# Segmentacija dubinske slike metodom lokalno konveksnih povezanih segmenata

**Ljepić, Srđan**



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

**Sveučilišni studij**

# SEGMENTACIJA DUBINSKE SLIKE METODOM LOKALNO KONVEKSNIH POVEZANIH SEGMENATA

**Završni rad**

**Srđan Ljepić**

**Osijek, 2017.**

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

**Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju**

**Osijek, 07.09.2017.**

**Odboru za završne i diplomske ispite**

# Prijedlog ocjene završnog rada

| | |
|---|---|
| **Ime i prezime studenta:** | Srđan Ljepić |
| **Studij, smjer:** | Preddiplomski sveučilišni studij Računarstvo |
| **Mat. br. studenta, godina upisa:** | R3666, 23.07.2014. |
| **OIB studenta:** | 15697455827 |
| **Mentor:** | Prof.dr.sc. Robert Cupec |
| **Sumentor:** | |
| **Sumentor iz tvrtke:** | |
| **Naslov završnog rada:** | Segmentacija dubinske slike metodom lokalno konveksnih povezanih segmenata |
| **Znanstvena grana rada:** | **Umjetna inteligencija (zn. polje računarstvo)** |
| **Predložena ocjena završnog rada:** | Izvrstan (5) |
| **Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:** | Primjena znanja stečenih na fakultetu: 3 bod/boda<br>Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda<br>Jasnoća pismenog izražavanja: 3 bod/boda<br>Razina samostalnosti: 3 razina |
| **Datum prijedloga ocjene mentora:** | 07.09.2017. |
| **Datum potvrde ocjene Odbora:** | 11.09.2017. |

| | |
|---|---|
| Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija: | Potpis: |
| | Datum: |

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

# IZJAVA O ORIGINALNOSTI RADA

**Osijek, 12.09.2017.**

| | |
|---|---|
| **Ime i prezime studenta:** | Srđan Ljepić |
| **Studij**: | Preddiplomski sveučilišni studij Računarstvo |
| **Mat. br. studenta, godina upisa:** | R3666, 23.07.2014. |
| **Ephorus podudaranje [%]:** | 12% |

Ovom izjavom izjavljujem da je rad pod nazivom**: Segmentacija dubinske slike metodom lokalno konveksnih povezanih segmenata**

izrađen pod vodstvom mentora Prof.dr.sc. Robert Cupec

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

# Sadržaj

# 1. UVOD

Jedan od najvažnijih zadataka današnje robotike je segmentacija slike. Da bi robot mogao uspješno manipulirati objektima u njegovom okruženju potrebno je da robot razlikuje te objekte. Ljudska percepcija omogućava trenutnu segmentaciju određene situacije, ali cilj je tu mogućnost implementirati u robotske sustave da bi se uz ostale procese stvorio funkcionalan robotski sustav. Metoda lokalno konveksnih povezanih segmenata omogućava segmentaciju dubinske slike na različite objekte zahvaljujući svojstvu konveksnosti objekata koji su odijeljeni konkavnim granicama. U ovom završnom radu obrađena je implementacija algoritma lokalno konveksnih povezanih segmenata koristeći programsko okruženje Microsoft Visual Studio 2103 uz Point Cloud Library (PCL) biblioteku gdje je cilj ispitati i utvrditi učinkovitost spomenutog algoritma u različitim situacijama. Navedena metoda je ispitana na skupu dubinskih slika koje su dobivene RGB-D kamerom i koje sadrže različiti broj predmeta i različiti položaj predmeta. Ispitana je osobina algoritma da uz jednostavnost i brzo izvođenje postiže rezultate usporedive sa znatno složenijim *state-of-the-art* algoritmima za segmentaciju slike.

U drugom poglavlju teorijski je obrađen robotski vid, ali i sam proces segmentacije koji je vezan uz tu granu računarstva. Treće poglavlje sadržava teorijsku podlogu u kojoj su objašnjeni postupci koji čine proces metode lokalno konveksnih povezanih segmenata. Četvrto poglavlje predstavlja programsku podršku potrebnu za implementaciju metode lokalno konveksnih povezanih segmenata te ukratko objašnjen korišteni kod algoritma. Rezultati su prezentirani u petom poglavlju ovoga rada.

## 1.1. Zadatak završnog rada

Zadatak završnog rada je teorijski proučiti metodu lokalno konveksnih povezanih segmenata kojom se obavlja segmentacija dubinske slike. Algoritam temeljen na spomenutoj metodi potrebno je implementirati u programu Microsoft Visual Studio te rezultate segmentacije usporediti s izvornim slikama. Konačno, potrebno je donjeti zaključke o učinkovitosti spomenutog algoritma u procesu segmentacije dubinske slike.

# 2. ROBOTSKI VID

Robotski vid je grana umjetne inteligencije koja omogućava oponašanje ljudskog osjetila vida zahvaljujući senzorima kojima su roboti opremljeni. Sustav robotskog vida sastoji se od percepcijskog senzora i programske podrške za obradu slike. Zahvaljujući odgovarajućim algoritmima ispunjavaju se osnovne zadaće robotskog sustava kao što su: prepoznavanje objekata u okolini robota, određivanje položaja objekta u odnosu na položaj robota, prepoznavanje prepreka u okolini robota i druge zadaće robotskog sustava. Zbog toga robotski vid igra važnu ulogu kod procesa dohvaćanja informacija iz okoline koje se mogu iskoristiti za kretanje robota i manipulaciju objektima. Percepcijski senzori mjere neku karakteristiku okruženja u kojem se robot kreće. Najčešće detektiraju prepreke i objekte beskontaktno tako što za dohvaćanje informacija najčešće koriste infracrveno ili neko drugo zračenje, zbog čega se zovu i radijacijski senzori.

## 2.1 RGB-D kamera

Za dobijanje slika korištenih kao ulazne varijable u algoritmu lokalno konveksnih povezanih segmenata koristi se RGB-D kamera. Korištenjem RGB-D kamere dobijaju se dubinske slike iz kojih je moguće dobiti dvodimenzionalno polje trodimenzionalnih točaka pomoću odgovarajućeg programa za rekonstrukciju. Jedan od modela RGB-D kamera nalazi se na slici 2.1..


Slika 2.1. Microsoft Kinect senzor (Slika preuzeta iz [5])

U trodimenzionalnoj grafici i računalnom vidu važan pojam je voksel koji predstavlja najmanji dio trodimenzionalnog prostora neke scene koji se može obrađivati ili prikazati. Zbog toga je moguće mijenjati svojstva voksela kao što su boja ili neke druge osobine. Predstavljaju analogiju

pikselima u dvodimenzionalnim slikama, a položaj određuju prema relativnim koordinatama u odnosu na druge voksele. Osim voksela postoje i supervokseli koji su veći od voksela ali imaju jednaku boju i ostala svojstva. Oblak točaka je podatkovna struktura koja se koristi za predstavljanje niza multidimenzionalnih točaka, najčešće koristeći trodimenzionalni koordinatni sustav radi vizualizacije trodimenzionalnih scena. [3]

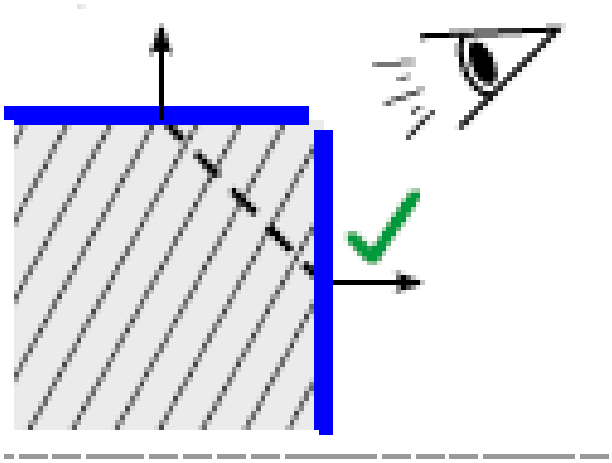
## 2.2. Segmentacija

Segmentacija je proces rastavljanja scene, odnosno slike, u njezine sastavne dijelove ili segmente. Obavlja se tako da se pikseli slike grupiraju u disjunktne podskupove, gdje svaki podskup, odnosno segment, predstavlja objekt od interesa, dok ostali predstavljaju pozadinu ili ostale objekte. Osnovne vrste segmentacije su: segmentacija na objekte od interesa i pozadinu, segmentacija na objekte i segmentacija na primitive. Na osnovi različitih karakteristika objekata, segmenata, ali i samih slika postoje različite tehnike segmentacije kao što su na primjer: segmentacija na temelju boje, segmentacija na temelju kontrasta, segmentacija na temelju teksture, segmentacija pomoću informacije o dubini. Algoritam lokalno konveksnih povezanih segmenata odgovara segmentaciji na primitive. Segmentacija na primitive bavi se podjelom slike na osnovu geometrijske strukture pa se time segmentiraju ravnine, cilindri, sfere, konveksne površine i ostale strukture. Nakon što se kreiraju grafovi konveksnosti koristi se segmentacija širenja područja te se zahvaljujući njoj omogućava povezivanje supervoksela u segmente koji predstavljaju objekte na slici. Segmentacija širenjem područja izvodi se tako da se određuju uniformni skupovi točaka slike koji se predstavljaju kao segmenti. Algoritmi širenja područja kreću od korijenske točke te se uspoređuju sa susjednim segmentiranim regijama te se odlučuje pripada li ta točka segmentu. Ukoliko se odluči da točka pripada, ona se pridružuje odabranom segmentu te se taj postupak ponavlja koristeći drugu dotad neprocesuiranu točku sve dok ima slobodnih točki.

# 3. METODA LOKALNO KONVEKSNIH POVEZANIH SEGMENATA

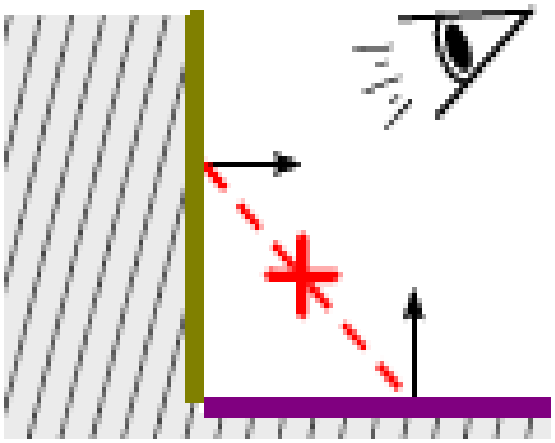
Metoda lokalno koveksnih povezanih segmenata temelji se na načelu koje govori da je moguće percepcirati objekte zahvaljujući povezanim konveksnim površinama koje su razdvojene konkavnim granicama. Zahvaljujući toj mogućnosti izvodi se segmentacija multidimenzionalnih oblaka točaka na različite objekte tako što se povezuju konveksni segmenti koje razdvajaju konkavni segmenti.

Konveksnost se definira tako da ukoliko postoje dvije površine određenog objekta takve da svaka točka pravca koji povezuje bilo koju točku jedne površine s bilo kojom drugom točkom druge površine se nalazi unutar samog objekta onda kažemo da su te dvije površine, odnosno točke konveksne. Grafički prikaz načela konveksnosti se može vidjeti na slici 3.1..



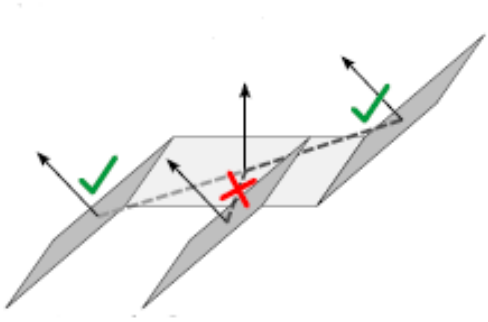
Sl. 3.1. Konveksne površine (Slika preuzeta iz [1])

Analogno konveksnosti može se definirati i i konkavnost. Ukoliko postoje dvije površine određenog objekta takve da svaka točka pravca koja spaja bilo koju točku jedne površine i bilo koju točku druge površine prolazi kroz „prazan prostor“, odnosno ne prolazi kroz površinu objekta onda za njih kažemo da su konkavne. Grafički prikaz konkavnosti nalazi se na slici 3.2..



Sl. 3.2. Konkavne površine (Slika preuzeta iz [1])

Osim konkavnih i konveksnih segmenata u praksi se često pojavljuju diskontinuirane plohe kod kojih često nije moguće primjenom navedenih kriterija odlučiti radi li se o lokalno konkavnim ili konveksnim segmentima jer bi primjenom tih načela vrlo vjerovatno došli do pogrešnih rezultata što bi dovelo do pogrešno segmentiranih predmeta. Stoga, da bi se uspješno riješio navedeni problem potrebno je primijeniti dodatni geometrijski kriterij. Na slici 3.3. prikazan je primjer jedne lokalno diskontinuirane površine.



Slika 3.3. Lokalno diskontinuirana površina (Slika preuzeta iz [1])

Segmentacija predmeta je vrlo složen proces zato što se većina objekata sastoji od više dijelova koji mogu imati različitu namjenu. Predmeti koji se sastoje od jednog dijela posebna su skupina predmeta koji su najjednostavniji za segmentirati jer njih nema potrebe dodatno razdvajati. Pretpostavka je pak da se svaki nejednodijelni predmet sastoji od dva dijela: tijela i drške. Cilj takve segmentacije predmeta je primjena u robotici gdje je jedan od osnovnih zadataka robota

manipulacija, odnosno rukovanje predmetima. Ukoliko je moguće uspješno segmentirati dršku predmeta od ostatka tijela tada je uvelike olakšano rukovanje predmetom. Ukoliko robot nije u stanju segmentirati dijelove vrlo je lako moguće da će nanjeti štetu ili potpuno uništiti predmet kojim rukuje. Temeljna pretpostavka segmentacije je stoga da su različiti dijelovi predmeta također razdvojeni konkavnim granicama.

## 3.1. Dijagram tijeka algoritma metode lokalno konveksnih povezanih segmenata

Algoritam se sastoji od četiri glavna koraka. Prije samog procesa segmentacije potrebno je unijeti oblak točaka koji dobijamo od slika koje su snimljene koristeći RGB-D kameru.

Prvi korak algoritma je kreiranje grafa povezanih čvorova pomoću kojih se aproksimira promatrana površina u oblaku točaka. Graf se kreira tako što se koriste aproksimacije oblaka točaka konačnog broja čvorova. Svaki supervoksel u grafu se uzima kao čvor i računa se njegov vektor normale. Koristi se graf susjedstva supervoksela $G(V,E)$ gdje se svaki supervoksel $\vec{\mathbf{p}}_\mathbf{i} = (\vec{\mathbf{x}}_\mathbf{i}, \vec{\mathbf{n}}_\mathbf{i}), \vec{\mathbf{p}}_\mathbf{i} \in V$ uzima kao čvor. Središte i-tog supervoksela, odnosno čvora označava se s $\vec{\mathbf{x}}_\mathbf{i}$, dok se vektor normale i-tog čvora označava s $\vec{\mathbf{n}}_\mathbf{i}$. Čvorovi također mogu imati i težinsku vrijednost pomoću koje se mogu prikazati određena svojstva kao što su boja ili smjer normale.

Drugi korak algoritma je kreiranje grafa konveksnosti tako što se određuju veze između čvorova grafa. U tom koraku određuje se da li je veza između dva čvora $e = (\vec{\mathbf{p}}_\mathbf{i}, \vec{\mathbf{p}}_\mathbf{j})$ konkavna ili konveksna. Prilikom odlučivanja o konveksnosti, odnosno konkavnosti veze između dva čvora koriste se dva kriterija. Prvi kriterij je kriterij konveksnosti a drugi kriterij je *sanity* kriterij koji omogućava veću preciznost odlučivanja. [1]

**Kriterij konveksnosti:**

Neka su $\vec{\mathbf{p}}_\mathbf{1}$ i $\vec{\mathbf{p}}_\mathbf{2}$ dva susjedna čvora koja imaju središta $\vec{\mathbf{x}}_\mathbf{1}$ i $\vec{\mathbf{x}}_\mathbf{2}$ i vektore normale $\vec{\mathbf{n}}_\mathbf{1}$ i $\vec{\mathbf{n}}_\mathbf{2}$. Je li veza između njih konveksna ili konkavna može se zaključiti iz odnosa njihovih vektora normala i vektora koji spaja središta čvorova. Vektor koji spaja središta čvorova označava se sljedećim izrazom:

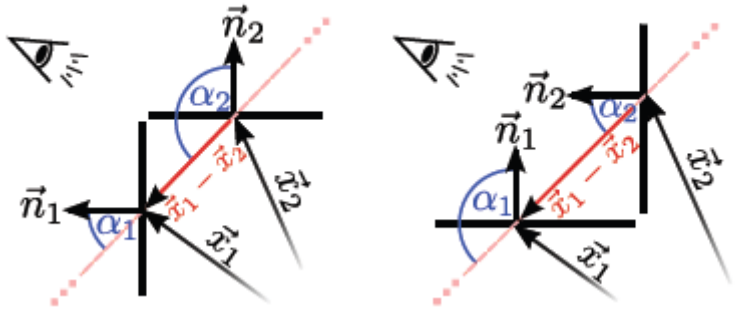
$$\vec{\mathbf{d}} = \vec{\mathbf{x}}_\mathbf{1} - \vec{\mathbf{x}}_\mathbf{2} \tag{3-1}$$

Kut između vektora $\vec{\mathbf{d}}$ i vektora normala čvora računa se pomoću vektorskog produkta:

$$\vec{\boldsymbol{d}} \cdot \vec{\boldsymbol{n}}_\mathbf{1} = |\vec{\mathbf{d}}| \cdot |\vec{\mathbf{n}}_\mathbf{1}| \cdot \cos(\alpha_1) \tag{3-2}$$

$$\vec{\boldsymbol{d}} \cdot \vec{\mathbf{n}}_\mathbf{2} = |\vec{\mathbf{d}}| \cdot |\vec{\mathbf{n}}_\mathbf{2}| \cdot \cos(\alpha_2) \tag{3-3}$$

Način na koji je moguće odrediti jesu li su veze između čvorova konveksne ili konkavne iz odnosa kuteva između vektora koji spaja čvorove i vektora normale moguće je utvrditi iz priložene slike 3.4..




Sl.3.4. Prikaz vektora $\vec{\mathbf{d}}$ i vektora $\vec{\mathbf{n}}$ u konveksnoj(lijevo) i konkavnoj(desno) vezi između čvorova (Slika preuzeta iz [1])

Na slici je jasno vidljivo da je kod konveksne veze kut $\alpha_2$ veći od kuta $\alpha_1$ dok je kod konkavne veze kut $\alpha_1$ veći od kuta $\alpha_2$. Iz toga je moguće doći do sljedećih relacija:

Konveksnost: $\alpha_1 < \alpha_2 \rightarrow \cos(\alpha_1) - \cos(\alpha_2) > 0$ , odnosno vrijedi da je:

$$\vec{\mathbf{n}}_1 \cdot \hat{d} - \vec{\mathbf{n}}_2 \cdot \hat{d} > 0 \qquad (3\text{-}4)$$

Konkavnost: $\alpha_1 > \alpha_2 \rightarrow \cos(\alpha_2) - \cos(\alpha_1) > 0$ , gdje također vrijedi da je:

$$\vec{\mathbf{n}}_1 \cdot \hat{d} - \vec{\mathbf{n}}_2 \cdot \hat{d} < 0 \qquad (3\text{-}5)$$

Također, $\hat{d}$ se izražava kao $\rightarrow \hat{d} = (\vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_2)/(\,\|\vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_2\,\|)$ (3-6)

U navedenim formulama izbor čvorova $\vec{\mathbf{x}}_1$ i $\vec{\mathbf{x}}_2$ je proizvoljan jer rezultat ne ovisi o smjeru vektora $\vec{\mathbf{d}}$ koji spaja te čvorove. Veze između čvorova čiji kut između normala iznosi manje od nekog graničnog kuta koji se označava s $\beta_{Thresh}$ označavati će se kao konveksne veze. Taj kut omogućava kompenzaciju nepreciznosti koja nastaje prilikom procjene normala i omogućava povezivanje malih konkavnih veza. Osim toga moguće je kompenzirati šum koji nastaje kod datoteka snimljenih RGB-D kamerom koristeći relaciju (3-7) kojom se dobivaju svi kutevi manji od graničnog iznosa kuta $\beta_{Thresh}$.

Kut $\beta$ predstavlja kut između normala $\vec{\mathbf{n}}_1$ i $\vec{\mathbf{n}}_2$ te iznosi:

$$\beta = |\alpha_1 - \alpha_2\,| = \cos^{-1}(\vec{\mathbf{n}}_1 \cdot \vec{\mathbf{n}}_2) < \beta_{Thresh} \qquad (3\text{-}7)$$

Svi kutevi koji zadovoljavaju navedeni kriterij (3-7) smatraju se konveksnima jer uobičajeno predstavljaju ravne površine čime se zanemaruju male konkavnosti koje se aprokisimiraju kao konveksne i onda se povezuju s ostalim konveksnim supervokselima.

Iz navedenih zakonitosti dolazi se do definicije kriterija konveksnosti (skraćeno *CC*):

$$CC(\vec{p}_i, \vec{p}_j) := \begin{cases} 1\,, \left(\vec{\mathbf{n}}_1 \cdot \hat{d} - \vec{\mathbf{n}}_2 \cdot \hat{d} > 0\right) \vee (\beta < \beta_{Thresh}) \\ 0\,, inače \end{cases} \tag{3-8}$$

Dakle, veza između dva supervoksela je konveksna ukoliko vrijedi da je $\alpha_1 < \alpha_2$ ili da je $\beta < \beta_{Thresh}$.

***Sanity* kriterij:**

U slučajevima kada se radi o lokalno diskontinuiranim površinama i kada nije moguće precizno odrediti radi li se o konkavnim ili konveksnim vezama koristi se *sanity* kriterij. Promatramo vektor $\vec{\mathbf{d}}$ koji povezuje središta čvorova kao i vektor $\vec{\mathbf{s}}$ koji predstavlja presječnicu dvaju površina. Vektor $\vec{\mathbf{s}}$ se može dobiti izrazom:

$$\vec{\mathbf{s}}(\vec{\mathbf{n}}_1, \vec{\mathbf{n}}_2) = \vec{\mathbf{n}}_1 \times \vec{\mathbf{n}}_2 \tag{3-9}$$

Kut između vektora $\vec{\mathbf{d}}$ i $\vec{\mathbf{s}}$ označava se s $\vartheta$. Zato što je orijentacija vektora $\vec{\mathbf{s}}$ proizvoljna kut $\vartheta$ se definira kao minimalni kut između orijentacija vektora:

$$\vartheta(\vec{\mathbf{p}}_1, \vec{\mathbf{p}}_2) = \min(\angle(\vec{\mathbf{d}},\vec{\mathbf{s}}) , \angle(\vec{\mathbf{d}},-\vec{\mathbf{s}})) = \min(\angle(\vec{\mathbf{d}},\vec{\mathbf{s}}) , 180° - \angle(\vec{\mathbf{d}},-\vec{\mathbf{s}})) \tag{3-10}$$

Kao i kod kriterija konveksnosti postoji granični kut $\vartheta_{\text{Thresh}}$ koji služi za kompenzaciju šuma koji nastaje zbog korištenog senzora. Kut $\vartheta_{Thresh}$ zapravo je sigmoidna funkcija kojom se računa kut između normala[1]:

$$\vartheta_{Thresh}^{max}(\beta) = \vartheta_{Thresh}^{max} \cdot (1 + \exp\,[-a \cdot (\beta - \beta_{off})])^{-1}, \tag{3-11}$$

gdje je $\beta\angle(\vec{\mathbf{n}}_1, \vec{\mathbf{n}}_2)$ kut između normale $\vec{\mathbf{n}}_1$ i normale $\vec{\mathbf{n}}_2$, a oznaka *a* označava prozivoljnu varijablu pomoću koje se može mjenjati iznos kuta. Da bi se ispunio uvjet konveksnosti potrebno je da je kut $\vartheta$ manji od kuta $\vartheta_{Thresh}$.

Tako se *sanity* kriterij(*SC*) definira sljedećom relacijom (3-12) :

$$SC := \begin{cases} 1, & \vartheta(\vec{\mathbf{p}}_\mathbf{i}, \vec{\mathbf{p}}_\mathbf{j}) > \vartheta_{Thresh}(\beta(\vec{\mathbf{n}}_1, \vec{\mathbf{n}}_2)) \\ 0, & \text{inače} \end{cases} \tag{3-12}$$

Treći korak algoritma je segmentacija dobivenih grafova konveksnosti da bi se pronašli svi supervokseli povezani konveksnim bridovima. Taj korak se postiže pomoću procesa širenja područja. Kreće se od nasumično odabranog korijenskog supervoksela koji predstavlja početnu točku. Zatim se koristi dubinska pretraga koja omogućava rast povezanih supervoksela samo preko konveksnih bridova sve dok se ne dođe do konkavnosti te se supervokseli odvojeni konkavnim granicama ne povezuju. Lokalna konveksnost se očituje samo ako su zadovoljeni i kriterij konveksnosti i *sanity* kriterij. Proces traje dok se ne povežu svi lokalno konveksni supervokseli u segmente, a tada se taj čitav proces ponavlja počevši od neke druge točke koja još nije procesuirana te prestaje djelovati kada ne preostane slobodnih supervoksela za pridruživanje području. Svojstvo komutativnosti omogućava jednake rezultate bez obzira o izboru početnih supervoskela.

Zadnji korak algoritma je filtriranje šuma nastalog prilikom upotrebe senzora ili procjenom vektora normale. Šum je manji što je manji granični kut $\beta_{\mathrm{Thresh}}$ u kriteriju konveksnosti što povećava preciznost segmentacije, međutim problem se pojavljuje kod premalih iznosa kuta $\beta_{\mathrm{Thresh}}$ jer je moguće pojavljivanje izoliranih segmenata nastalih zbog šumova koji se ponekad tretiraju kao samostalni objekti. Navedeni problem se uklanja upravo zadnjim korakom algoritma koji se zove filtriranje. U procesu filtriranja se koristi jednostavni filter koji ima kao parametar prirodni broj $\mathrm{n}_{\mathrm{filter}}$. Nakon segmentacije provjerava se veličina svakog dobijenog segmenta i uspoređuje s veličinom filtera. Ukoliko je broj supervoksela nekog određenog segmenta manji ili jednak veličini filtera tada se on povezuje s najvećim susjednim segmentom i smatra se dijelom njega. Proces filtriranja obavlja se sve dok postoje segmenti koji imaju susjedne segmente i manji su ili jednaki od veličine $\mathrm{n}_{\mathrm{filter}}$.

## 3.2. Mjere kvalitete segmentacije

Kvaliteta segmentacije može se ocijeniti usporedbom segmentirane slike s *ground-truth* slikom koja predstavlja segmentaciju urađenu od strane čovjeka pomoću odgovarajućeg programa za obradu slike. Iako ograničenost programa može smanjiti preciznost segmentacije, ljudska percepcija ima sposobnost da segmentira sliku vrlo precizno pa zbog toga predstavlja dobru podlogu za provjeru ispravnosti i kvalitete segmentacije dobivene različitim metodama računalnog vida. Segmenti koji postoje na *ground-truth* slici označavaju se slovom *G*, gdje postoji skup segmenata $G=\{G_1, G_2,\ldots,G_M\}$, te se uspoređuju sa segmentima koji su dobiveni procesom segmentacije, $S=\{S_1, S_2, \ldots, S_N\}$. Prva mjera koja se koristi je maksimalno preklapanje

koje se označava s *WOv*. Kod te mjere uspoređuju se *ground-truth* segmenti sa segmentima dobivenim segmentacijom gdje se za svaki objekt koji je predstavljen na *ground-truth* segmentima uzima se onaj segment koji ima najveće preklapanje. Upravo najveće preklapanje segmenata s *ground-truth* segmentima se označava kao preklapanje, koje se označava s $Ov_i$. Maksimalno preklapanje se označava kao [1]:

$$Ov_i = \max_{sj}(|G_i \cap S_j|)/(G_i \cup S_j) \tag{3-13}$$

Težinsko preklapanje se računa kao težinski prosjek preklapanja u odnosu na veličinu segmenata:

$$WOv = (1/\Sigma_i|G_i|) \cdot (\Sigma_i|G_i| \cdot Ov_i) \tag{3-14}$$

Težinsko preklapanje može poprimati vrijednosti između 0 i 1, gdje se 1 smatra savršenom segmentacijom i gdje se *ground-truth* slika i segmentirana slika u potpunosti preklapaju.

Osim težinskog preklapanja, kvaliteta segmentacije se može izraziti preko *True-positive(tp) i False-positive(fp)* koeficijenata. *True-positive* je koeficijent koji se određuje kao broj točno segmentiranih točaka te predstavlja presjek skupova $G_i$ i $S_i$. *False-positive* koeficijent se izražava pomoću razlike skupova $S_i$ i $TP_i$. Osim njih postoji i *false-negative* koeficijent koji predstavlja one točke koje postoje samo u *ground-truth* slici a ne i na segmentiranoj slici. Stoga su ti koeficijenti izraženi pomoću sljedećih (3-15) formula gdje se prilikom računanja koristi i ukupan broj segmenata koji postoje na *ground-truth* slici, a koji se označava s $N_G$ [1]:

$$tp = (1/N_G) \cdot (\Sigma_i((|TP_i|)/(|G_i|)), \qquad fp = (1/N_G) \cdot (\Sigma_i((|FP_i|)/(|S_i|)),$$

$$fn = (1/N_G) \cdot (\Sigma_i((|FN_i|)/(|G_i|)) \tag{3-15}$$

Kvalitetu segmentacije moguće je izraziti i pomoću *over-* i *under-* segmentacije. *Over-segmentacija* (koja se izražava kao i $F_{os}$) predstavlja broj točno određenih piksela u odnosu na ukupan broj piksela koji čine objekte na slici, dok je *under-segmentacija* broj netočno određenih piksela u odnosu na piksele svih objekata na slici. Te dvije mjere se mogu iskazati formulama:

$$F_{os} = 1 - (N_{točno}/N_{ukupno}) \tag{3-16}$$

$$F_{us} = (N_{netočno}/N_{ukupno}) \tag{3-17}$$

# 4.SUČELJE ZA IMPLEMENTACIJU METODE LOKALNO KONVEKSNIH POVEZANIH SEGMENATA

Za realizaciju metode lokalno konveksnih povezanih segmenata korišteno je razvojno okruženje Microsoft Visual Studio 2013, biblioteka Point Cloud Library (PCL) 1.8 te VTK biblioteka koja se koristi za trodimenzionalnu vizualizaciju slike. PCL je biblioteka otvorenog koda koja sadrži algoritme kojima se koristeći oblake točaka izvode različiti zadatci usko povezani s 3D geometrijom. Biblioteka sadrži algoritme koji se mogu koristiti za vizualizaciju i percepciju scene u robotici, za filtriranje datoteka koje sadrže šumove, povezivanje 3D oblaka točaka u jedinstvene segmente, segmentiranje važnih dijelova scene, isticanje ključnih objekata te omogućavanje prepoznavanja objekata na osnovu njihovog geometrijskog oblika.[3] Za implementaciju PCL biblioteke u program Microsoft Visual Studio 2013 potrebno je postaviti varijable okruženja sustava imena: „PCL180_ROOT“, „OPENCV_DIR“, „OPENNI2_REDIST64“, „OPENNI2_ LIB64“ i „OPENNI_INCLUDE64“ u postavkama sustava. Osim toga, potrebno je u samom programu dodati datoteku „PCL1.8.0_X64“ koja omogućava dodavanje svih dodatnih PCL direktorija koji sadrže dokumente potrebne za korištenje PCL biblioteke. Programski kod algoritma pisan je pomoću objektno-orijentiranog programskog jezika C++ i u cijelosti je priložen u poglavlju u kojem se nalaze prilozi. Programu se prilaže .pcd datoteka koju je cilj segmentirati pomoću algoritma lokalno konveksnih povezanih segmenata. Datoteka sadržava dvodimenzionalno polje trodimenzionalnih točaka koje se dobije pomoću programa za trodimenzionalnu rekonstrukciju. Datoteka se učitava sljedećim nizom naredbi:

```
pcl::PCLPointCloud2 input_pointcloud2;
pcl::io::loadPCDFile("C:/Users/Srđan/Desktop/zavrsni/slika.pcd", input_pointcloud2)
```

Kao što je vidljivo potrebno je navesti putanju do željene datoteke, a za segmentaciju različitih datoteka potrebno je samo promijeniti ime datoteke. Nakon što je datoteka učitana provjerava se postoje li vektori normale već u priloženoj datoteci. Ukoliko ne postoje, tada algoritam računa vektore normale na slici. Sljedeći korak predstavlja definiranje parametara segmentacije. Prvo je potrebno postaviti parametre vezane uz supervoksele. Pod njih spadaju rezolucija voksela, rezolucija točaka, te varijable koje označavaju važnost boje supervoksela, važnost skalarnog produkta vektora normale i važnost prostorne udaljenosti između supervoksela. Nakon toga je potrebno postaviti parametre segmentacije pod koje spadaju granični kut $\beta_{Thresh}$, minimalna veličina jednog segmenta te logičke varijable kojima se može birati hoće li se koristiti dodatni

kriterij konveksnosti kao i *sanity* kriterij. Dodatni kriterij konveksnosti temelji se na varijabli „k_factor“ koja ukoliko je veća od nule tada da bi veza između dva supervoksela bila konveksna mora postojati broj susjednih supervoksela koji moraju imati konveksnu vezu s oba promatrana supervoksela jednak iznosu te varijable. Tada se odgovarajućim metodama postavljaju vrijednosti svih parametara. Nakon toga se prema postavljenim parametrima točke slike grupiraju u supervoksele, što se postiže sljedećom naredbom:

```
pcl::SupervoxelClustering<PointT>super(voxel_resolution,seed_resolution);
```

Na priloženom ispisu programa koji se nalazi na slici 4.1. vidi se da program računa broj supervoksela koji ispisuje naredbom:

```
temp<<"Broj temp<<"Broj  supervoksela:"<<supervoxel_clusters.size()<<"\n";
```

Nakon što su dobiveni supervokseli grupirani dolazi se do postupka dohvaćanja matrice susjedstva koje se obavlja sljedećim naredbama:

```
std::multimap<uint32_t, uint32_t> supervoxel_adjacency;
super.getSupervoxelAdjacency(supervoxel_adjacency);
```

Sljedeći postupak se sastoji od dohvaćanja oblaka točaka s vektorima normale na osnovu grupiranih supervoksela:

```
pcl::PointCloud<pcl::PointNormal>::Ptr sv_centroid_normal_cloud=
pcl::SupervoxelClustering<PointT>::makeSupervoxelNormalCloud(supervoxel_clusters);
```

Kada su svi navedeni procesi obavljeni tada se počinje s procesom segmentacije tako što se koristi lccp klasa gdje se pomoću metoda za postavljanje parametara unose postavljene vrijednosti, a onda se poziva sljedeća metoda kojom se povezuju lokalno konveksni segmenti:

```
lccp.segment();
```

Također važan je i korak koji obuhvaća dubinsku pretragu grafa, koja se obavlja pomoću dva brojača: jednog koji se koristi za pretragu supervoksela i drugog koji se koristi za provjeru matrice susjedstva. Taj korak se obavlja koristeći sljedeće naredbe:

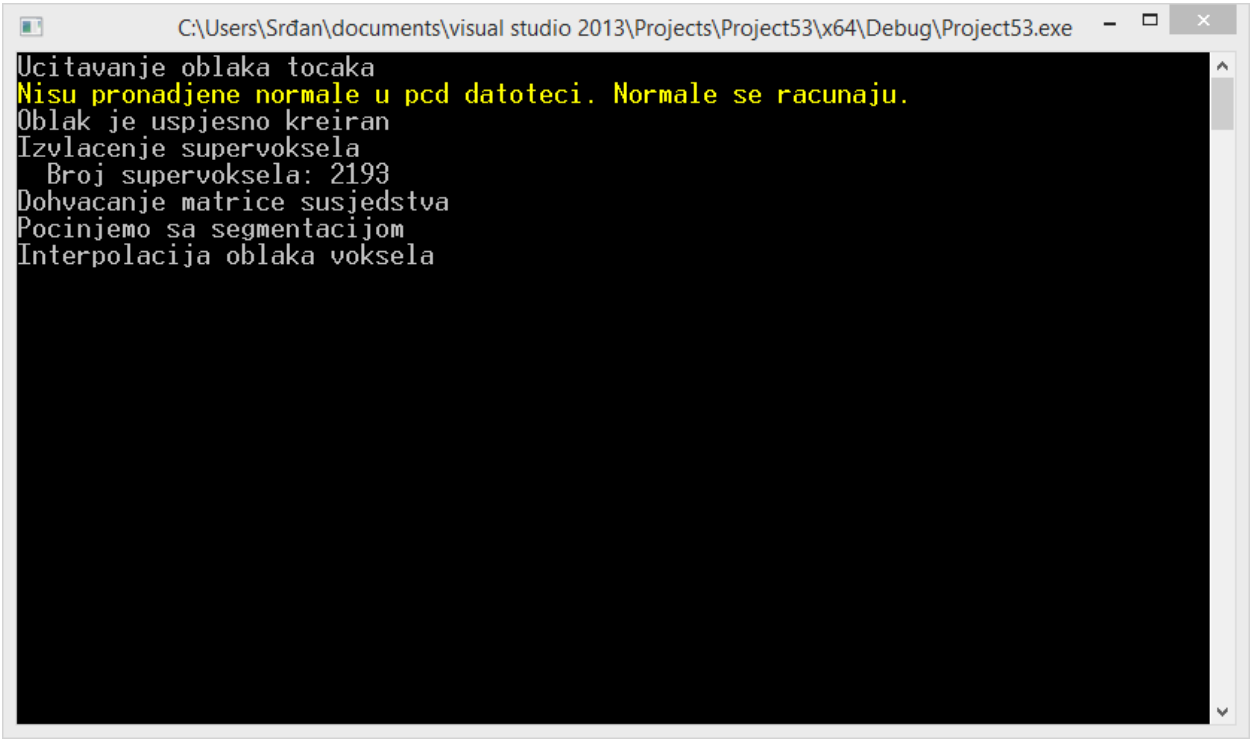
```
std::pair<VertexIterator, VertexIterator> vertex_iterator_range;
vertex_iterator_range = boost::vertices(sv_adjacency_list);
```

Kreira se oblak točaka, a onda se obavlja proces indeksiranja točaka unutar oblaka točaka. Nakon toga koriste se funkcije da bi se oblaku točaka pridjelila određena boja zavisno o tome je li veza konkavna ili konveksna na temelju čega se taj oblak segmentira kao objekt. Podatkovna struktura *polydata* je kreirana da bi se podatci spremali u tu strukturu. Pomoću brojača prolazi se kroz sve supervoksele te se provjeravaju odnosi sa susjednim supervokselima. Kada se dohvate supervokseli provjerava se ukoliko je veza među njima konveksna ili konkavna te se stoga određena boja pridodjeljuje toj vezi. Bilježe se dobiveni podatci o točkama te podatci o

susjednim točkama za svaku procesuiranu točku, a zaim se točke dodaju kreiranoj podatkovnoj strukturi u kojoj se spremaju sve informacije. Nakon što su sve infromacije spremljene unutar strukture *polydata* završava se proces vizualizacije za čiji je prikaz potrebno još postaviti preglednik što se obavlja sljedećim nizom naredbi:

```
pcl::visualization::PCLVisualizer::Ptrviewer(new
pcl::visualization::PCLVisualizer("3DPreglednik"));
        viewer->setBackgroundColor(0, 0, 0);
        viewer->registerKeyboardCallback(keyboardEventOccurred, 0);
        viewer->addPointCloud(lccp_labeled_cloud, "maincloud");
```

Nakon što je primijenjen programski kod koji se nalazi u prilogu, dobije se povratna informacija o procesu koji se odvija te se dobiju rezultati segmentacije unutar 3D preglednika. Na slici 4.1. moguće je vidjeti prozor u kojem se nalazi ispis procesa koji se obavljaju u procesu segmentacije:

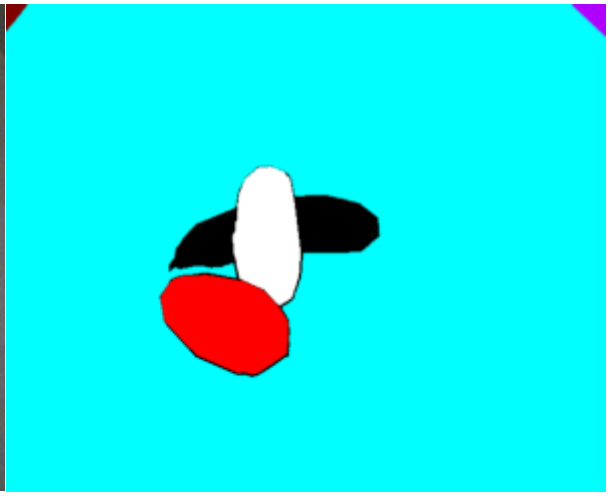



Slika 4.1. Prikaz prozora u kojem se ispisuju koraci procesa segmentacije

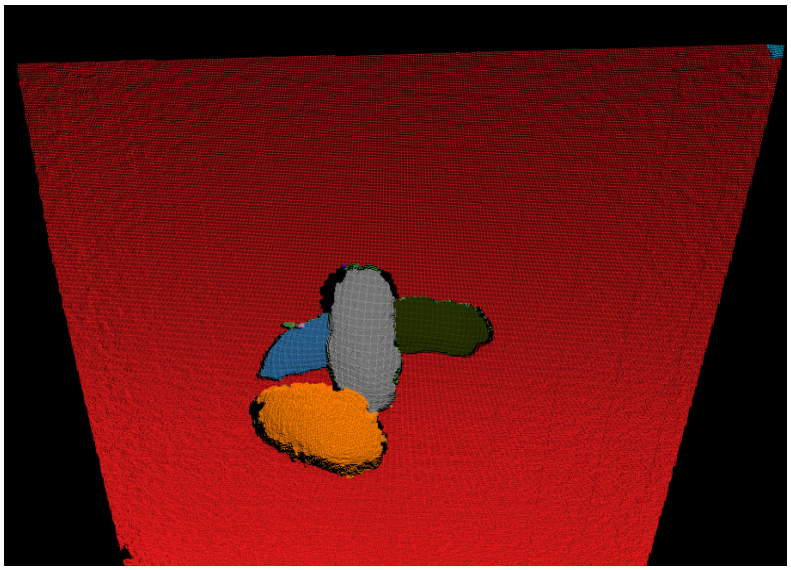
# 5. POSTIGNUTI REZULTATI

Primjenom algoritma lokalno konveksnih povezanih segmenata u programu Microsoft Visual Studio 2013 dobiveni su rezultati segemntacije koji su pokazali da je navedenom metodom moguće segmentirati željene slike koje su snimljene pomoću RGB-D kamere. U procesu segmentacije korišteni su parametri koji se nalaze unutar programskog koda u prilogu. U ovom poglavlju su priloženi rezultati procesa segmentacije uz kratki osvrt na svaki od rezultata. Za svaku segmentiranu sliku priložene su: originalna slika, *ground-truth* slika segmentirana ručno u programu paint.net, koja odgovara ljudskoj percepciji segmentacije i slika segmentirana metodom lokalno konveksnih povezanih segmenata.

Sl.5.1. Originalna slika



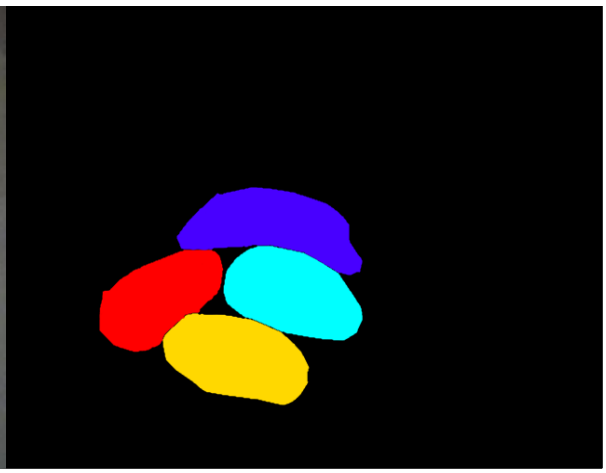
Sl.5.2. *Ground-truth* slika



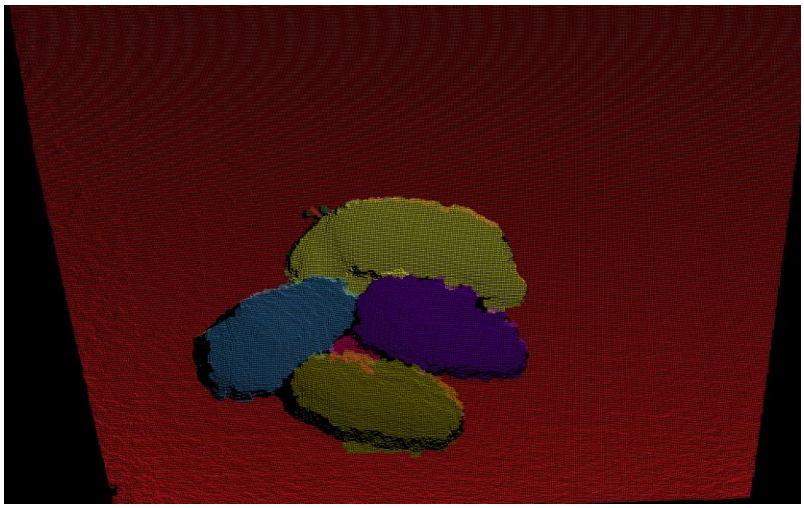
Sl.5.3. Slika segmentirana metodom lokalno konveksnih povezanih segmenata

Prema slici 5.3. moguće je vidjeti da je jednostavnu sliku koja se sastoji od tri objekta na ravnoj pozadini moguće segmentirati s velikom preciznošću. Točnost segmentacije uvelike ovisi o izboru parametara segmentacije, a iz priložene slike 5.3. moguće je vidjeti da su svi objekti gotovo identični onima na originalnoj slici. Ipak, može se primjetiti da postoji utjecaj šuma zbog čega se pojavljuju mala odstupanja koja su naročito vidljiva na rubovima objekata. Osim toga, program je različito segmentirao dijelove predmeta kojega drugi predmet preklapa i samim time dijeli na dva dijela iako oni pripadaju istom objektu. Osim toga, na slici je vidljivo da se u gornjim rubovima pojavljuje prijelaz koji predstavlja rubove stola. Na segmentiranoj slici taj prikaz je vidljiv na desnom rubu gdje postoji i veći segment koji odudara od ostatka površine stola dok na lijevoj strani nije detektiran taj prijelaz.

Sl.5.4. Originalna slika



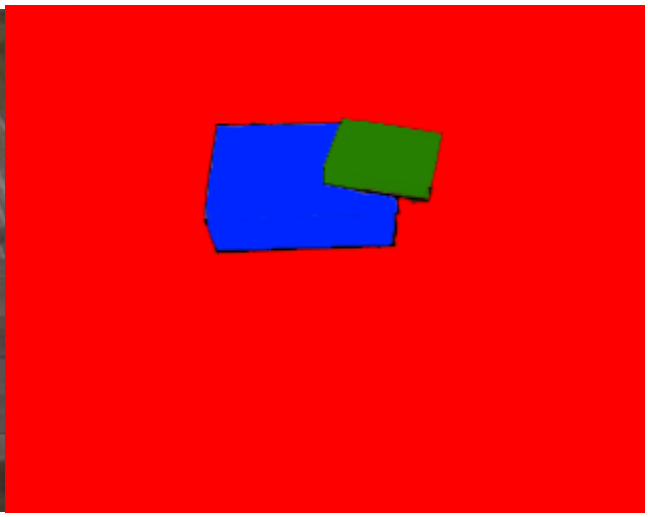
Sl.5.5. *Ground-truth* slika



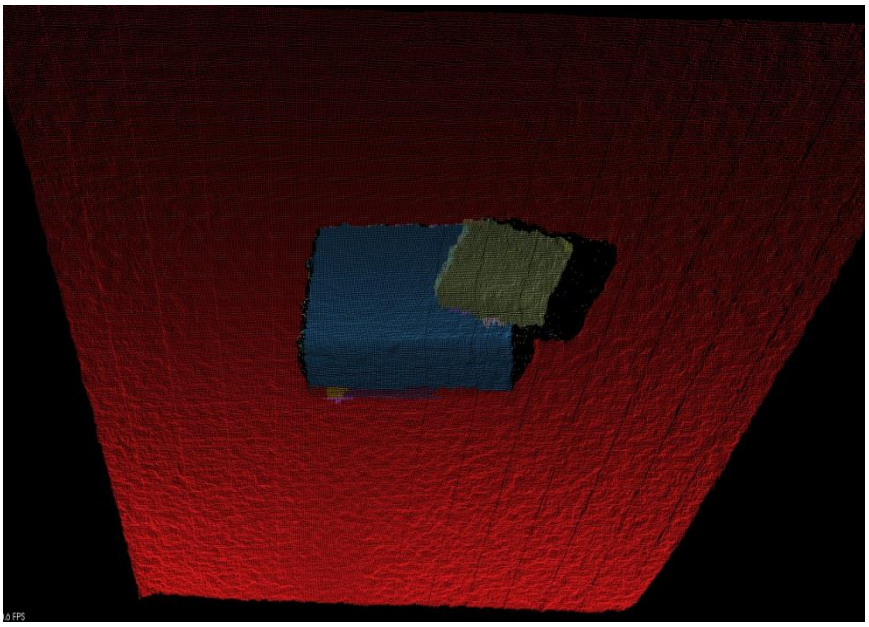
Sl.5.6. Slika segmentirana metodom lokalno konveksnih povezanih segmenata

Na slici 5.6. moguće je vidjeti rezultate segmentacije druge priložene slike 5.4., koja se sastoji od četiri objekta i pozadine. Moguće je vidjeti da je svaki predmet uspješno segmentiran od pozadine. Kod pojedinih predmeta se ponovno javlja blaga *oversegmentacija* ali su ipak segmentirani s velikom uspješnošću. Međutim, kao što je i vidljivo postoje dijelovi pozadine koji su odvojeni predmetima od ostatka pozadine. Ta područja su odvojeno segmentirana kao posebni objekti upravo zbog odijeljenosti od ostatka površine pozadine.

Sl.5.7. Originalna slika



Sl.5.8. *Ground-truth* slika

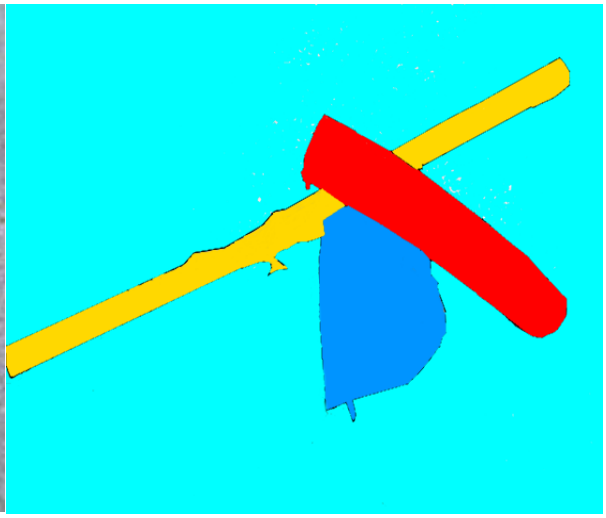


Sl.5.9. Slika segmentirana metodom lokalno konveksnih povezanih segmenata

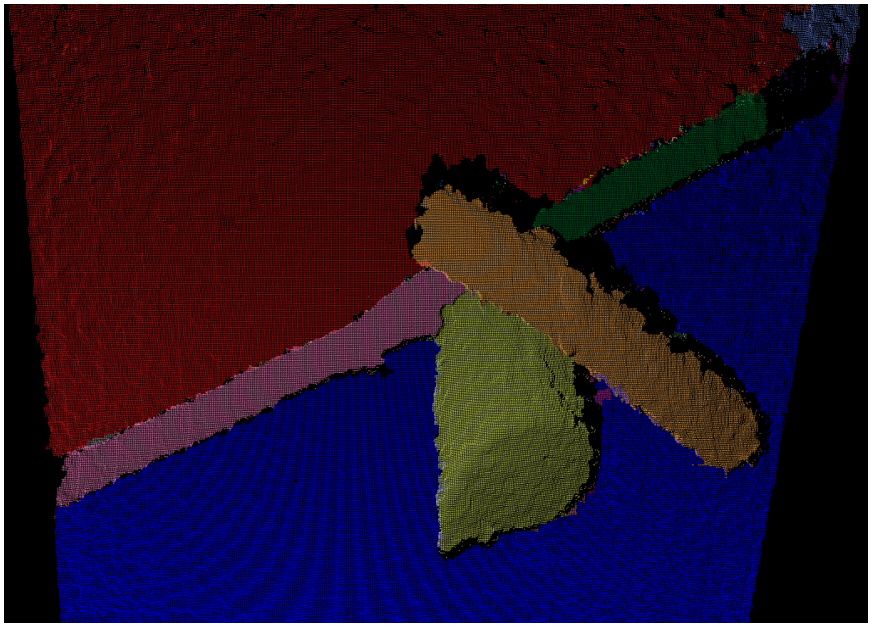
Priložena slika 5.7 sadrži dva objekta koji se nalaze jedan na drugome. Predmeti su jednodijelni a prema slici 5.9. moguće je vidjeti da su ti predmeti dosta precizno i točno segmentirani iako postoje na rubovima mali segmenti koji odudaraju od oblika predmeta moguće je zaključiti da

metodom lokalno konveksnih povezanih segmenata je moguće segmentirati i objekte koji se nalaze na površini drugog objekta.

Sl.5.10. Originalna slika



Sl.5.11. *Ground-truth* slika

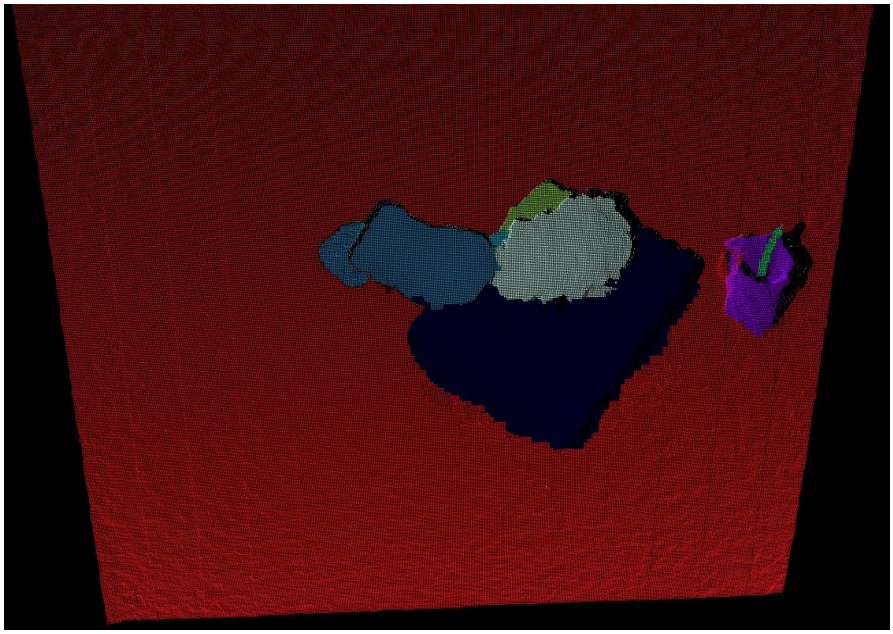


Sl.5.12. Slika segmentirana metodom lokalno konveksnih povezanih segmenata

Na slici 5.12. moguće je vidjeti rezultate segmentacije slike 5.10.. Kao što se može primjetiti objekti sa slike su opet vrlo precizno segmentirani. Međutim, ponovno se pojavljuje rezultat da se predmet kojega drugi predmet preklapa i dijeli na dva dijela i segmentira u ta dva dijela kao da su različiti predmeti. Osim toga na slici 5.12. je vidljivo da se i pozadina dijeli na dva različita dijela zbog objekta koji se gotovo proteže od jednoga do drugoga kraja.

Sl.5.13. Originalna slika

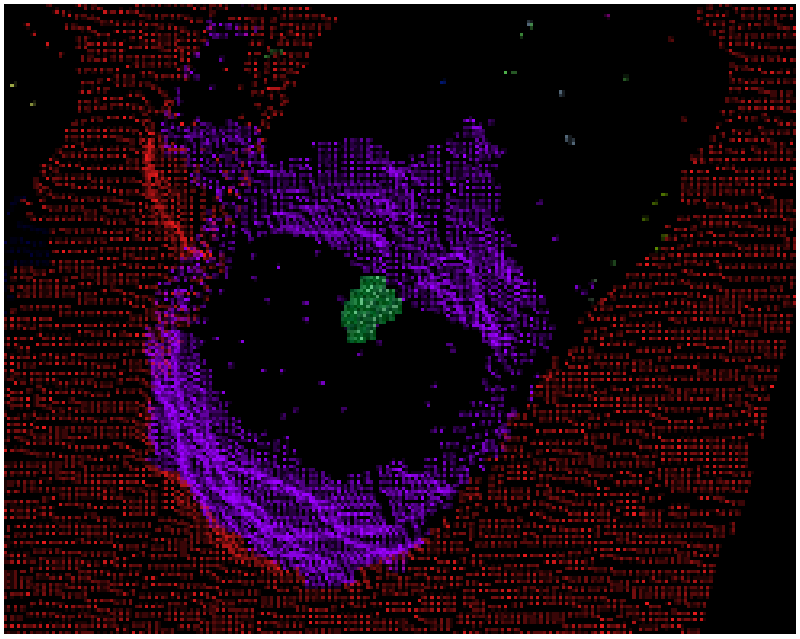
Sl.5.14. *Ground-truth* slika



Sl.5.15. Slika segmentirana metodom lokalno konveksnih povezanih segmenata

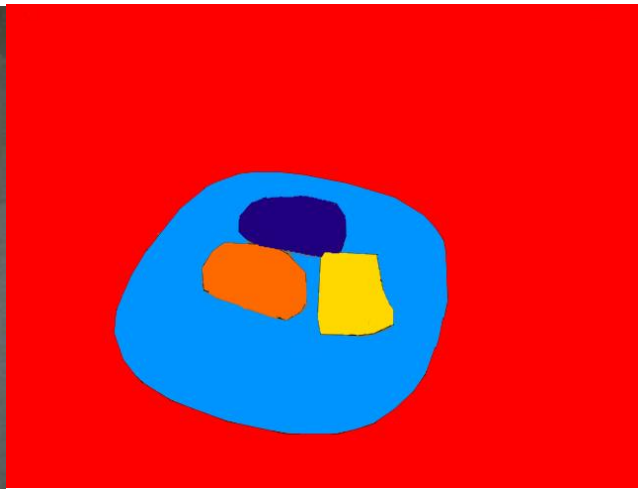
Usporedbom slike 5.13. i slike 5.15. možemo zaključiti da je sliku koja se sastoji od većeg broja objekata moguće segmentirati uz veliku preciznost. Iako je vidljivo da je svaki predmet prepoznat i segmentiran moguće je primjetiti i manje probleme prilikom segmentacije. Kao i u prethodnim situacijama, predmet kojeg drugi predmeti dijeli na više dijelova je odvojeno segmentiran pa je tako kutija sa slike segmentirana u dva dijela. Osim toga pojavljuju se manje nepreciznosti kod ostalih predmeta, ali su one u usporedbi s dimenzijama tih predmeta gotovo zanemarive. Ipak, važno je primjetiti da ručka šalice nije pravilno segmentirana. Također, na

slici 5.16. gdje je prikazana segmentirana šalica iz drugog kuta moguće je primjetiti da algoritam ne segmentira uspješno unutrašnjost šupljih predmeta iako je ona dio tog predmeta. To područje je označeno crnom bojom te ne pripada niti šalici niti pozadini slike.

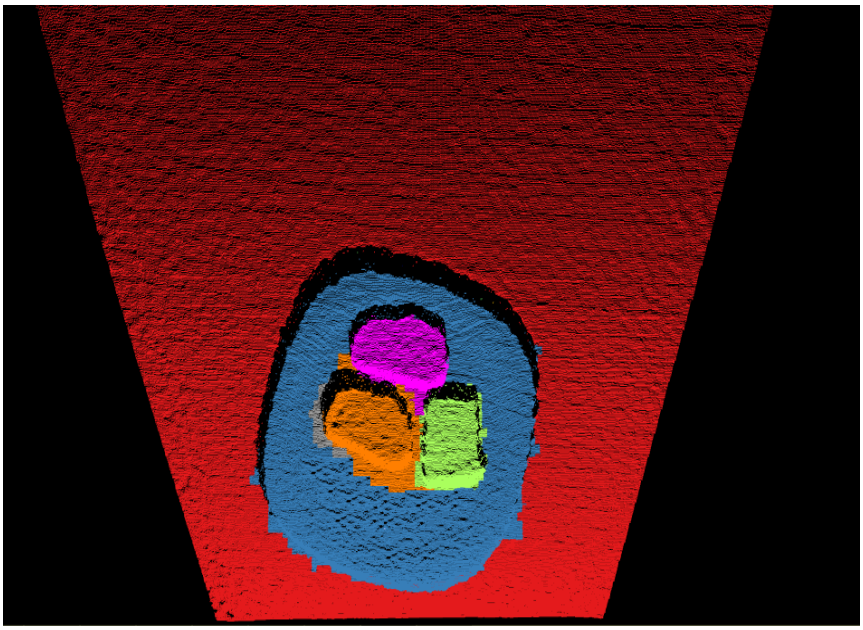


Slika5.16. Prikaz šalice iz drugog kuta

Sl.5.17. Originalna slika



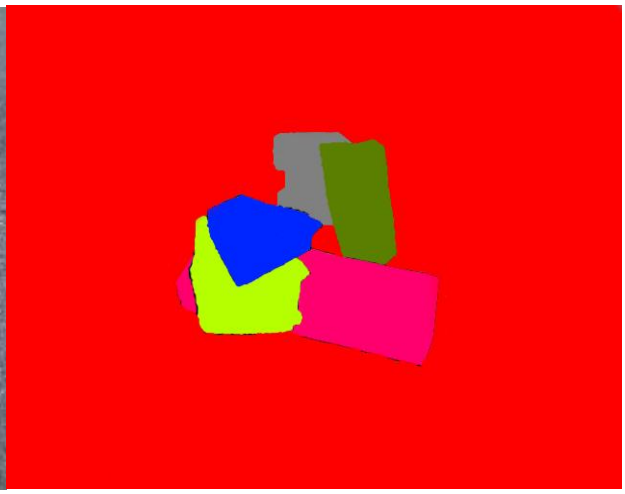
Sl.5.18. *Ground-truth* slika



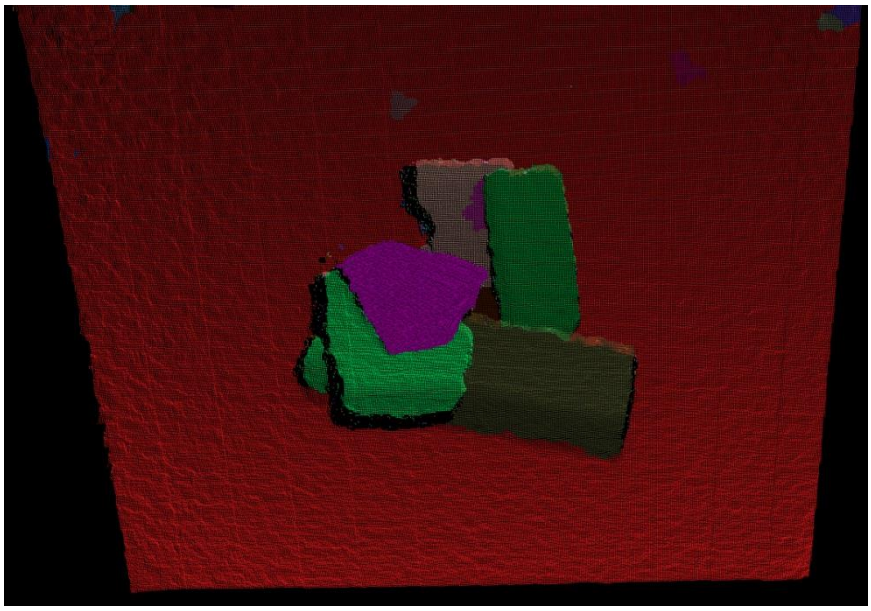
Sl.5.19. Slika segmentirana metodom lokalno konveksnih povezanih segmenata

Usporedbom slika 5.17. i 5.19. moguće je vidjeti rezultate segmentacije šeste priložene slike. Predmeti sa slike su uspješno segmentirani i različito obojani kao odvojeni objekti ali je također vidljivo da postoji *oversegmentacija* predmeta koji se vrlo malo izdižu iznad površine na kojoj stoje, u ovom slučaju tanjura što predstavlja problem metodi segmentacije.

Sl.5.20. Originalna slika



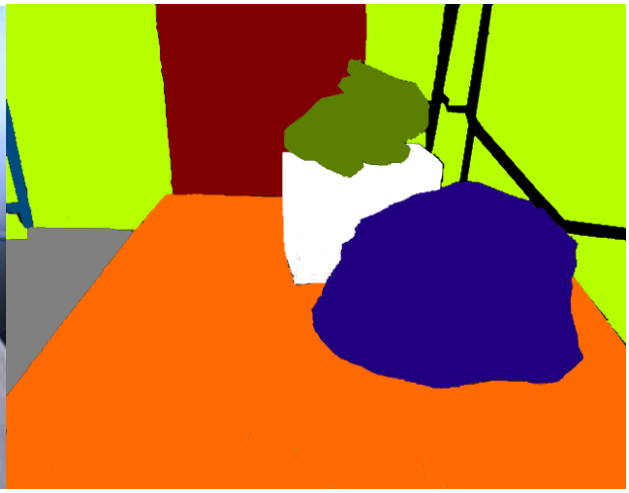
Sl.5.21. *Ground-truth* slika



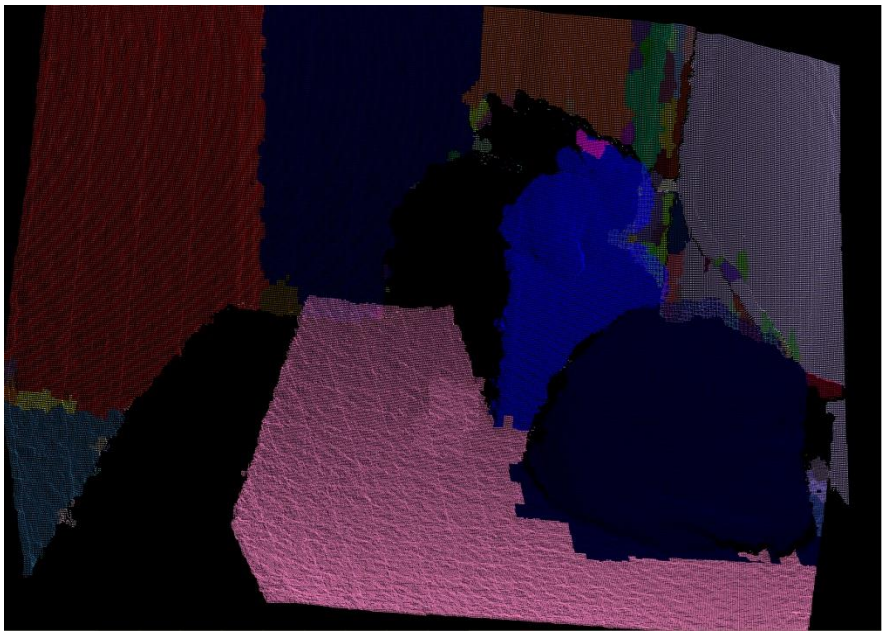
Sl.5.22. Slika segmentirana metodom lokalno konveksnih povezanih segmenata

Usporedbom slike 5.20. i slike 5.22. vidljivo je da je svaki objekt semgentiran uspješno iako se pojavljuju manji segemnti koji su pogrešno segmentirani. Vidljivo je da je dio jednog bloka segmentiran zajedno s blokom koji se nalazi na njemu. Usprkos tome, proces segmentacije je proizvео zadovoljavajuće rezultate.

Sl.5.23. Originalna slika



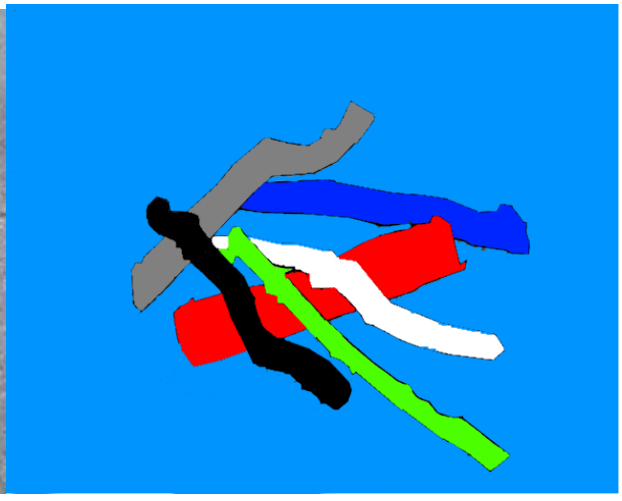
Sl.5.24. *Ground-truth* slika



Sl.5.25. Slika segmentirana metodom lokalno konveksnih povezanih segmenata

Na slici 5.23. moguće je vidjeti sljedeću sliku koja je segmentirana. Slika se sastoji od stola na kojemu se nalaze tri objekta te složene pozadine. Kao što je vidljivo na slici 5.25. pozadina slike vrlo uspješno segmentirana osim malog dijela koji se nalazi u desnom kutu gdje je pozadina prikazana većim brojem manjih segmenata. Stol je uspješno segmentiran i odvojen od objekata na njemu. Iako je vreća sa slike uspješno segmentirana problem se pojavio prilikom segmentacije kutije i plišanog zeca koji se nalazi na kutiji. Naime, ta dva objekta su zajedno segmentirana kao jedinstveni objekat.

Sl.5.26. Originalna slika



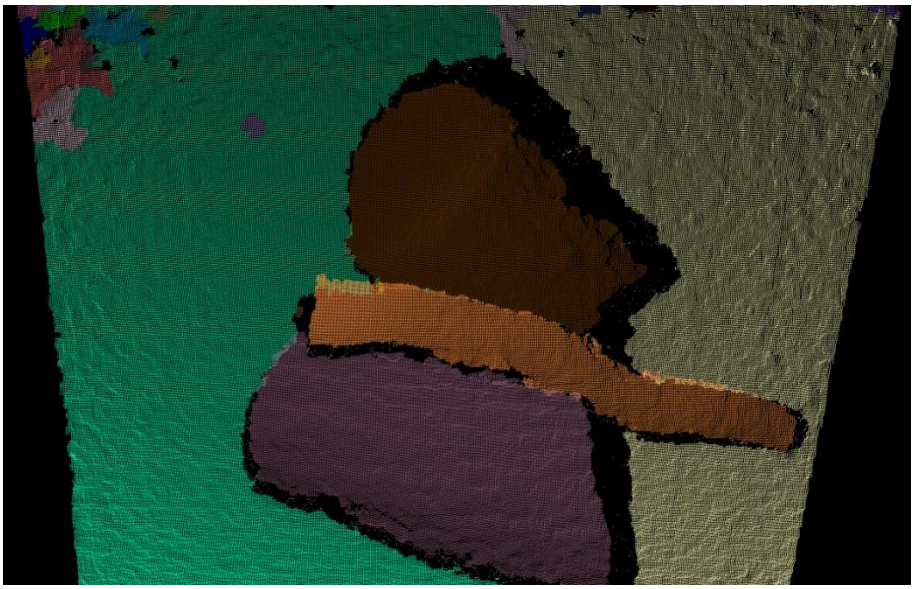
Sl.5.27. *Ground-truth* slika

Sl.5.28. Slika segmentirana metodom lokalno konveksnih povezanih segmenata

Prema slici 5.28. možemo vidjeti da je slika 5.26. uspješno segmentirana. Iako se pojavljuju male nepreciznosti one nisu toliko velike da utječu na kvalitetu segmentacije. Usprkos tome što je većina grana uspješno segmentirana moguće je vidjeti da su dvije grane spojeno segmentirane s granom koja se nalazi ispod njih zbog toga što su polegnute vrlo malo iznad površine donje grane.

Sl.5.29. Originalna slika

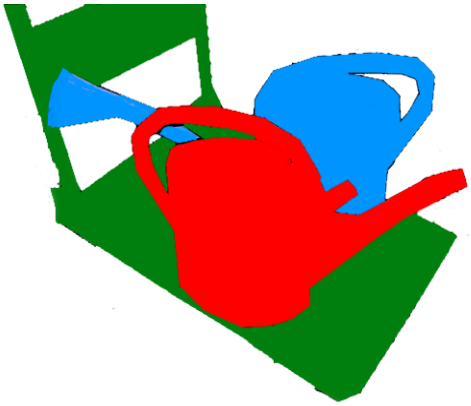
Sl.5.30. *Ground-truth* slika



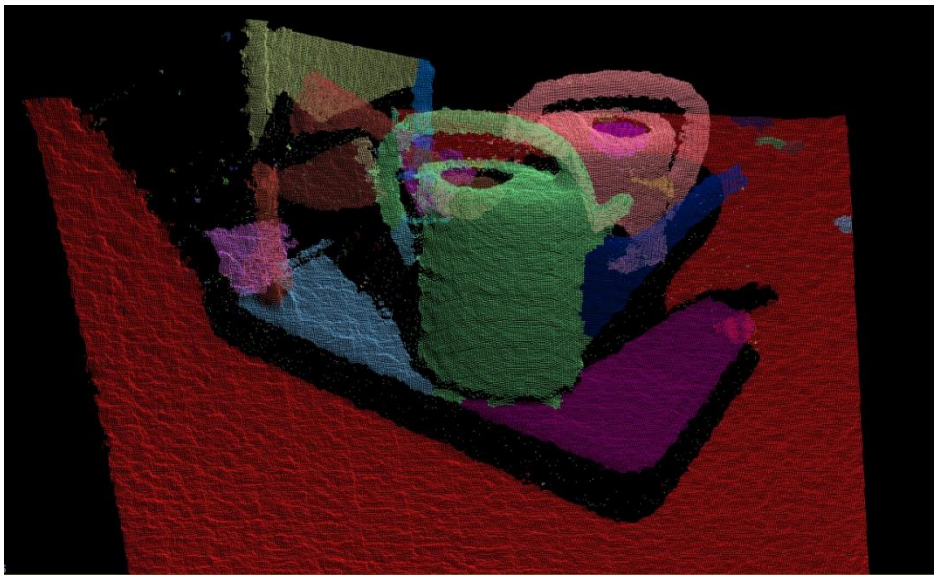
Sl.5.31. Slika segmentirana metodom lokalno konveksnih povezanih segmenata

Na slici 5.31. moguće je vidjeti još jedan uspješan primjer segmentacije većih objekata gdje segmentirani objekti gotovo odgovaraju onima na slikama 5.29. i 5.30.. Ipak, vidljivo je da se pojavljuju određeni šumovi na vrhu slike koji dovode do manjih segmenata formiranih na površini pozadine.

Sl.5.32. Originalna slika



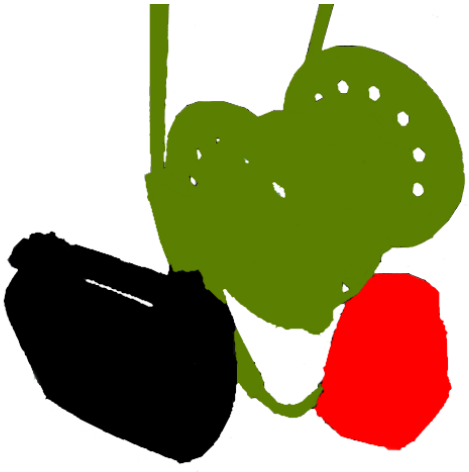
Sl.5.33. *Ground-truth* slika



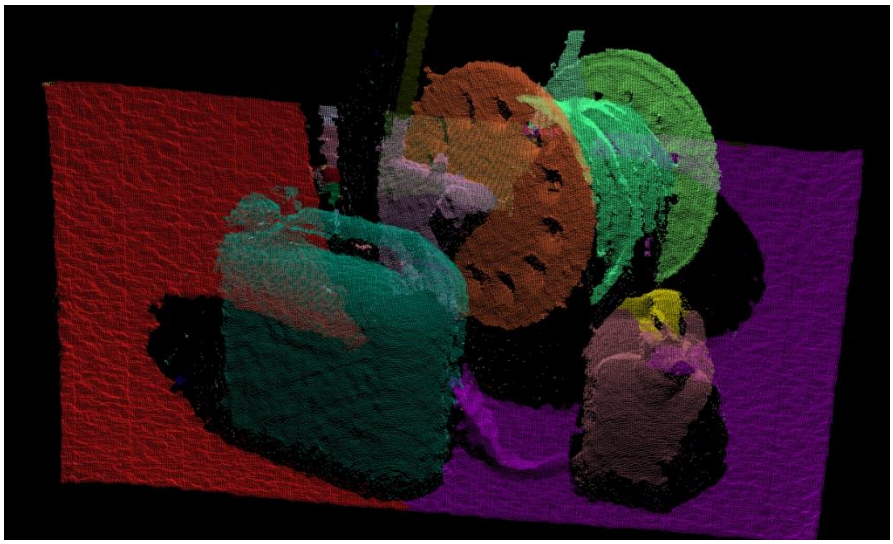
Sl.5.34. Slika segmentirana metodom lokalno konveksnih povezanih segmenata

Slika 5.34. predstavlja rezultat segmentacije slike 5.32.. Na slici se nalaze kolica i dvije kante za zalijevanje. Svaki predmet je segmentiran uz vrlo visoku preciznost. Na ovom primjeru može se primjetiti i druga funkcija metode lokalno konveksnih povezanih segmenata, a to je da se i objekti dodatno segmentiraju u dijelove. Dno kolica je podiljeno na dva dijela zbog objekata koji se nalaze na površini kolica i koji ih dijele na dva dijela. Osim toga ručke kolica su posebno segmentirane. Kante za zalijevanje su pak podijeljene na tijelo, s kojim su segmentirane ručke i na prednji dio koji se sastoji od cijevi za zalijevanje. Nedostatak u ovom procesu segmentacije bi bio, u slučaju da se želi postići segmentacija na dijelove, to što su ručke segmentirane zajedno s tijelom kante. Kao i u jednom od prethodnih primjera unutrašnjost šupljeg predmeta koja je vidljiva je posebno segmentirana te ne pripada istom segmentu kao i ostatak tijela kante.

Sl.5.35. Originalna slika



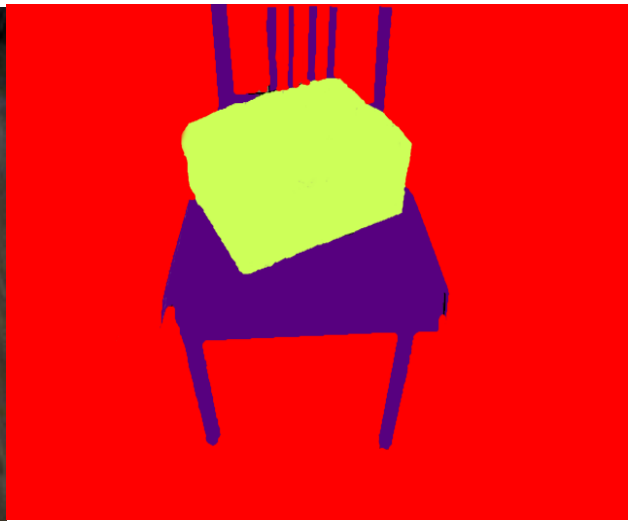
Sl.5.36. *Ground-truth* slika



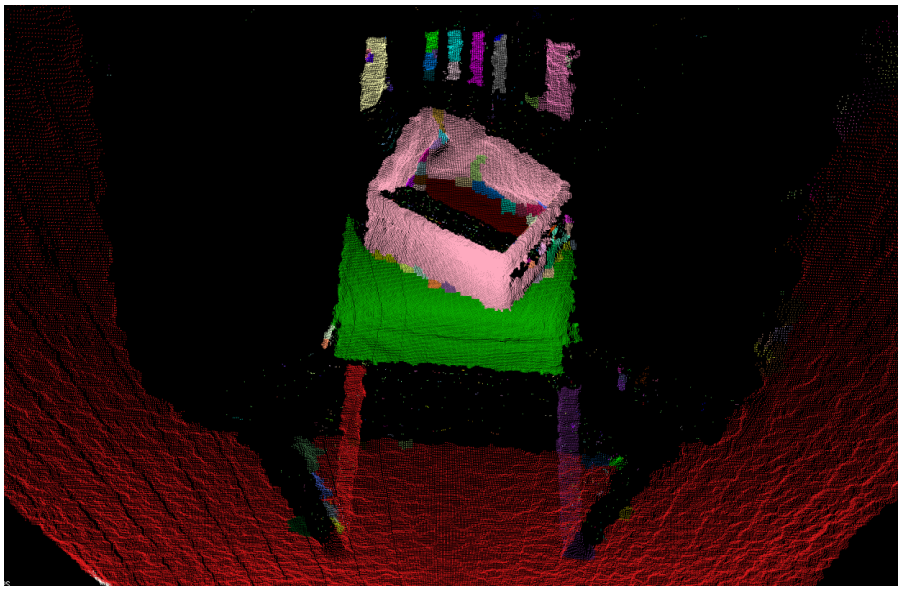
Sl.5.37. Slika segmentirana metodom lokalno konveksnih povezanih segmenata

Na priloženoj slici 5.37. prikazan je rezultat segmentacije slike 5.35.. Na slici su dva kanistera i jedna kolotura koji su uspješno prepoznati i segmentirani uz veliku preciznost. Kanisteri su jednodijelni pa su jednostruko i segmentirani iako se na manjem kanisteru pojavljuje jedan manji segment na jednom dijelu površine. Kolotura je uspješno segmentirana u više dijelova tako da svaki važniji sastavni dio iste predstavlja odvojeni segment.

Sl.5.38. Originalna slika



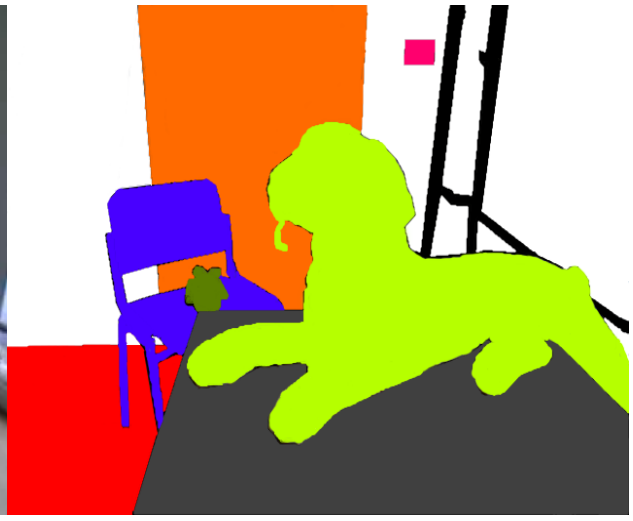
Sl.5.39. *Ground-truth* slika



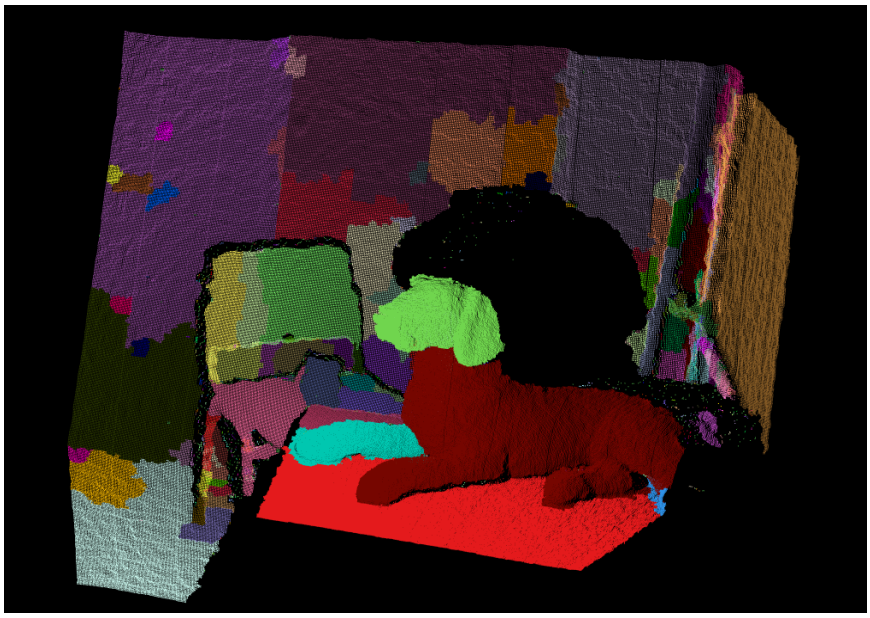
Sl.5.40. Slika segmentirana metodom lokalno konveksnih povezanih segmenata

Na slici 5.37. može se vidjeti rezultat segmentacije slike 5.35.. Vidljivo je da je stolica posebno segmentirana i podijeljena na više dijelova. Iako se pojavljuju manji segmenti na pojedinim dijelovima stolice svaki segment je vrlo precizno prikazan i segmentiran. Kutija koja se nalazi na stolici je također odvojeno segmentirana. Iako je vanjska površina kutije precizno segmentirana, moguće je vidjeti da se pojavio problem prilikom segmentacije unutrašnjosti kutije. U unutrašnjosti kutije kao i u prije promatranim šupljim predmetima pojavljuje se praznina koja je nastala prilkom segmentacije te se unutrašnja površina ne smatra za isti segment kao i ostatak kutije. Osim toga u unutrašnjosti kutije moguće je vidjeti više manjih segmenata nastalih zbog neprecizne segmentacije unutrašnjosti šupljeg predmeta.

Sl.5.41. Originalna slika



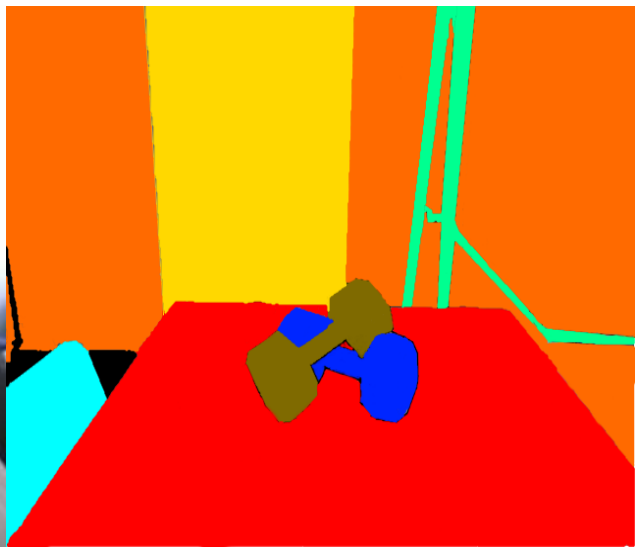
Sl.5.42. *Ground-truth* slika



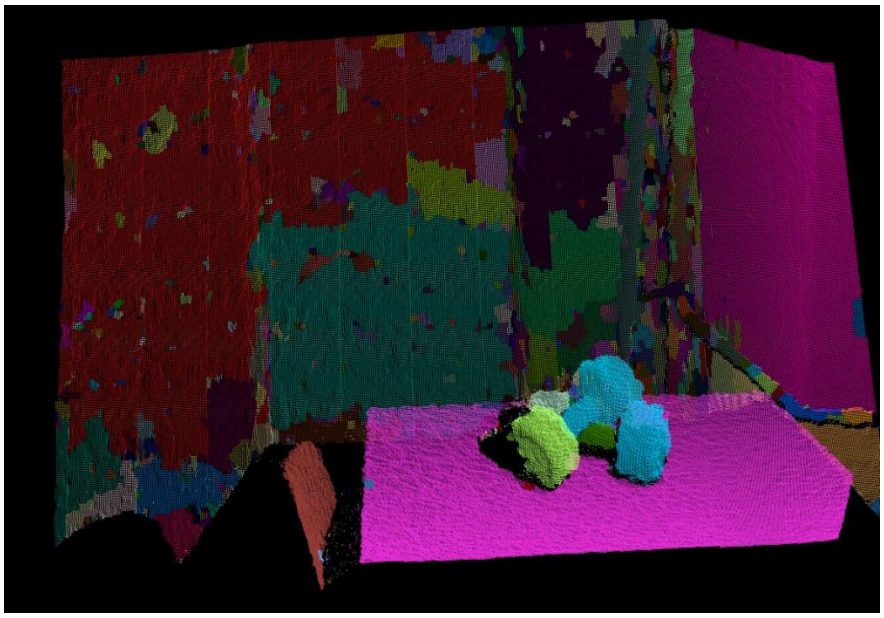
Sl.5.43. Slika segmentirana metodom lokalno konveksnih povezanih segmenata

Usporedbom slika 5.41., 5.42. i 5.43. moguće je vidjeti rezultat segmentacije sljedeće priložene slike. Slika sadrži složenu pozadinu koju je teže segmentirati zbog mjernog šuma, pa je zbog toga vidljivo da su pojedini jednostruki segmenti sa slike 5.41. na slici 5.43. segmentirani u više dijelova. Stolica je segmentirana odvojeno od pozadine i segmentirana u svoje dijelove. Igračka koja se nalazi na stolici precizno je segmentirana. Na slici se nalaze još i stol i plišani pas koji se nalazi na stolu. Pas je segmentiran u tri dijela: glava, tijelo i prednja desna noga dok su ostale noge segmentirane zajedno s tijelom.

Sl.5.44. Originalna slika



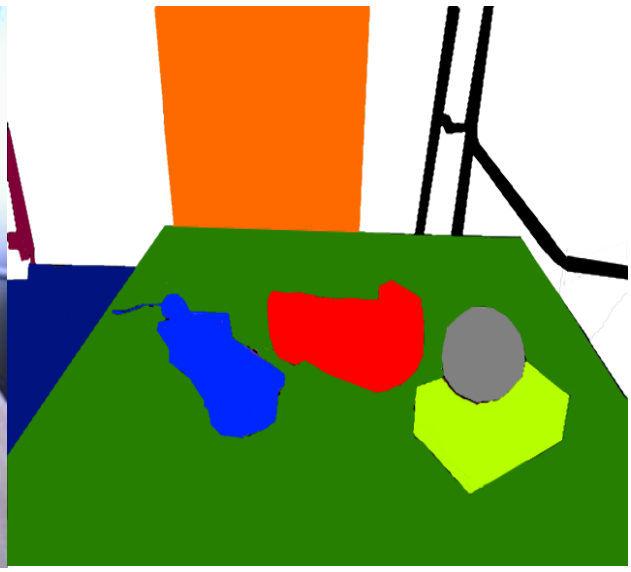
Sl.5.45. *Ground-truth* slika



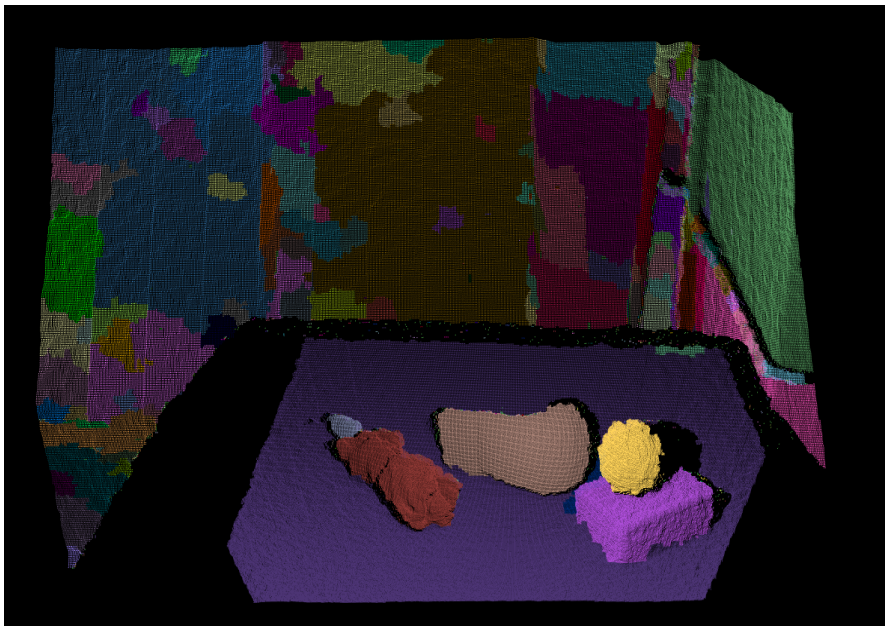
Sl.5.46. Slika segmentirana metodom lokalno konveksnih povezanih segmenata

Segmentacijom priložene slike 5.44. dobije se rezultat priložen na slici 5.46..Kao što je i vidljivo pozadina slike je prilično lošije segmentirana. Razlog zbog kojega se javlja *oversegmentacija* pozadine je mjerni šum čiji utjecaj raste s udaljenošću od kamere. Ipak, objekti u fokusu kao što su stol i stolica pored njega su precizno i uspješno segmentirani. Utezi koji se nalaze na stolu su također uspješno segmentirani, ali je vidljivo da je dio donjeg utega segmentiran zajedno s dijelom gornjeg utega zbog toga što su ta dva utega polegnuti jedan na drugi što metodi otežava prepoznavanje različitih predmeta.

Sl.5.47. Originalna slika



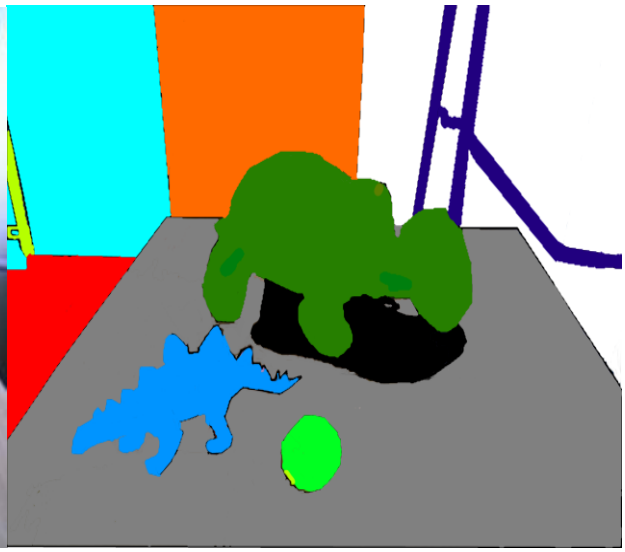
Sl.5.48. *Ground-truth* slika



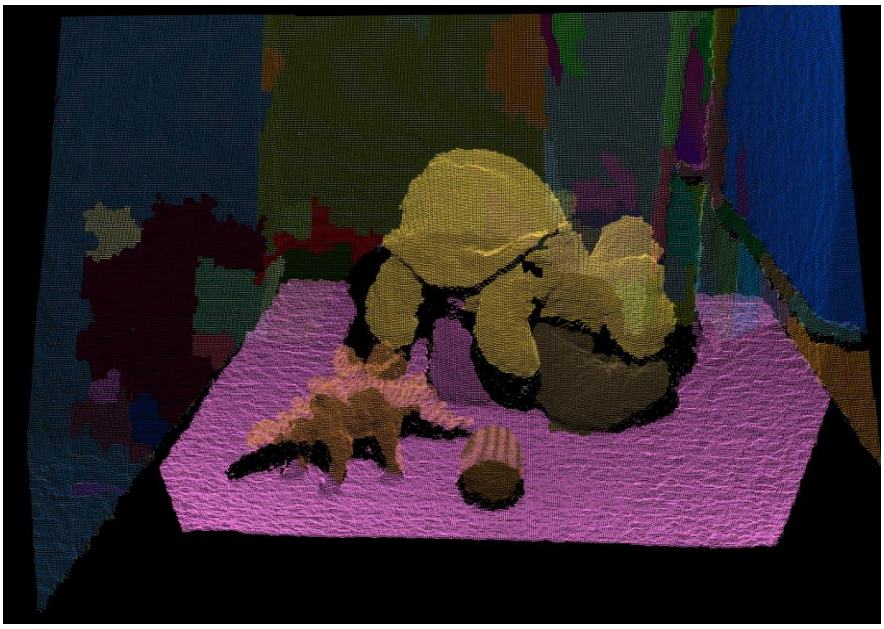
Sl.5.49. Slika segmentirana metodom lokalno konveksnih povezanih segmenata

Prema slici 5.49. koja predstavlja rezultat segmentacije slike 5.47. može se zaključiti da je metoda lokalno konveksnih povezanih segmenata rezultirala time da su svi objekti na slici prepoznati i vrlo precizno segmentirani. Pozadina je i ovdje slabije segmentirana zbog utjecaja mjernog šuma. Predmeti na stolu su precizno i točno segmentirani, a kišobran je segmentiran tako da su tijelo i ručka kišobrana odvojeno segmentirani. Loptica, spužva i rukavica su također točno segmentirani.

Sl.5.50. Originalna slika



Sl.5.51. *Ground-truth* slika

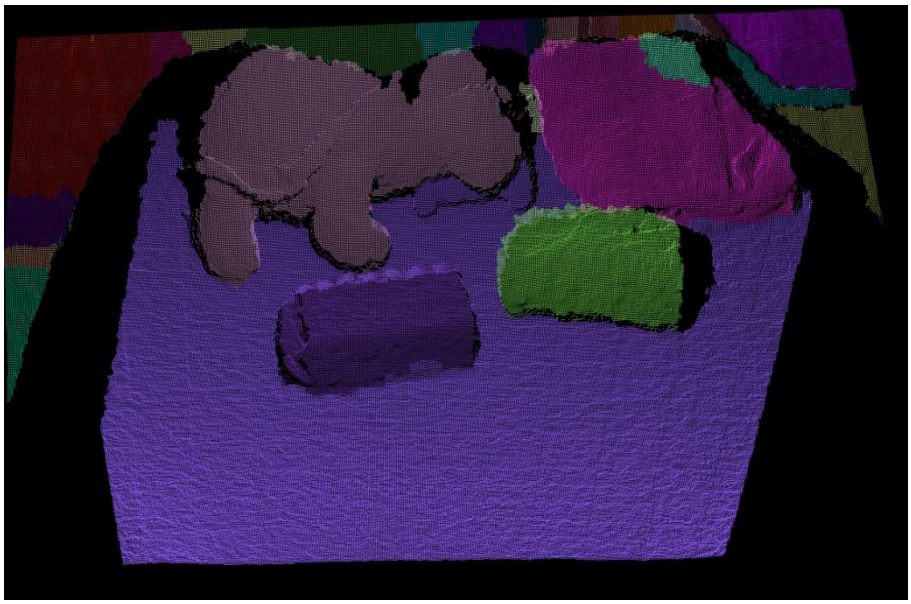


Sl.5.52. Slika segmentirana metodom lokalno konveksnih povezanih segmenata

Rezultat segemntacije sljedeće priložene slike 5.50. vidljiv je na slici 5.52.. Segmentacija predmeta je dala vrlo zadovoljavajuće rezultate jer je vidljivo da su svi predmeti precizno i uspješno segmentirani. Pozadina je vrlo dobro segmentirana iako su ponovno vidljiva mala odstupanja. Plišana kornjača nije dalje segmentirana u sastavne dijelove u slučaju da se traži segmentacija predmeta u dijelove.

Sl.5.53. Originalna slika

Sl.5.54. *Ground-truth* slika

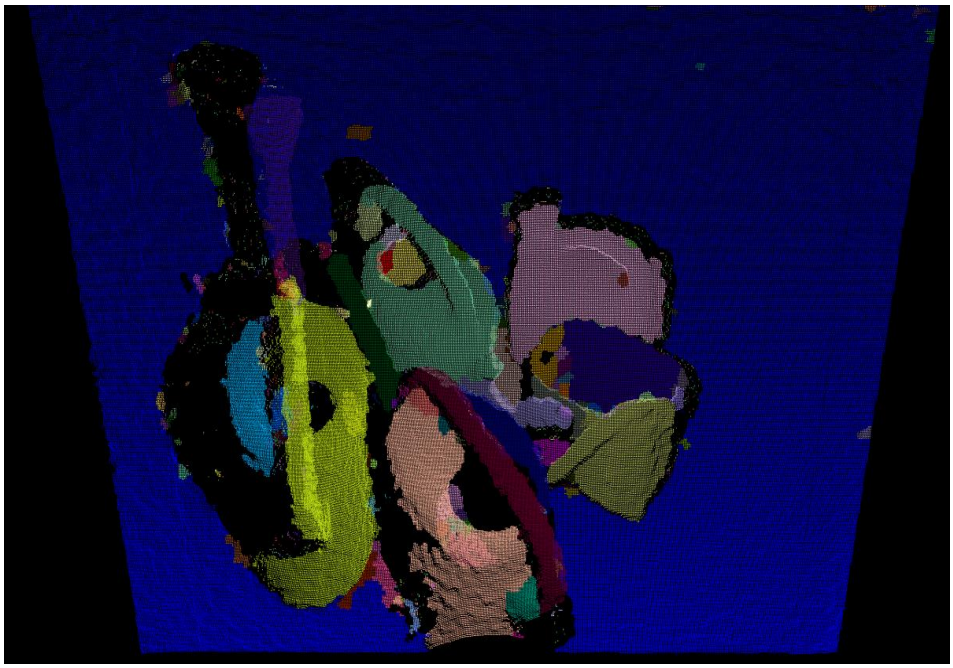


Sl.5.55. Slika segmentirana metodom lokalno konveksnih povezanih segmenata

Analizom priloženih slika 5.54. i 5.55. može se vidjeti rezultat segmentacije priložene slike 5.53.. Pozadina koja je segmentirana je nešto jednostavnija nego u prethodnim primjerima pa je i nešto točnije segmentirana. Uteg koji se nalazi na stolu je segmentiran istom bojom kao i stol dok su ostali predmeti točno i precizno segmentirani. Plišana kornjača je još jednom jednodijelno segmentirana.

Sl.5.56. Originalna slika

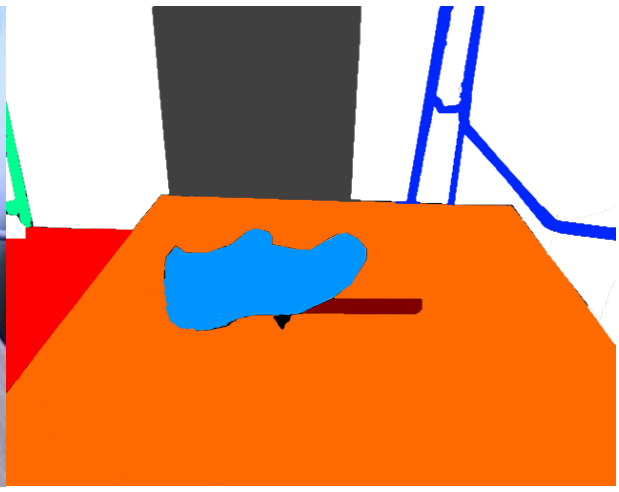
Sl.5.57. *Ground-truth* slika



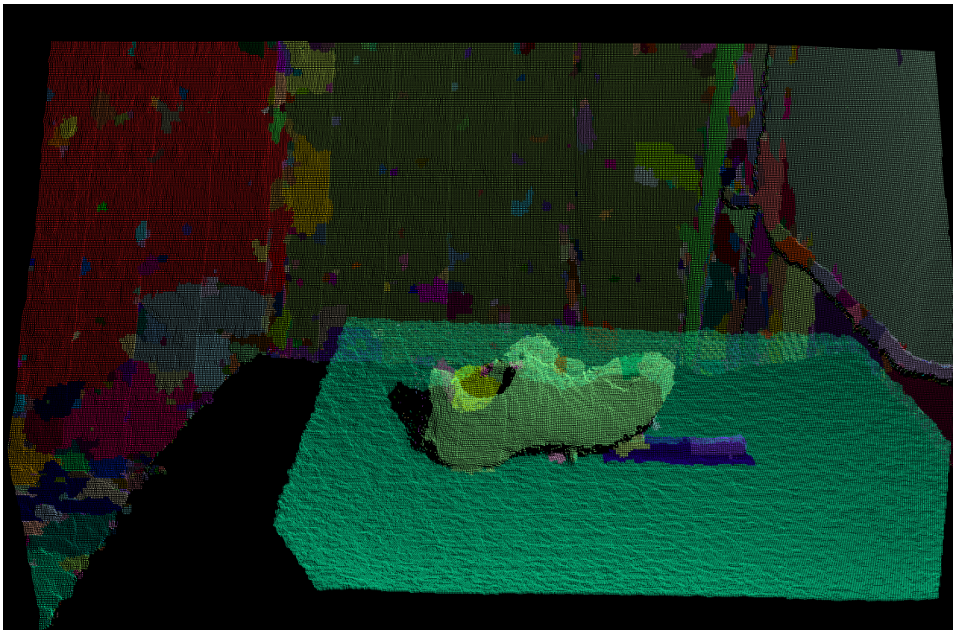
Sl.5.58. Slika segmentirana metodom lokalno konveksnih povezanih segmenata

Prema slici 5.58. slika 5.56. je segmentirana u dijelove. Različiti objekti su točno segmentirani u odnosu na pozadinu. Jednodijelni kanister je segmentiran u jednome dijelu dok se ostali objekti sastoje od više segmenata. Unutrašnjost šuplje kante je drugačije segmentirana od vanjske površine što se pojavljivalo i prilikom segmentacije drugih supljih objekata. Na slici su još i tri kante za zalijevanje gdje je samo jednoj kanti odijeljena ručka od tijela kante. Cijevi za zalijevanje su odvojeno segmentirane a vidljivo je kante koje ručka dijeli na dva dijela imaju različito segmentirane dijelove.

Sl.5.59. Originalna slika



Sl.5.60. *Ground-truth* slika



Sl.5.61. Slika segmentirana metodom lokalno konveksnih povezanih segmenata

Na slici 5.61. je prikazan rezultat segmentacije slike 5.59.. Vidljiva je *oversegmentacija* pozadine zbog mjernog šuma. Cipela koja se nalazi na stolu je precizno segmentirana osim segmenta koji predstavlja unutrašnjost cipele. Ručku čekića je bilo teže segmentirati jer se ona vrlo malo izdiže iznad površine na kojoj je polegnuta. Zbog toga je vidljiva *oversegmentacija* ručke čekića i razdvojenost segmenata kojima se ista prikazuje.

# 6. ZAKLJUČAK

Cilj ovoga rada je opisati i testirati metodu lokalno konveksnih povezanih segmenata koja se koristi u procesu segmentacije dubinske slike. Metoda se bazira na svojstvima konveksnosti i konkavnosti objekata. Algoritam metode je implementiran pomoću razvojnog okruženja Microsoft Visual Studio 2013., biblioteke Point Cloud Library i programskog jezika C++. Metoda je primjenjena na nekoliko dubinskih slika priloženih u radu. Analizom dobivenih rezultata zaključeno je da metoda lokalno konveksnih povezanih segmenata, iako vrlo jednostavna za implementaciju i brza za izvođenje, postiže sasvim zadovoljavajuće rezultate u procesu segmentacije dubinske slike. Može se koristiti za segmentaciju objekata koji se nalaze na slici, ali također i za dodatnu segmentaciju objekata na njihove sastavne dijelove. Metoda je posebno učinkovita za segmentiranje jednostrukih objekata na slikama s jednostavnijim pozadinama. Iako su postignuti rezultati usporedivi s rezultatima znatno složenijih *state-of-the-art* algoritama za segmentaciju slike, kod većine priloženih slika nerijetko su se pojavljivali manji segmenti koji predstavljaju šumove na slici, ali su njihove dimenzije često bile zanemarive u odnosu na dimenzije segmentiranih predmeta. Metoda ima problem s objektima koji se vrlo malo izdižu iznad površine na kojoj stoje jer ih često segmentira zajedno s površinom na kojoj stoje jer ih prepozna kao konkavne površine. Također, vidljiv je i utjecaj mjernog šuma koji je proporcionalan s udaljenošću od kamere, zbog čega je pozadina u nekoliko primjera bila presegmentirana. Segmentacijom šupljih predmeta, kao što su šalice, kutije i kante moguće je primjetiti da se unutrašnjost tih predmeta segmentira odvojeno od vanjske površine ili se uopće niti ne prepoznaje. Osim toga, pokazalo se da segmentacija uvelike ovisi o izboru parametara, pa je zbog toga moguće da se u određenim situacijama dobiju i nešto slabiji rezultati. Usprkos tome, zbog jednostavnosti metode i brzine izvođenja ona prestavlja vrlo učinkovitu i praktično primjenjivu metodu za segmentaciju.

# 7. LITERATURA

[1]S.C.Stein,F. Worgotter,M. Schoeler,J. Papon,T. Kulvicius: Convexity based object partitioning for robot applications, Proceedings of 2014 IEEE International Conference on Robotics and Automation, pp. 3213-3220 (2014.)

[2] S.C. Stein, M. Schoeler, J. Papon, F. Worgotter: Object partitioning using local convexity, in Computer Vision and Pattern Recognition (CVPR); 2014 IEEE Conference on. IEEE 2014, pp.304-311

[3] R.B. Rusu,S. Cousins: 3D is here:Point Cloud Library (PCL), Robotics and Automation (ICRA),2011 IEEE International Conference on, page1-4.(May 2011)

[4]https://www.fer.unizg.hr/_download/repository/09-OI-SegmentacijaSlike%5B2%5D.pdf, lipanj 2017

[5] https://msdn.microsoft.com/en-us/library/hh438998.aspx ,lipanj 2017.

[6] ROBOTSKI VID Ak. God. 2014./2015. Predavanja: Prof.dr.sc Robert Cupec

[7] http://docs.pointclouds.org/trunk/classpcl_1_1_l_c_c_p_segmentation.html , lipanj 2017.

[8]https://github.com/PointCloudLibrary/pcl/blob/master/examples/segmentation/example_lccp_segmentation.cpp , lipanj 2017.

# 8. SAŽETAK

Opisana je metoda lokalno konveksnih povezanih segmenata, te je ispitana njezina primjena za segmentaciju dubinske slike na objekte. Metoda lokalno konveksnih povezanih segmenata se koristi za segmentaciju slike na osnovu svojstava konveksnosti i konkavnosti objekata. Primjena te metode omogućava dekompoziciju scene na različite objekte što je vrlo važan zadatak robotike. Algoritam se temelji na kreiranju grafa susjedstva supervoksela čiji su čvorovi supervokseli i gdje se promatraju susjedni odnosi između supervoksela koji se mogu promatrati kao konveksne ili konkavne veze. Tada se detektiraju povezani podgrafovi koji predstavljaju segmente povezane konveksnim vezama. Algoritam je izveden koristeći C++ programski jezik u programskom okruženju Microsoft Visual Studio 2013. te PCL biblioteku koja omogućava trodimenzionalnu vizualizaciju slike. Programu su priložene RGB-D slike, a analizom dobivenih rezultata segmentacije moguće je uočiti veliku učinkovitost metode lokalno konveksnih povezanih segmenata u procesu segmentacije slike.

Ključne riječi: segmentacija slike, konveksnost, dubinska slika, supervoksel, PCL biblioteka, C++ programski jezik

# 9. ABSTRACT

A method of locally convex connected patches was described and its application for depth image segmentation is evaluated. The method of locally convex connected patches is used for depth image segmentation based on object convexity or concavity. This method is applicable in robotics and it enables scene decomposition into objects. The considered algorithm creates an adjacency graph of supervoxels, where every patch represents one supervoxel. Edges in the graph are then classified as either convex or concave. Then it detects subgraphs which represent segments. The algorithm is written in C++ programming language and it was implemented in Microsoft Visual Studio 2013. software. PCL library is used for three-dimensional image visualisation. The program was tested using RGB-D images representing the input to the segmentation algorithm. According to the obtained results, the method of locally convex connected patches is shown to be a very effective and reliable.

Keywords: image segmentation, convexity, depth image, supervoxel, pcl library, c++ programming language

# 10. ŽIVOTOPIS

Srđan Ljepić rođen je 8. ožujka 1996. u Vukovaru. Završio je Osnovnu školu „Tenja“ u Tenji. Nakon toga upisuje II. gimnaziju u Osijeku koju uspješno završava 2014. godine. Iste godine upisuje preddiplomski smjer Računarstva na Fakultetu Elektrotehnike, Računarstva i Infromacijskih Tehnologija gdje trenutno pohađa treću godinu studija.

# 11. PRILOZI

Prilog 1: Programski kod

```
#include <stdlib.h>
#include <cmath>
#include <limits.h>
#include <boost/format.hpp>
// PCL input/output biblioteke
#include <pcl/console/parse.h>
#include <pcl/io/pcd_io.h>
#include <pcl/visualization/pcl_visualizer.h>
#include <pcl/visualization/point_cloud_color_handlers.h>
#include <pcl/filters/passthrough.h>
#include <pcl/segmentation/supervoxel_clustering.h>
#include <pcl/segmentation/lccp_segmentation.h>
// VTK biblioteke
#include <vtkImageReader2Factory.h>
#include <vtkImageReader2.h>
#include <vtkImageData.h>
#include <vtkImageFlip.h>
#include <vtkPolyLine.h>

typedef pcl::PointXYZRGBA PointT;  // Koristi se za unos točaka
typedef pcl::LCCPSegmentation<PointT>::SupervoxelAdjacencyList SuperVoxelAdjacencyList;
bool show_supervoxels = false;

int main(int argc,char ** argv)
{
	bool show_visualization = (!pcl::console::find_switch(argc, argv, "-novis"));
	bool ignore_provided_normals = pcl::console::find_switch(argc, argv, "-nonormals");
	bool add_label_field = pcl::console::find_switch(argc, argv, "-add");
	pcl::PointCloud<PointT>::Ptr input_cloud_ptr(new pcl::PointCloud<PointT>);
	pcl::PointCloud<pcl::Normal>::Ptr input_normals_ptr(new pcl::PointCloud<pcl::Normal>);
	  bool has_normals = false;

	  std::string pcd_filename = argv[1];
	  PCL_INFO("Ucitavanje oblaka tocaka\n");
// provjeravamo postoje li vektori normale u priloženoj .pcd datoteci
	  pcl::PCLPointCloud2 input_pointcloud2;
	  if
(pcl::io::loadPCDFile("C:/Users/Srđan/Desktop/zavrsni/SLIKA.pcd",input_pointcloud2)) ///
putanja do slike je proizvoljna u ovisnosti o slici koju želimo koristiti za segmentaciju
	  {
		PCL_ERROR("ERROR: Ulazni oblak tocaka nije moguce ucitati %s.\n",
pcd_filename.c_str());
		return (3);
	  }
	  pcl::fromPCLPointCloud2(input_pointcloud2, *input_cloud_ptr);
	  if (!ignore_provided_normals)
```

```
	{
if (pcl::getFieldIndex(input_pointcloud2, "normal_x") >= 0)
		{
			pcl::fromPCLPointCloud2(input_pointcloud2, *input_normals_ptr);
			has_normals = true; //vektori normale postoje unutar .pcd datoteke
			if (input_normals_ptr->sensor_orientation_.w() == 0)
			{
				input_normals_ptr->sensor_orientation_.w() = 1;
				input_normals_ptr->sensor_orientation_.x() = 0;
				input_normals_ptr->sensor_orientation_.y() = 0;
				input_normals_ptr->sensor_orientation_.z() = 0;
			}
		}
		else
	PCL_WARN("Nisu pronadjene normale u pcd datoteci. Normale se racunaju.\n");
	}
	PCL_INFO("Oblak je uspjesno kreiran\n");

// Parametri supervoksela
	float voxel_resolution = 5.1f;
	float seed_resolution = 26.1f;
	float color_importance = 1.0f;
	float spatial_importance = 4.0f;
	float normal_importance = 4.0f;
	bool use_single_cam_transform =false;
	bool use_supervoxel_refinement = false;

	// Parametri LCCP segmentacije
	float concavity_tolerance_threshold = 10;
	float smoothness_threshold = 0.1;
	uint32_t min_segment_size = 0;
	bool use_extended_convexity =false;
	bool use_sanity_criterion = true;

use_single_cam_transform = pcl::console::find_switch(argc, argv, "-tvoxel");
	use_supervoxel_refinement = pcl::console::find_switch(argc, argv, "-refine");
	// Potrebno je parsirati argumente
	pcl::console::parse(argc, argv, "-v", voxel_resolution);
	pcl::console::parse(argc, argv, "-s", seed_resolution);
	pcl::console::parse(argc, argv, "-c", color_importance);
	pcl::console::parse(argc, argv, "-z", spatial_importance);
	pcl::console::parse(argc, argv, "-n", normal_importance);
	pcl::console::parse(argc, argv, "-ct", concavity_tolerance_threshold);
	pcl::console::parse(argc, argv, "-st", smoothness_threshold);
	use_extended_convexity = pcl::console::find_switch(argc, argv, "-ec");
	unsigned int k_factor = 0;
	if (use_extended_convexity)
		k_factor = 1; //kada je k>0 tada konveksne veze između segmenata pi i pj moraju
imati barem k zajeničkih susjeda koji imaju konveksnu vezu s oba segmenta
	use_sanity_criterion = pcl::console::find_switch(argc, argv, "-sc");
	pcl::console::parse(argc, argv, "-smooth", min_segment_size);
```

```
pcl::SupervoxelClustering<PointT> super(voxel_resolution, seed_resolution);
//Grupiranje supervoksela
super.setUseSingleCameraTransform(use_single_cam_transform);
	super.setInputCloud(input_cloud_ptr);
	if (has_normals)
		super.setNormalCloud(input_normals_ptr);
	super.setColorImportance(color_importance);
	super.setSpatialImportance(spatial_importance);
	super.setNormalImportance(normal_importance);
	std::map<uint32_t, pcl::Supervoxel<PointT>::Ptr> supervoxel_clusters;

	PCL_INFO("Izvlacenje supervoksela \n");
	super.extract(supervoxel_clusters);

	if (use_supervoxel_refinement)
	{
		PCL_INFO("Prerađivanje supervoksela \n");
		super.refineSupervoxels(2, supervoxel_clusters);
	}
	std::stringstream temp;
	temp << "  Broj supervoksela: " << supervoxel_clusters.size() << "\n";
	PCL_INFO(temp.str().c_str());

	PCL_INFO("Dohvacanje matrice susjedstva\n");
	std::multimap<uint32_t, uint32_t> supervoxel_adjacency;
	super.getSupervoxelAdjacency(supervoxel_adjacency);

	// Dohvaćanje oblaka supervoksela sa normalama i obojanih oblaka supervoksela
	pcl::PointCloud<pcl::PointNormal>::Ptr sv_centroid_normal_cloud =
	pcl::SupervoxelClustering<PointT>::makeSupervoxelNormalCloud(supervoxel_clusters);

	//Proces LCCP segmentacije

	PCL_INFO("Pocinjemo sa segmentacijom\n");
	pcl::LCCPSegmentation<PointT> lccp;
	lccp.setConcavityToleranceThreshold(concavity_tolerance_threshold);
	lccp.setSanityCheck(use_sanity_criterion);
	lccp.setSmoothnessCheck(true, voxel_resolution, seed_resolution,
smoothness_threshold);
	lccp.setKFactor(k_factor);
	lccp.setInputSupervoxels(supervoxel_clusters, supervoxel_adjacency);
	lccp.setMinSegmentSize(min_segment_size);
	lccp.segment();

	PCL_INFO("Interpolacija oblaka voksela\n");
	pcl::PointCloud<pcl::PointXYZL>::Ptr sv_labeled_cloud = super.getLabeledCloud();
	pcl::PointCloud<pcl::PointXYZL>::Ptr lccp_labeled_cloud = sv_labeled_cloud-
>makeShared();
	lccp.relabelCloud(*lccp_labeled_cloud);
	SuperVoxelAdjacencyList sv_adjacency_list;
	lccp.getSVAdjacencyList(sv_adjacency_list);  // Koristi se za vizualizaciju
```

```
if (show_visualization)
        {
        // Vizualizacija grafa susjednih supervoksela
using namespace pcl;
                typedef LCCPSegmentation<PointT>::VertexIterator VertexIterator;
                typedef LCCPSegmentation<PointT>::AdjacencyIterator AdjacencyIterator;
                typedef LCCPSegmentation<PointT>::EdgeID EdgeID;

                std::set<EdgeID> edge_drawn;

                const unsigned char convex_color[3] = { 255, 255, 255 };
                const unsigned char concave_color[3] = { 255, 0, 0 };
                const unsigned char* color;
        // Supervokseli iz liste susjednih supervoksela su središta supevoksela

        //Brojač prilazi kroz supervoksele i traži rubove

        std::pair<VertexIterator, VertexIterator> vertex_iterator_range;
        vertex_iterator_range = boost::vertices(sv_adjacency_list);

        // Kreira se oblak voksela i obavlja se indeksiranje točaka

        vtkSmartPointer<vtkPoints> points = vtkSmartPointer<vtkPoints>::New();
        vtkSmartPointer<vtkCellArray> cells = vtkSmartPointer<vtkCellArray>::New();
        vtkSmartPointer<vtkUnsignedCharArray> colors =
        vtkSmartPointer<vtkUnsignedCharArray>::New();
                colors->SetNumberOfComponents(3);
                colors->SetName("Colors");
        // Kreira se polydata u koju se spremaju podatci

        vtkSmartPointer<vtkPolyData> polyData = vtkSmartPointer<vtkPolyData>::New();
                for (VertexIterator itr = vertex_iterator_range.first; itr !=
                vertex_iterator_range.second; ++itr)
                {
                const uint32_t sv_label = sv_adjacency_list[*itr];
                std::pair<AdjacencyIterator, AdjacencyIterator> neighbors =
                boost::adjacent_vertices(*itr, sv_adjacency_list);

        for (AdjacencyIterator itr_neighbor = neighbors.first; itr_neighbor != neighbors.second;
        ++itr_neighbor)
                {
        EdgeID connecting_edge = boost::edge(*itr, *itr_neighbor, sv_adjacency_list).first;
        //Dohvaćanje ruba koji povezuje supervoksele
if (sv_adjacency_list[connecting_edge].is_convex)
color = convex_color;
else
color = concave_color;
colors->InsertNextTupleValue(color);
colors->InsertNextTupleValue(color);
```

```
pcl::Supervoxel<PointT>::Ptr supervoxel = supervoxel_clusters.at(sv_label);
pcl::PointXYZRGBA vert_curr = supervoxel->centroid_;
const uint32_t sv_neighbor_label = sv_adjacency_list[*itr_neighbor];
pcl::Supervoxel<PointT>::Ptr supervoxel_neigh = supervoxel_clusters.at(sv_neighbor_label);
pcl::PointXYZRGBA vert_neigh = supervoxel_neigh->centroid_;
points->InsertNextPoint(vert_curr.data);
points->InsertNextPoint(vert_neigh.data);
vtkSmartPointer<vtkPolyLine> polyLine = vtkSmartPointer<vtkPolyLine>::New();
                        polyLine->GetPointIds()->SetNumberOfIds(2);
                        polyLine->GetPointIds()->SetId(0, points->GetNumberOfPoints() - 2);
                        polyLine->GetPointIds()->SetId(1, points->GetNumberOfPoints() - 1);
                                cells->InsertNextCell(polyLine);
                        }
                }
                        polyData->SetPoints(points);

polyData->SetLines(cells);

                polyData->GetPointData()->SetScalars(colors);
        //Kraj računanja vizualizacije grafa susjedstva

        // Postavljanje preglednika

pcl::visualization::PCLVisualizer::Ptr viewer(new pcl::visualization::PCLVisualizer("3D
Viewer"));
                viewer->setBackgroundColor(0, 0, 0);

                viewer->addPointCloud(lccp_labeled_cloud, "maincloud");
                while (!viewer->wasStopped())
                {
                        viewer->spinOnce(100);


                                viewer->updatePointCloud((show_supervoxels) ? sv_labeled_cloud
: lccp_labeled_cloud, "maincloud");
                        }

}  /// Kraj if-a (show_visualization)

        return (0);

        }  // Kraj maina
```