da je sistem posebej primeren, kadar imamo vnaprej znane objekte, ki jih želimo prepoznavati ter se jih lahko tako vnaprej naučimo. Sistem, ki smo ga uporabili za testiranje na zbirki oblakov točk smo implementirali tako, da smo za učenje uporabili različne instance iste kategorije objekta. Ugotovili smo, da so tako pridobljeni rezultati bistveno slabši, saj so baze oblakov točk premajhne za dobro posplošitev pri učenju. To lahko izpostavimo kot slabost, saj v primeru pre-

poznavanja objektov s slik ni težko pridobiti velike zbirke podatkov.

Glavna slabost sistema je, da je potrebno objekte iz oblaka točk najprej segmentirati. Tudi pri uporabi lokalnih opisnikov je v našem primeru potrebna predhodna segmentacija. Lokalne opisnike nato združujemo z metodo vreče besed. Dobra segmentacija objekta je zelo pomembna za uspešno prepoznavo. V diplomskem delu smo predpostavili, da prepoznavamo objekt na ravni površini tako, da smo lahko segmentirali ravnine. Pri praktičnih implementacijah to ni tako. Kot izboljšavo predlagamo uporabo algoritmov za gručenje korespondenc lokalnih opisnikov, kjer se izognemo segmentaciji.

Literatura

[1] A Gentle Introduction to Support Vector Machines in Biomedicine. http://www.med.nyu.edu/chibi/sites/default/files/chibi/Final.pdf/. (dostopano 9. 8. 2015).

[2] Domača stran podjetja Modri planet d.o.o. http://www.modriplanet.si//. (dostopano 9. 8. 2015).

[3] Evklidsko gručenje - PCL. http://www.pointclouds.org/documentation/tutorials/cluster_extraction.php. (dostopano 9. 8. 2015).

[4] Google avtonomno vozilo. http://www.google.com/selfdrivingcar/. (dostopano 9. 8. 2015).

[5] Moduli knjižnice PCL. http://www.pointclouds.org/documentation/tutorials/. (dostopano 11. 8. 2015).

[6] Oblak točk avtonomnega vozila. http://www.popsci.com/cars/article/2013-09/google-self-driving-car/. (dostopano 11. 8. 2015).

[7] Point Feature Histograms (PCL). http://pointclouds.org/documentation/tutorials/pfh_estimation.php/. (dostopano 9. 8. 2015).

[8] SVM jedrni trik. http://www.cs.berkeley.edu/~jordan/courses/281B-spring04/lectures/lec3.pdf. (dostopano 23. 8. 2015).

[9] A. Aldoma, Zoltan-Csaba Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R.B. Rusu, S. Gedikli, and M. Vincze. Tutorial: Point cloud library: Three-dimensional object recognition and 6 DOF pose estimation. *Robotics Automation Magazine, IEEE*, 19(3):80–91, Sept 2012.

[10] Luís A. Alexandre. 3d descriptors for object and category recognition: a comparative evaluation. In *in Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

[11] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 144–152, New York, NY, USA, 1992. ACM.

[12] G. Bradski. *Dr. Dobb's Journal of Software Tools*, 2000.

[13] Alexander Ceron and Flavio Prieto. Evaluating and comparing of 3d shape descriptors for object recognition. In George Bebis, Richard Boyle, Bahram Parvin, Darko Koracin, Baoxin Li, Fatih Porikli, Victor Zordan, James Klosowski, Sabine Coquillart, Xun Luo, Min Chen, and David Gotz, editors, *Advances in Visual Computing*, volume 8034 of *Lecture Notes in Computer Science*, pages 484–492. Springer Berlin Heidelberg, 2013.

[14] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[15] C.S. Dhir, N. Iqbal, and Soo-Young Lee. Efficient feature selection based on information gain criterion for face recognition. In *Information Acquisition, 2007. ICIA '07. International Conference on*, pages 523–527, July 2007.

[16] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[17] Chris Harris and Mike Stephens. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147–151, 1988.

[18] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[19] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 433–449, 1999.

[20] Kourosh Khoshelham and Sander Oude Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437, 2012.

[21] K. Lai, Liefeng Bo, Xiaofeng Ren, and D. Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1817–1824, May 2011.

[22] Guillaume Lavoué. Bag of words and local spectral descriptor for 3d partial shape retrieval. In *Eurographics Workshop on 3D Object Retrieval 2011, Llandudno, UK, April 10, 2011. Proceedings*, pages 41–48, 2011.

[23] D.G. Lowe. Object recognition from local scale-invariant features. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1150–1157 vol.2, 1999.

[24] M. Madry, C.H. Ek, R. Detry, Kaiyu Hang, and D. Kragic. Improving generalization for 3d object categorization with global structure histograms. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 1379–1386, Oct 2012.

[25] Stephen O'Hara and Bruce A. Draper. Introduction to the bag of features paradigm for image classification and retrieval. *CoRR*, abs/1101.3354, 2011.

[26] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[27] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[28] Marko Robnik-Šikonja and Igor Kononenko. Theoretical and empirical analysis of relieff and rrelieff. *Mach. Learn.*, 53(1-2):23–69, October 2003.

[29] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*, ICRA'09, pages 1848–1853, Piscataway, NJ, USA, 2009. IEEE Press.

[30] Radu Bogdan Rusu, Gary R. Bradski, Romain Thibaux, and John M. Hsu. Fast 3d recognition and pose using the viewpoint feature histogram. In *IROS*, pages 2155–2162. IEEE, 2010.

[31] Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *ICRA*. IEEE, 2011.

[32] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, and Michael Beetz. Persistent Point Feature Histograms for 3D Point Clouds. In *Proceedings of the 10th International Conference on Intelligent Autonomous Systems (IAS-10), Baden-Baden, Germany*, 2008.

[33] S. Salti, F. Tombari, and L. Di Stefano. A performance evaluation of 3d keypoint detectors. In *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2011 International Conference on*, pages 236–243, May 2011.

[34] Jonathon Shlens. A tutorial on principal component analysis. In *Systems Neurobiology Laboratory, Salk Institute for Biological Studies*, 2005.

[35] Jain Siddarth, Matsuda Nathan, and Yang Yiyang. Learnng descriptors for 3D Object Recognition and Classification from Point Clouds. `https://sites.google.com/site/vfhclassification/home`. (dostopano 9. 8. 2015).

[36] I. Sipiran and Benjamin Bustos. Harris 3D: A robust extension of the Harris operator for interest point detection on 3D meshes. *The Visual Computer*, 11(27):963–976, 2011.

[37] Fengxi Song, Dayong Mei, and Hongfeng Li. Feature selection based on linear discriminant analysis. In *Intelligent System Design and Engineering Application (ISDEA), 2010 International Conference on*, volume 1, pages 746–749, Oct 2010.