Robotska manipulacija primjenom računalnog vida

Mihaljević, Maja

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

Sveučilišni studij

ROBOTSKA MANIPULACIJA PRIMJENOM RAČUNALNOG VIDA

Diplomski rad

Maja Mihaljević

Osijek, 2018.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada

Osijek, 22.09.2018.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu diplomskog rada

Ime i prezime studenta:	Maja Mihaljević
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D-804R, 25.09.2017.
OIB studenta:	43363858257
Mentor:	Prof.dr.sc. Robert Cupec
Sumentor:	Petra Đurović
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Doc.dr.sc. Emmanuel-Karlo Nyarko
Član Povjerenstva:	Petra Đurović
Naslov diplomskog rada:	Robotska manipulacija primjenom računalnog vida
Znanstvena grana rada:	**Procesno računarstvo (zn. polje računarstvo)**
Zadatak diplomskog rada:	Izraditi računalni program koji na temelju dubinske slike scene snimljene 3D kamerom prepoznaje objekt od interesa, određuje njegov položaj u odnosu na koordinatni sustav robota i obavlja zadatak hvatanja objekta robotskom rukom i premještanja na željenu poziciju. (Sumentor: Petra Đurović)
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 2 razina
Datum prijedloga ocjene mentora:	22.09.2018.

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, 23.09.2018.

Ime i prezime studenta:	Maja Mihaljević
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D-804R, 25.09.2017.
Ephorus podudaranje [%]:	1%

Ovom izjavom izjavljujem da je rad pod nazivom**: Robotska manipulacija primjenom računalnog vida**

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

i sumentora Petra Đurović

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

Robotski su se manipulatori počeli izrađivati krajem pedesetih godina prošlog stoljeća s ciljem odrađivanja poslova opasnih po život i ljudsko zdravlje u industrijskim postrojenjima [1]. Danas se industrijski roboti koriste za poslove koji se ponavljaju te koji zahijevaju veliku brzinu i preciznost u strogo kontroliranim uvjetima, odnosno u strukturiranoj okolini. U skorijoj budućnosti očekuje se da roboti budu što samostalniji, odnosno da budu sposobni obavljati zadatke i u dinamičnoj i nestrukturiranoj okolini, te da budu što jeftiniji kako bi se njihova primjena proširila na kućanstva te znanstvene i medicinske ustanove [2].

Razvojem vizualnih senzora, posebice stereo i RGB-D kamera, počinje i interes i razvoj u računalnom vidu te korištenje 3D vizije kao načina percepcije okoline. Danas na tržištu postoji mnoštvo jeftinih 3D senzora koji daju relativno gustu dubinsku sliku iz koje se mogu izvući trodimenzionalni podaci o okolini potrebni za lokalizaciju robota i/ili prepoznavanje objekata u okolini.

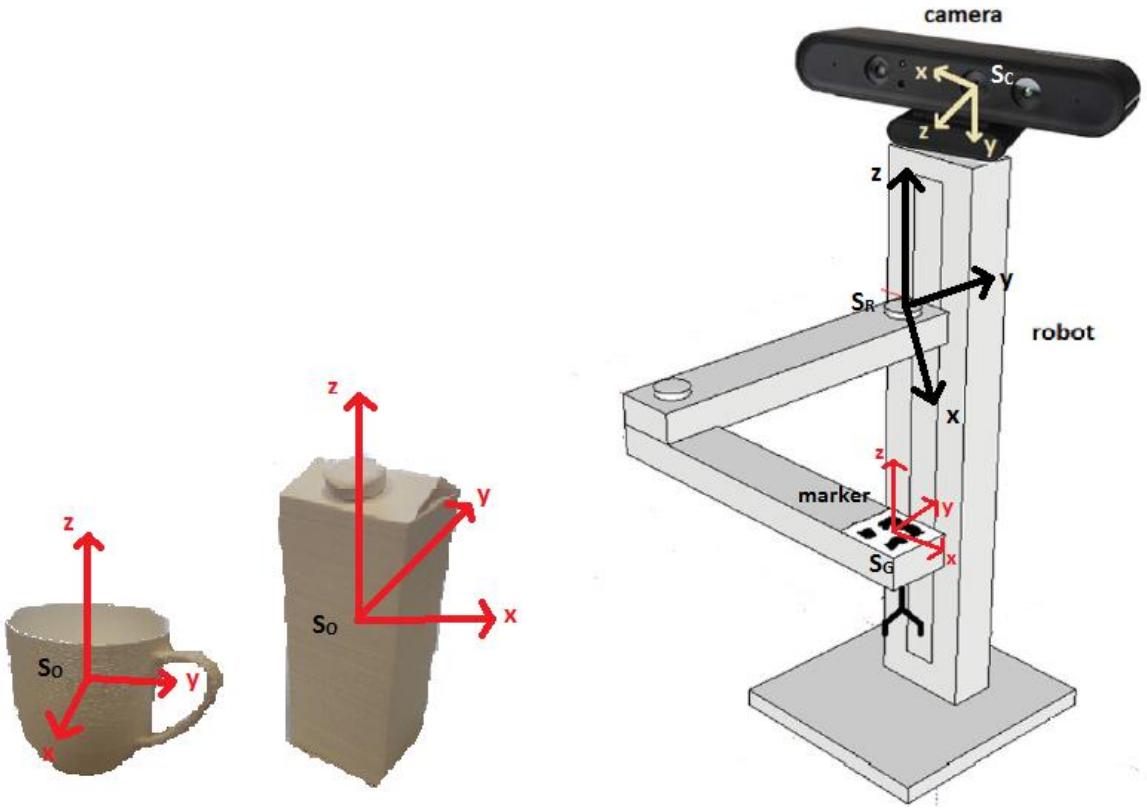
Cilj diplomskog rada jest pomoću takvog jednog vizualnog senzora detektirati objekt od interesa na sceni, pozicionirati alat iznad objekta koristeći vizualni servoing, uhvatiti ga s odgovarajućom hvataljkom te prebaciti na željeno odredište. Trenutni položaj robota određuje se detekcijom i lokalizacijom markera koji je montiran iznad alata, a ciljni je položaj određen detekcijom objekta od interesa na sceni. Koraci pri hvatanju objekta uključuju kalibraciju kamere i robota, određivanje svih položaja i robota i objekta u koordinatnom sustavu kamere, pozicioniranje alata iznad objekta te rotaciju hvataljke za određeni kut.

U drugom poglavlju ovog rada obrađuje se metoda korištena za kalibraciju kamere i robota kao i metoda vizualnog servoinga potrebnog za navođenje robotske ruke do ciljnog položaja. Također je opisano planiranje hvatanja određenog objekta koje ovisi o njegovom obliku i veličini kao i o vrsti hvataljke koja se koristi. Treće poglavlje predstavlja programsku implementaciju predloženog načina hvatanja. Eksperimentalna evaluacija opisanih metoda prikazana je u četvrtom poglavlju, dok peto, a ujedno i posljednje poglavlje, predstavlja zaključak cijelog diplomskog rada.

2. METODE RAČUNALNOG I ROBOTSKOG VIDA ZA ROBOTSKU MANIPULACIJU

U ovom je poglavlju dan pregled korištene metode kalibracije kamere i robota kao i korištene metode vizualnog servoinga potrebnog za navođenje robotske ruke od trenutnog do ciljnog položaja. Objašnjen je i postupak planiranja hvatanja određenog objekta, koji ovisi o obliku i veličini objekta kao i o vrsti hvataljke koja se koristi.

Na slici 2.1. prikazan je cjelokupni sustav koji se sastoji od robotske ruke, kamere te objekata kojima robot manipulira. Na slici su također prikazani i svi koordinarni sustavi koje je potrebno definirati zbog kalibriracije kamere i robotske ruke, vizualnog serovinga te na kraju hvatanja zadanog objekta. Koordinatni sustavi prikazani na slici predstavljaju koordinatni sustav kamere – S_C, koordinatni sustav robota – S_R, koordinatni sustav hvataljke – S_G te koordinatni sustav objekta – S_O.

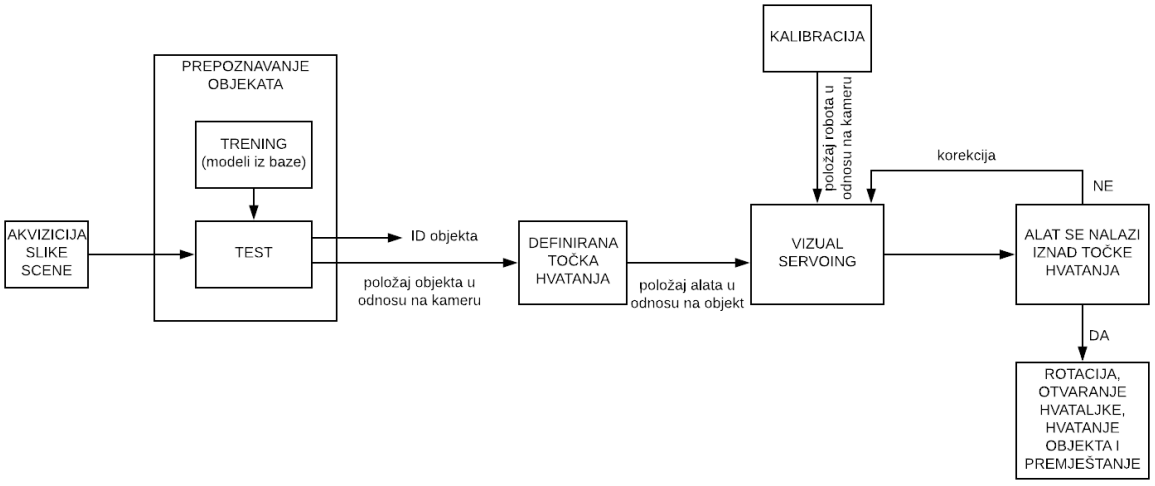


Slika 2.1. Prikaz cjelokupnog sustava potrebnog za robotsku manipulaciju

Na slici 2.2. prikazana je funkcionalna povezanost svih dijelova sustava. Nakon što kamera simi scenu, slika se šalje u dio sustava koji vrši prepoznavanje objekata na sceni. Može se vidjeti kako se dio sustava za prepoznavanje objekata na sceni sastoji od dva dijela. Prvi dio, ili „TRENING“ dio, kreira bazu modela objekata od interesa te određuje njihove značajke. Ovaj dio predstavlja pripremnu fazu, koja se izvodi prije testiranja. U drugom se dijelu, ili „TEST“ dijelu, određuje nalazi li se model iz baze na sceni, odnosno postavljaju se hipoteze koje se zatim evaluiraju. Program za prepoznavanje objekata na kraju vraća, uz položaj k.s. objekta u odnosu na k.s. kamere ${}^{C}\mathbf{T_O}$, i „ID objekta“, odnosno broj prepoznatog modela iz baze kada se računanje točke hvatanja u potpunosti predaje programu. U ovome radu „ID objekta“ nije korišten, budući da prepoznavanje objekata na sceni nije predmet razmatranja u ovome radu, već se na početku korisnika pita koji će objekt biti postavljen na sceni. Definiranjem točke hvatanja određuje se položaj k.s. hvataljke u odnosu na k.s. objekta ${}^{O}\mathbf{T_G}$. Kalibracija određuje odnos k.s. robota i k.s. kamere, ${}^{R}\mathbf{T_C}$, i nakon toga vizualni servoing navodi robota na ciljni položaj računajući položaj k.s. hvataljke u odnosu na k.s. robota na sljedeći način:

$$ {}^{R}\mathbf{T_G} = {}^{R}\mathbf{T_C} \cdot {}^{C}\mathbf{T_O} \cdot {}^{O}\mathbf{T_G} \tag{2-1} $$

Ako se hvataljka nalazi u neposrednoj blizini točke hvatanja, odnosno unutar zadane toleracije koja je definirana od strane korisnika, vizualni servoing završava te se hvataljka rotira za kut potreban kako bi se objekt mogao uhvatiti i premjestiti na odredište. Ako se, pak, hvataljka ne nalazi unutar zadane tolerancije od točke hvatanja, vizualni servoing vrši korekciju sve dok hvataljka ne dođe na položaj koji se nalazi unutar zadane tolerancije.



Slika 2.2. Funkcionalna povezanost različitih dijelova sustava

2.1. Prepoznavanje objekata na RGB-D slici

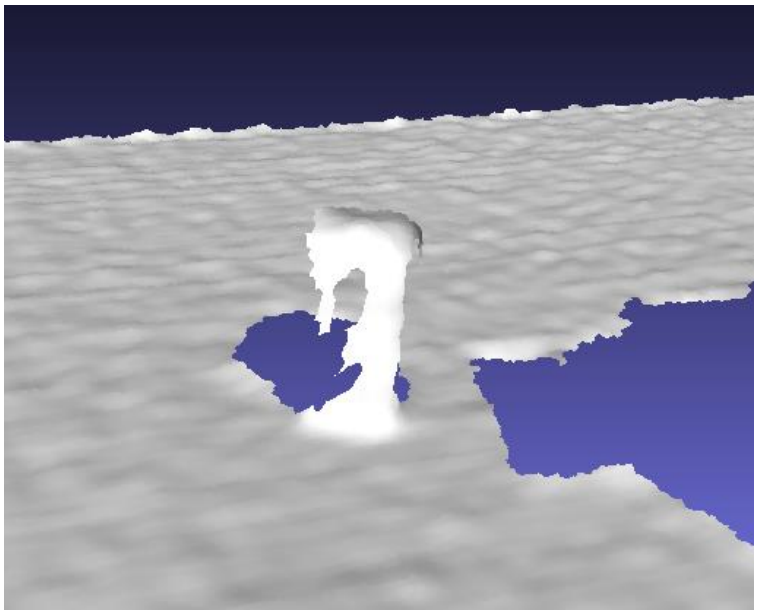
Prepoznavanje objekata jedno je od najvažnijih problema računalnog vida. Razvojem vizualnih senzora počinje i zanimanje za algoritme vezane uz prepoznavanje objekata koji rade s 3D podacima. Prema [3], problem prepoznavanja 3D objekata može se postaviti kako slijedi. Dan je skup modela objekata od interesa $M = \{M_1, M_2, ..., M_n\}$ i scena S gdje su i modeli M_i i scena S predstavljeni skupovima 3D točaka. Potrebno je odrediti postoji li transformacija nekog modela iz skupa M takva da odgovara nekom podskupu scene S. Izlaz iz algoritma za prepoznavanje objekata je skup $\{(M_{k1}, T_1), ..., (M_{kr}, T_r)\}$, odnosno skup hipoteza, gdje je $M_{kj} \in M$ prepoznata instanca modela, a T_j je transformacija koja pozicionira M_{kj} na scenu S.

Metoda koja je predložena u [3] može se koristiti za prepoznavanje objekata na scenama koje su zašumljene kao i na scenama gdje su objekti od interesa zaklonjeni nekim drugim objektima. Na početku se objekti od interesa modeliraju u obliku 3D oblaka točaka. Zatim se određuju značajke modela. Značajke korištene u ovoj metodi su parovi orijentiranih točaka. Orijentirana točka je točka kojoj je pridružena normala na površinu objekta u neposrednoj blizini te točke. Hipoteze su generirane koristeći RANSAC algoritam [4]. Kako bi se ubrzao proces sparivanja značajki, kreira se *hash* tablica sa svim deskriptorima izračunatim za svaki par značajki modela. U fazi prepoznavanja, određuju se parovi značajki sa scene te se dohvaćaju deskriptori iz *hash* tablice koji najbolje opisuju taj par značajki. Evaluacija hipoteza vrši se na način da se podržavaju hipoteze u kojima se veći broj tranformiranih točaka modela podudara s točkama scene, a kažnjavaju se one hipoteze s manjim brojem točaka koje se podudaraju sa scenom ili s točkama koje zaklanjaju točke scene.

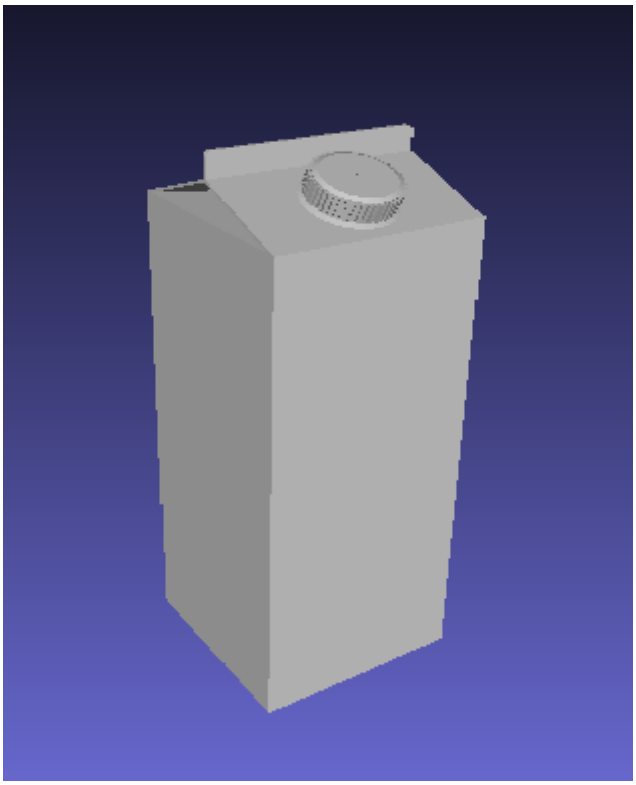
U metodi predloženoj u [5], za značajke se koriste 3D točke koje su ravnomjerno uzorkovane s površine modela i scene. Zatim su svakoj takvoj točki pridruženi SHOT [6] lokalni deskriptori. Hipoteze se generiraju koristeći *Correspondence Grouping* (CG) algoritam. CG algoritam grupira parove značajki sličnih deskriptora koji zadovoljavaju određena geometrijska ograničenja. Za svaku grupu parova značajki provodi se RANSAC algoritam koji rezultira hipotezom položaja modela. Taj dobiveni položaj se zatim poboljšava koristeći ICP algoritam. Svakoj hipotezi dodjeljuje se bool varijabla, odnosno vrijednost 1 ili 0. Ako je hipotezi dodijeljena vrijednost 1, tada je hipoteza ispravna, a ako joj je dodijeljena vrijednost 0, tada je hipoteza netočna. U fazi evaluacije hipoteza, koristi se *Simulated Annealing* algoritam koji minimizira funkciju troška.

Algoritam korišten u [7] započinje s određivanjem normala na oblaku 3D točaka. Određuju se smjerovi glavnih zakrivljenosti modela i FPFH značajke. Za svaki par orijentiranih značajki modela i scene, određuje se mogući položaj objekta koristeći *Bingham Procrustean Alignment* (BPA) algoritam. Nakon toga slijedi grupiranje parova značajki koje ovisi o udaljenosti jednog para od drugog, gdje se grupe od manje od 2 para značajki odbacuju. Ostali grupirani parovi značajki predstavljaju hipoteze mogućih položaja objekta. Druga faza potvrđivanja ili odbacivanja hipoteza koristi funkciju bodovanja. Za izračun te funkcije u obzir se uzimaju razlike u dubini između točaka modela i scene kao i razlike između normala, zatim omjer točaka modela koje nisu zaklonjene, podudaranje rubnih točaka te omjer rubnih točaka koje nisu zaklonjene. U zadnjoj se fazi evaluacije hipoteze rangiraju po bodovima te se ona s najvišom ocjenom smatra kao ispravna i postavlja se na scenu.

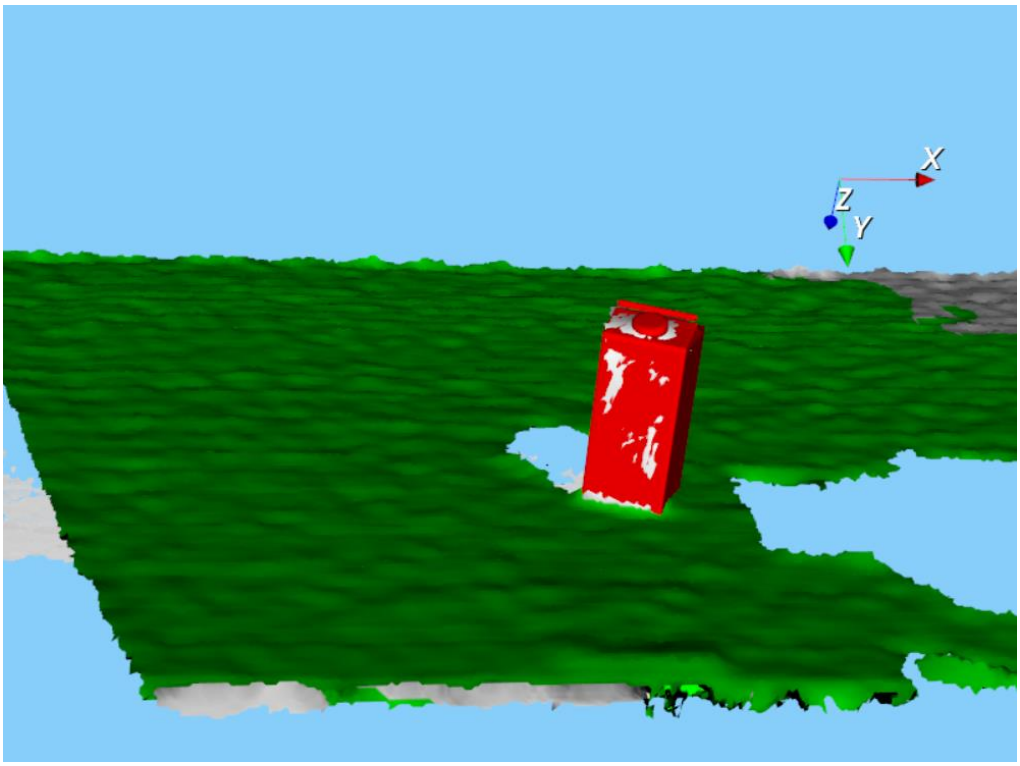
Za potrebe snimanja objekata na sceni u ovom radu, korištena je RGB-D *Orbbec Astra S* kamera [8] koja omogućuje 3D viziju. Za sve objekte od interesa napravljeni su 3D modeli u 3D CAD programu te oni tvore bazu koju će koristiti algoritam za prepoznavanje objekata na sceni. Korištena je unaprijeđena verzija metode prepoznavanja objekata [9] na slici, koja pronalazi hipoteze, odnosno sve potencijalne modele na sceni, a koji se nalaze u bazi. Nakon postavljanja hipoteza vrši se evaluacija čiji je rezultat, između ostalog, transformacijska matrica ishodišta modela u koordinatnom sustavu kamere $^{C}\mathbf{T}_{M}$, koja ujedno predstavlja položaj objekta u odnosu na kameru $^{C}\mathbf{T}_{O}$. Za potrebe hvatanja, smatra se da je hipoteza kojoj je algoritam prepoznavanja dodijelio najvišu ocjenu ispravna te se translacijski vektor te hipoteze smatra ishodištem koordinatnog sustava modela, odnosno njegovim centrom mase. Na slici 2.3. prikazan je oblak točaka scene i objekta na sceni. Slika 2.4. prikazuje CAD model objekta u bazi, a slika 2.5. prikazuje kako se prepoznati model (crveno) transformiran pomoću matrice homogene transformacije $^{C}\mathbf{T}_{M}$, koju daje algoritam za prepoznavanje objekata, podudara s objektom na sceni.



Slika 2.3. Prikaz oblaka točaka scene i objekta na sceni



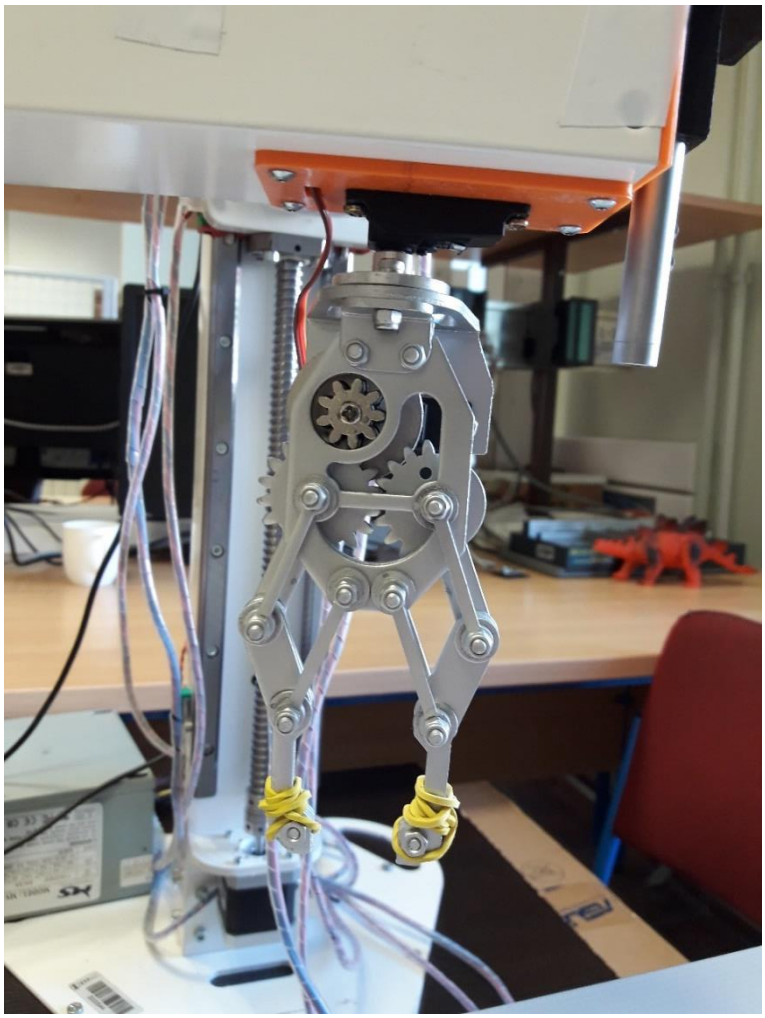
Slika 2.4. Prikaz CAD modela u bazi



Slika 2.5. Podudaranje prepoznatog modela s objektom na sceni

2.2. Planiranje hvatanja

U uvodu je objašnjeno da ciljni položaj, odnosno točka hvatanja na koju se želi dovesti hvataljka robota, ovisi o položaju predmeta na sceni kojeg se želi uhvatiti. To znači da nije moguće unaprijed isplanirati hvatanje koje će odgovarati svim predmetima, nego je potrebno isplanirati hvatanje za svaki predmet posebno ovisno o veličini i obliku tog predmeta te njegovoj poziciji i orijentaciji na sceni, ali i o tipu hvataljke koja se koristi (Sl.2.6.).



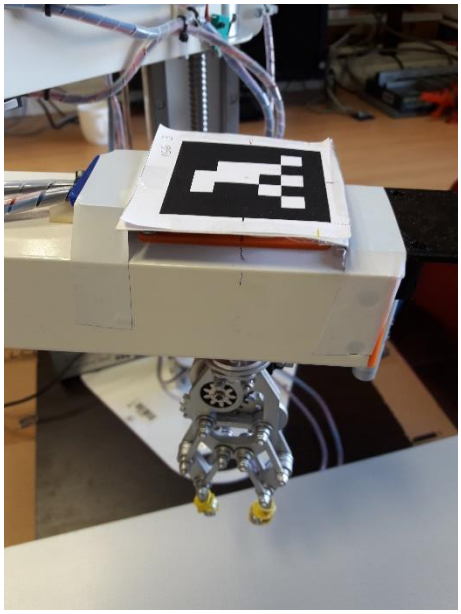
Slika 2.6. Hvataljka s jednim stupnjem slobode

Primjerice, za hvatanje tetrapaka hvataljkom prikazanom na slici 2.6. potrebno je, nakon detekcije tog objekta na sceni, odrediti u kojem se položaju on nalazi. Nakon što se odredi položaj objekta na sceni, potrebno je odrediti optimalnu točku hvatanja. Točka hvatanja trebala bi omogućiti ispravno hvatanje i prenošenje objekta na željenu destinaciju. U slučaju tetrapaka, prikladno je onda navesti alat iznad središta objekta i hvatati ga za vanjske stijenke. Dakle, da bi se isplaniralo hvatanje tetrapaka, potrebno je za početak odrediti središte objekta. Nakon što se hvataljka pozicionira iznad središta objekta, potrebno je izračunati optimalan kut rotacije hvataljke, kako bi se objekt mogao čvrsto uhvatiti. Ako su dimenzije hvataljke takve da vrh hvataljke ne može doseći sredinu objekta, potrebno je odrediti za koliko se hvataljka najviše može spustiti. Zadnji korak uključuje određivanje širine objekta koji se hvata jer o tome ovisi širina otvora hvataljke.

2.3. Pozicioniranje hvataljke vizualnim servoingom

U svrhu pojeftinjenja robota i robotskih manipulatora kako bi postali pristupačniji za svakodnevno korištenje, istražuju se roboti bazirani na koračnim motorima koji nemaju apsolutne enkodere te ne znaju apsolutne položaje svojih zglobova. Umjesto toga, koriste se vizualni senzori koji daju informacije potrebne za navođenje robotske ruke na ciljni položaj [10]. Vizualni podaci mogu biti dobiveni od vizualnog senzora koji je montiran direktno na robotski manipulator ili je senzor montiran fiksno negdje u radnom prostoru [10,11].

U ovom je slučaju vizualni senzor montiran fiksno u radnom prostoru iznad robotskog manipulatora i nadgleda njegove pokrete. Kako bi se robot doveo na ciljni položaj, koristi se vizualni servoing čija je svrha izračunavanje promjena vrijednosti varijabli zglobova potrebnih za pomicanje alata iz trenutnog u ciljni položaj. Trenutni položaj alata određuje se detekcijom i lokalizacijom markera koji je smješten na robotskoj ruci iznad središta alata (Sl. 2.7.). Položaj markera izračunava se u odnosu na k.s. kamere, a kasnije se i sve ostale pozicije, na koje se želi dovesti robot, zadaju u istom koordinatnom sustavu. Ciljni položaj robota ovisi o položaju predmeta kojeg se želi uhvatiti, a koji se nalazi u radnoj okolini robota [10].



Slika 2.7. Marker smješten iznad alata robotskog manipulatora

Korištena metoda vizualnog servoinga računa potrebne promjene u vertikalnom položaju, što se postiže pomicanjem prvog translacijskog zgloba, kao i promjene varijabli drugog i trećeg

rotacijskog zgloba. Ovaj se algoritam provodi iterativno sve dok marker ne dođe na ciljni položaj unutar zadane tolerancije [10].

Kako bi se dobili parametri potrebni za vizualni servoing, potrebno je provesti kalibraciju kamere i robota. Kalibracija kamere i robota nužna je kako bi se odredio odnos između njih, odnosno kako bi se odredio k.s. robota u k.s. kamere jer se, kako je već rečeno ranije, sve pozicije na koje se želi dovesti robot zadaju u k.s. kamere. Korištena metoda kalibracije provodi se u dvije faze. U prvoj se fazi određuje dominatna ravnina na kojoj se nalazi objekt od interesa. Prepoznavanje dominantne ravnine ključno je zbog određivanja orijentacije prvog rotacijskog zgloba robota. Druga faza uključuje pokret prvog rotacijskog zgloba robota za zadani kut, a rezultat toga je pomicanje markera s početne na zadanu poziciju, a te su pozicije dobivene preko programa za detekciju i lokalizaciju markera. Parametri dobiveni ovom metodom kalibracije dovoljni su za vizualni servoing. Ova metoda ne vraća transformacijsku matricu, već promjene kuteva zglobova robota jer je namijenjena relativnom pozicioniranju, odnosno bitno je da robot dođe iznad objekta, a nije potrebno znati apsolutnu poziciju objekta u k.s. robota [10].

3. IMPLEMENTACIJA RAČUNALNOG PROGRAMA

Programska izvedba opisanog sustava implementirana je u C++ jeziku u *Visual Studio 2013* razvojnom okruženju. Za upravljanje robotskim manipulatorom pozivaju se funkcije pisane u Pythonu. Za pokretanje kamere i snimanje scene korišten je *Astra SDK* [8]. Za dohvaćanje i obradu informacija sa slike, korištena je *OpenCV* programska biblioteka. *ArUco* biblioteka [12], koja se temelji na *OpenCV* biblioteci, korištena je za detekciju markera postavljenog na alat, a za prepoznavanje objekata na sceni koriste se funkcije razvijene unutar RVL (engl. *Robot Vision Library*) biblioteke[1]. Za neke matematičke proračune korištena je biblioteka *Eigen* [13]. U tablici 3.1. prikazano je koje direktorije i biblioteke treba uključiti u projekt kako bi se program mogao pokrenuti i izvršiti.

Tablica 3.1. Popis uključenih direktorija i biblioteka

Additional Include Directories	**Additional Library Directories**	**Additional Dependencies**
Anaconda2\include	OpenCV 3.1\opencv\build\x64\vc12\lib	aruco201.lib
AstraSDK-0.5.0-20160426Z103143Z-vs2013-win64\include	AstraSDK-0.5.0-20160426T103143Z-vc2013-win64\lib	opencv_world310d.lib
OpenCV 3.1\opencv\build\include	Anaconda2\libs	astra.lib
OpenCV 3.1\opencv\build\include\opencv	RVL\x64\Release	astra_core.lib
aruco-2.0.12\aruco-2.0.12\src	aruco-2.0.12\build\bin\Release	astra_core_api.lib
RVL\RVLCore\Include		python27.lib
RVL\RVLObjectDetection\Include		RVLCore.lib
RVL\RVLPCLTools\Include		RVLPCLTools.lib
RVL\RVLRecognition\Include		RVLPCSegment.lib
RVL\RVLPCSegment\Include		RVLRecognition.lib
RVL\RapidXML\include		RVLObjectDetection.lib

Na početku programa korisnika se pita koji će objekt biti postavljen na sceni, odnosno treba unijeti 1 ako će biti postavljen tetrapak, ili 2 ako će biti postavljena šalica. Taj se broj sprema u varijablu *k* koja se kasnije koristi pri izračunu točke hvatanja (Pr.k.3.1.).

[1] Robot Vision Library (RVL) je programska biblioteka razvijena na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek

```
int k;
printf("Unesite 1 za tetrapak ili 2 za salicu: ");
scanf("%d", &k);
```

Programski kod 3.1. Odabir objekta koji će se nalaziti na sceni

Na početku programa stvaraju se memorijski spremnici *mem0* i *mem*. Memorijski spremink *mem0* služi za pohranu podataka koji ostaju nepromijenjeni tokom cijelog izvođenja programa, dok *mem* služi za pohranu podataka koji predstavljaju rezultat obrade jedne slike i brišu se na početku obrade sljedeće slike. Nakon toga, poziva se funkcija *Init* klase *RMM* koja sadrži funkcije za robotsku manipulaciju**.** Ta funkcija pokreće inicijalizacijske funkcije kamere i robota. Nakon toga, uzima se slika scene funkcijom *astra_temp_update***,** koja trenutnu sliku snimljenu kamerom pohranjuje u za to određeno polje u memoriji računala. Prepoznavanju objekata prethodi predobradba podataka koji će se koristiti dalje u programu i u tu se svrhu koriste funkcije klasa *PCLMeshBuilder*, *ObjectDetector* te *SurfelGraph*. Klasa *PCLMeshBuilder* služi za učitavanje 3D modela iz datoteke i pretvaranje istih u format prikladan za korištenje od strane programa za prepoznavanje. Naredbe *CreateParamList* i *LoadParams* koriste klase *RVL*-a za učitavanje parametara iz konfiguracijske datoteke. Ime konfiguracijske datoteke zadano je varijablom *cfgFileName*. Klasa *ObjectDetector* prvenstveno služi za detekciju pojedinačnih objekata na složenoj sceni, a u ovom radu je korištena samo za spremanje scene u *.ply* formatu. Varijablom *objectCfgFileName* specificira se konfiguracijska datoteka klase *ObjectDetector*. Klasa *PlanarSurfelGraph* služi za detekciju ravninskih segmenata, koji se pohranjuju u objekt klase *SurfelGraph* (Pr.k.3.2.).

```
// Inicijalizacija instance klase PCLMeshBuilder
RVL::PCLMeshBuilder meshBuilder;
meshBuilder.CreateParamList(&mem0);
meshBuilder.ParamList.LoadParams(cfgFileName);

// Sirina (w) i visina(h) slike
int w = 640;
int h = 480;

pcl::PointCloud<pcl::PointXYZRGBA>::Ptr PC(new
pcl ::PointCloud<pcl::PointXYZRGBA>(w, h));

meshBuilder.PC = PC;

// Inicijalizacija instanci klasa za detekciju ravninskih segmenata
RVL::SurfelGraph surfels;
surfels.pMem = &mem;
surfels.CreateParamList(&mem0);
surfels.ParamList.LoadParams(cfgFileName);
RVL::PlanarSurfelDetector surfelDetector;
surfelDetector.CreateParamList(&mem0);
```

```
surfelDetector.ParamList.LoadParams(cfgFileName);

// Inicijalizacija instance klase objectDetector
RVL::ObjectDetector objectDetector;
objectDetector.pMem0 = &mem0;
objectDetector.pMem = &mem;
objectDetector.cfgFileName = RVLCreateString(objectCfgFileName);
objectDetector.Init();
objectDetector.vpMeshBuilder = &meshBuilder;
objectDetector.LoadMesh = RVL::LoadMesh;
objectDetector.CreateMesh = RVL::CreateMesh;

// Akvizicija slike s RGB-D kamere. Dubinska slika se sprema u polje depthImage.
RVL::Array2D<short int> depthImage;
cv::Mat pMC_(3, 1, CV_32F);
astra_temp_update();
cv::flip(rmm.Image, rmm.ImageF, 1);

memcpy(rmm.depth.data, imgD, w * h * sizeof(int16_t));
cv::flip(rmm.depth, rmm.depthF, 1);

depthImage.Element = (short *)(rmm.depthF.data);
depthImage.w = w;
depthImage.h = h;

// Brisanje memorijskog spremnika mem na pocetku obrade slike
mem.Clear();

// Pretvaranje dubinske slike u mrezu trokuta i spremanje u datoteku scene.ply
objectDetector.DetectObjects(NULL, &depthImage, NULL);
```

Programski kod 3.2. Funkcije korištene za predobradbu podataka

Prepoznavanje objekata obavlja se koristeći funkcije iz klase *PSGM*. Funkcija *Learn* te klase kao ulaz prima ime datoteke u kojoj se nalazi popis modela za prepoznavanje u *.ply* obliku. Ona kao rezultat generira deskriptore i provodi se *off-line*. Iduća funkcija, *LoadModelDataBase*, učitava te deskriptore. Funkcija *LoadModelMeshDB* učitava bazu 3D modela objekata od interesa. Funkcija *Interpret* kao ulaz prima *mesh*, odnosno 3D model scene, koji učitava pomoću funkcije *LoadMesh*. Funkcija *Interpret* kao rezultat generira hipoteze, odnosno vraća rotacijsku matricu **RICP** i translacijski vektor **tICP** koji opisuju položaj objekta u k.s. kamere, kao i ID objekta koji označava indeks prepoznatog objekta na sceni. Nakon što je prepoznat objekt na sceni, funkcija *ICP_refined*, koristeći *Iterative closest point* (ICP) algoritam, poboljšava estimaciju položaja objekta na sceni. Funkcija *HypothesisEvaluation* dodjeljuje procjenu vjerojatnosti svakoj hipotezi, dok funkcija *GetHypothesesCollisionConsensus* uklanja one hipoteze koje su proturječne nekoj vjerojatnijoj hipotezi. U vektor **consensusHypotheses** spremaju se indeksi hipoteza poredani od onog najvjerojatnijeg. Pomoću funkcije *GetMatch* dohvaća se prvi element, odnosno hipoteza koja

je dobila najveću ocjenu, te se sprema u *RVL::RECOG::PSGM_::MatchInstance *pHypothesis* (Pr.k.3.3.).

```
	// Inicijalizacija instance klase PSGM
	RVL::PSGM recognition;
	recognition.CreateParamList(&mem0);
	recognition.ParamList.LoadParams(cfgFileName);
	recognition.pMem = &mem;
	recognition.pMem0 = &mem0;
	recognition.vpMeshBuilder = &meshBuilder;
	recognition.LoadMesh = RVL::LoadMesh;
	recognition.pSurfels = &surfels;
	recognition.pSurfelDetector = &surfelDetector;
	recognition.MTGSet.pMem = recognition.pMem0;
	recognition.Init(cfgFileName);

	// Ucitavanje baze deskriptora
	recognition.LoadModelDataBase();

	// Ucitavanje 3D modela objekata od interesa iz datoteke modelsFileName
	recognition.LoadModelMeshDB(modelsFileName, &recognition.vtkModelDB, false,
0.4);

	// Ucitavanje scene iz datoteke scene.ply
	RVL::Mesh mesh;
	LoadMesh(&meshBuilder, "scene.ply", &mesh, false);

	// Brisanje memorijskog spremnika mem na pocetku obrade slike
	mem.Clear();

	// Generiranje hipoteza o objektima na sceni
	recognition.Interpret(&mesh);

	// Poboljsanje tocnosti polozaja objekta za najvjerojatnije hipoteze
	GenerateSegmentNeighbourhood(&recognition, 0.1);
	recognition.ICP_refined(RVL::PCLICP, RVL::PCLICPVariants::Point_to_plane,
recognition.bestSceneSegmentMatches2);

	// Konacno izracunavanje vjerojatnosti hipoteza
	recognition.HypothesisEvaluation(recognition.bestSceneSegmentMatches2, true);

	// Hipoteze koje se preklapaju s vjerojatnijom hipotezom se odbacuju
	recognition.noCollisionHypotheses.clear();
	recognition.transparentHypotheses.clear();
	recognition.envelopmentColisionHypotheses.clear();
	recognition.GetHypothesesCollisionConsensus(&recognition.noCollisionHypotheses,
&recognition.bestSceneSegmentMatches2, 10);

	// Pokazivac na najvjerojatniju hipotezu se sprema u pHypothesis.
	RVL::RECOG::PSGM_::MatchInstance                *pHypothesis                =
recognition.GetMatch(recognition.consensusHypotheses[0]);
```

Programski kod 3.3. Funkcije korištene za prepoznavanje objekata na sceni

Za određivanje točke hvatanja koristi se funkcija *DefineGrasp* klase *RMM*. Ta funkcija prima sljedeće parametre: prvi je broj koji određuje koji će objekt biti postavljen na sceni(1 ako je tetrapak, a 2 ako je šalica), zatim rotacijsku matricu **RICP** te translacijski vektor **tICP**

ishodišta modela u k.s. kamere. Ako je postavljeni objekt na sceni tetrapak, funkcija određuje je li on postavljen u uspravnom ili u polegnutom položaju kao i potrebnu širinu otvora hvataljke. Ako je tetrapak postavljen u uspravnom položaju, određuje se pomak po z-osi budući da hvataljka ne doseže sredinu objekta. Ako je postavljeni objekt na sceni šalica, određuju se pomaci po z-osi i po x-osi, jer se u slučaju šalice hvata rub, kao i širina otvora hvataljke. Točka hvatanja se, dakle, računa na sljedeći način:

$$\mathbf{pGC} = \mathbf{pRFMC} + \boldsymbol{xicp}*pomakx + \mathbf{yicp}*pomaky + \mathbf{zicp}*pomakz \qquad (3\text{-}1)$$

gdje je:

- **pGC** – točka hvatanja,
- **pRFMC** – ishodište modela u k.s. kamere,
- **xicp** – x-os modela,
- *pomakx* – pomak po x-osi modela,
- **yicp** – y-os modela,
- *pomaky* – pomak po y-osi modela,
- **zicp** – z-os modela,
- *pomakz* – pomak po z-osi modela.

Točka **pRFMC** te osi **xicp**, **yicp** i **zicp** su vrijednosti dobivene iz translacijskog vektora koji sadrži ishodište modela objekta u k.s. kamere, i rotacijske matrice najbolje hipoteze (Pr.k.3.4.).

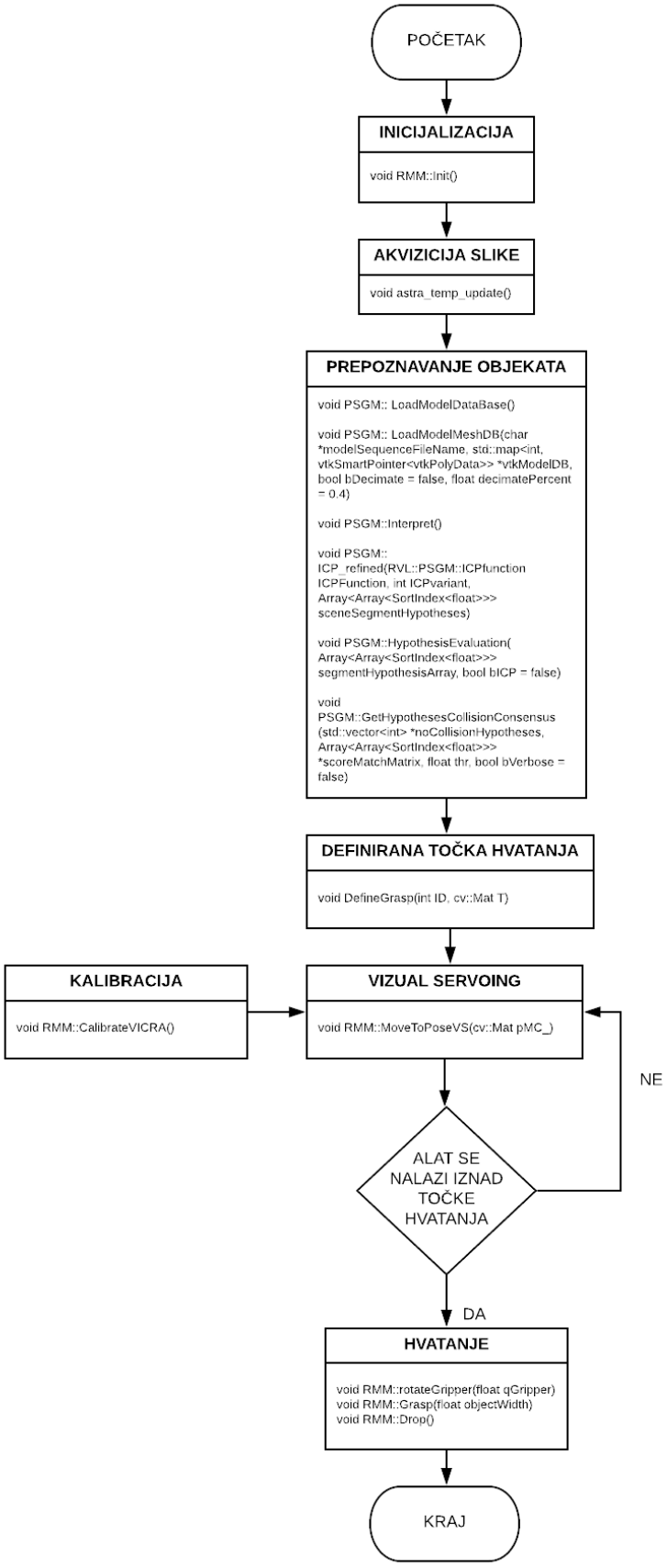
```
xicp.at<float>(0) = pHypothesis->RICP[0];
xicp.at<float>(1) = pHypothesis->RICP[3];
xicp.at<float>(2) = pHypothesis->RICP[6];

yicp.at<float>(0) = pHypothesis->RICP[1];
yicp.at<float>(1) = pHypothesis->RICP[4];
yicp.at<float>(2) = pHypothesis->RICP[7];

zicp.at<float>(0) = pHypothesis->RICP[2];
zicp.at<float>(1) = pHypothesis->RICP[5];
zicp.at<float>(2) = pHypothesis->RICP[8];

pRFMC.at<float>(0) = pHypothesis->tICP[0];
pRFMC.at<float>(1) = pHypothesis->tICP[1];
pRFMC.at<float>(2) = pHypothesis->tICP[2];
```

Programski kod 3.4. Određivanje osi i ishodišta modela objekta

Ako se dogodi da objekt bude postavljen u obrnutom smjeru, odnosno da z-os bude usmjerena u površinu na kojoj je objekt postavljen, točka hvatanja računa se na sljedeći način:

$$\mathbf{pGC} = \mathbf{pRFMC} + \mathbf{xicp}*pomakx + \mathbf{yicp}*pomaky - \mathbf{zicp}*pomakz \quad (3\text{-}2)$$

Funkcija *CalibrateVICRA* klase *RMM,* kao rezultat daje transformacijsku matricu k.s. robota u odnosu na k.s. kamere i to tako da pomiče prvi rotacijski zglob za predefinirane kuteve *qinit* i *q1***.** Nakon toga, pokreće se funkcija *MoveToPoseVS* klase *RMM* koja pomiče robotsku ruku na ciljnu poziciju u k.s. kamere. Kada se hvataljka pozicionira iznad točke hvatanja, poziva se funkcija *rotateGripper* klase *RMM* koja rotira hvataljku na zadanu orijentaciju. Funkcija *Grasp* zatvara hvataljku na zadanu širinu, a kao ulaz prima širinu otvora hvataljke što se odredi u funkciji *DefineGrasp*. Nakon što se objekt uhvati i pozicionira iznad željenog odredišta, poziva se funkcija *Drop* koja otvara hvataljku i ispušta objekt. Na slici 3.1. prikazan je dijagram toka programa.



Slika 3.1. Dijagram toka programa

4. EKSPERIMENTALNA EVALUACIJA

U ovom će se poglavlju dati pregled eksperimentalne analize korištenih metoda za vizualni servoing i planiranje hvatanja objekata.

4.1. Eksperimentalni postav

Eksperimentalna evaluacija izvršena je koristeći robotski manipulator (Sl.4.1.) koji je izrađen na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek u *SCARA* (*Selective Compliance Assembly Robot Arm*) konfiguraciji, a nazvan je *VICRA (Vision Controlled Robot Arm).* Robot ima jedan translacijski i tri rotacijska zgloba. Prva tri zgloba pokretana su koračnim motorima i ona određuju poziciju alata, dok je četvrti zglob pokretan DC servomotorom i on određuje orijentaciju alata. Robot je upravljan Arduino mikrokontrolerom koji s računalom komunicira preko USB-a [10].

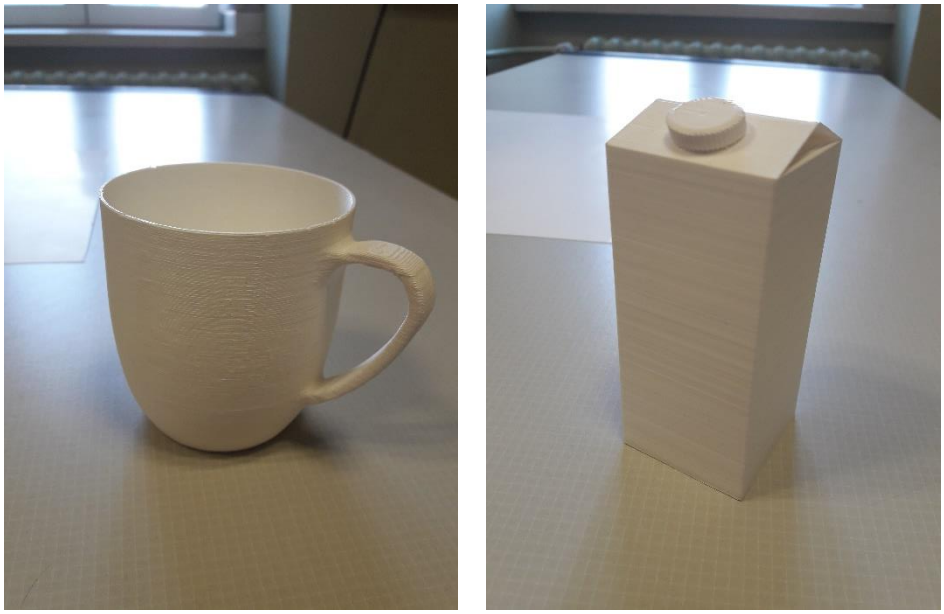
Slika 4.1. Robotski manipulator u SCARA konfiguraciji

Na vrhu robota montirana je RGB-D kamera koja nadgleda radni prostor robota i optimizirana je za korištenje pri kraćim dometima, između 0.4 m i 2 m. Kamera omogućuje 3D viziju te je korištena kako bi se dobile vizualne informacije potrebne za navođenje robotske ruke od početne do ciljne pozicije [8].

Iznad alata postavljen je marker kojeg detektira ArUco biblioteka i na taj način određuje položaj alata, odnosno hvataljke u k.s. kamere [12].

4.2. Pokusi hvatanja

S ciljem evaluacije korištenih metoda kalibracije, vizualnog servoinga te opisanog planiranja hvatanja, provedeni su pokusi hvatanja dvaju objekata; šalice (Sl.4.2.a) i tetrapaka (Sl.4.2.b).

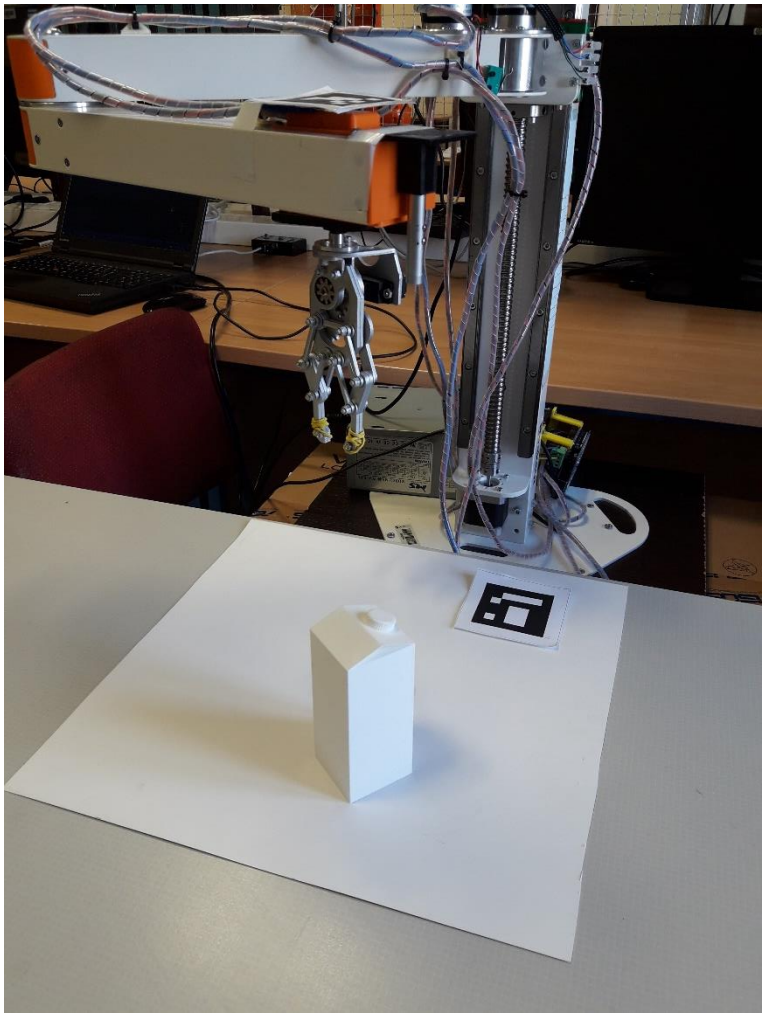


a) b)

Slika 4.2. Objekti za hvatanje; a)šalica, b)tetrapak

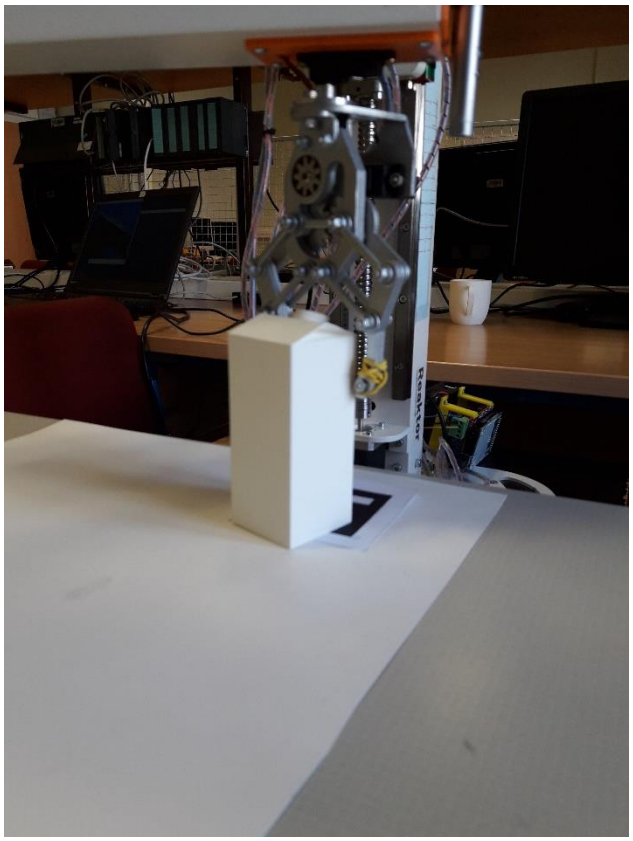
Provedeno je ukupno 30 pokusa hvatanja i to 10 pokusa za šalicu te po 10 pokusa za tetrapak u uspravnom i u polegnutom položaju. Kod svake operacije hvatanja objekt je bio položen na

horizontalnu ravninu u uspravnom ili polegnutom položaju negdje u radnoj okolini robota. Objekt je zatim bio prepoznat te je određena točka hvatanja kao i kut rotacije alata kako je opisano u potpoglavlju 2.3.. Vizualni servoing doveo je hvataljku robota iznad željene točke te je izvršeno hvatanje. Objekt je zatim premješten na ciljno odredište u radnoj okolini robota koja je predstavljena markerom na horizontalnoj ravnini (Sl.4.3.).



Slika 4.3. Početni položaj robota i objekta

Pokus je smatran uspješnim ako je objekt ispravno detektiran, ako je hvataljka ispravno pozicionirana i rotirana te ako je objekt uspješno uhvaćen i prenesen na ciljno odredište (Sl. 4.4.).

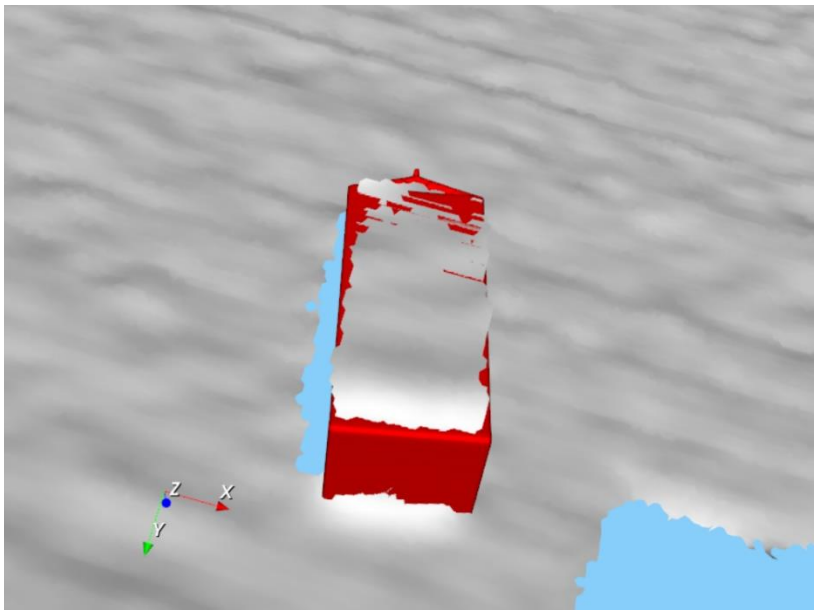


Slika 4.4. Uspješno hvatanje

U tablici 4.1. prikazani su rezultati pokusa. Od ukupno 30 pokusa, 4 su bila neuspješna zbog pogrešno izračunate točke centra mase objekta. Ta pogreška dovodi do pogrešnog pozicioniranja alata iznad objekta kao i do pogrešnog kuta rotacije. Do pogreške određivanja centra mase objekta došlo je zbog netočnog postavljanja modela na sceni, odnosno model se ne podudara u potpunosti sa snimljenim objektom (Sl. 4.5.). Ostali pokusi bili su uspješni.

Tablica 4.1. Rezultati pokusa hvatanja objekata

	Uspješno	Neuspješno
Šalica	10	0
Tetrapak (uspravno)	10	0
Tetrapak (polegnuto)	6	4
Postotak	86.67	13.33



Slika 4.5. Prepoznati model (crveno) blago zarotiran u odnosu na stvarni objekt (sivo)

5. ZAKLJUČAK

U ovom je radu opisan sustav upravljanja robotskom rukom koji koristi vizualnu informaciju za pozicioniranje alata i hvatanje objekata. Na početku se provodi kalibracija kamere i robota kako bi se odredio položaj robota u odnosu na kameru jer se sve pozicije na koje želimo dovesti robot zadaju u koordinatnom sustavu kamere. Razlog tome je korištenje relativnog umjesto apsolutnog pozicioniranja budući da nam je za hvatanja objekta bitno da se alat pozicionira iznad njega, a nije nam bitana njegova aposlutna pozicija u odnosu na robota. Odgovarajući program za prepoznavanje objekata prepoznaje objekt od interesa na sceni te određuje njegov položaj u odnosu na kameru. Nakon toga, potrebno je odrediti točku hvatanja koja ovisi o obliku i veličini predmeta, njegovom položaju na sceni te o vrsti hvataljke koja se koristi. Na kraju je još potrebno izračunati kut rotacije kako bi objekt mogao stati u hvataljku te kako bi ga se moglo prebaciti na željeno odredište.

Pokusi hvatanja vršili su se nad dvama objektima, šalici i tetrapaku. Hvatanje je bilo uspješno u 86.67% slučajeva, i to 100% uspješno pri hvatanju šalice, 100% uspješno pri hvatanju tetrapaka u vertikalnom položaju te 60% uspješno pri hvatanju tetrapaka u horizontalnom položaju. Jedan od razloga ovih neuspješnih hvatanja leži u netočnom podudaranju modela i objekta na snimljenoj sceni. Razlog tomu je što je 3D kamera osjetiljiva na osvjetljenje koje pada na objekt te ga zbog toga može percipirati drugačijim nego što on jest. Rješenje ovog problema može se postići korištenjem boljeg osvjetljenja, stvaranjem što manje sjena ili korištenjem kvalitetnije kamere. Drugi razlog neuspješnog hvatanja leži u ograničenjima hvataljke. Da se hvataljka mogla raširiti više, problem pozicioniranje modela na sceni ne bi bio toliko izražen.

Može se zaljučiti kako je predložena metoda bila uspješna za korištene modele, te se može nadograditi na veću bazu modela i omogućiti hvatanje više objekata za redom.

LITERATURA

[1] International Federation of Robotics, dotupno na: https://ifr.org/ [Lipanj 2018.]

[2] P. Đurović, R. Grbić, R. Cupec, D. Filko, Low Cost Robot Arm with Visual Guided Positioning, 40th Jubilee International Convention on Information and Communication Technology, Electronics and Microelectronics, Opatija, Hrvatska, 22.-26. 05. 2017

[3] C. Papazov and D. Burschka, An Efficient RANSAC for 3D Object Recognition in Noisy and Occluded Scenes, In Proceedings of the 10th Asian Conference on Computer Vision (ACCV'10), November 2010, dostupno na: http://www.i6.in.tum.de/Main/Publications/Papazov2010.pdf [Kolovoz 2018.]

[4] Fischler MA & Bolles RC, *Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography*, Graphics and Image Processing, vol. 24, no. 6, pp. 381–395, 1981

[5] A. Aldoma, F. Tombari, L. Di Stefano, and M. Vincze, "A global hypothesis verification method for 3d object recognition, in European Conference on Computer Vision (ECCV), 2012., dostupno na: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.700.7729&rep=rep1&type=pdf [Kolovoz 2018.]

[6] Tombari F, Salti S, Di Stefano L, *Unique signatures of Histograms for local surface description*, in Proc. 11th European Conference on Computer Vision (ECCV), 2010.

[7] J. Glover and S. Popovic, Bingham Procrustean Alignment for Object Detection in Clutter, IEEE/RSJ Int. Conf. Intell. Robots & Systems (IROS), Tokyo, 2013, dostupno na: http://lis.csail.mit.edu/pubs/glover-iros13.pdf [Kolovoz 2018.]

[8] Astra - Orbbec, dostupno na: https://orbbec3d.com/product-astra/ [Lipanj 2018.]

[9] I. Vidović, P. Đurović, D. Filko, R. Cupec, Convex Template Alignment for Efficient Hypothesis Evaluation, Tehničko izvješće ARP3D.TR12, Sveučilište J. J. Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek, 2017.

[10] P. Đurović, R. Grbić, R. Cupec, Visual servoing for low-cost SCARA robots using an RGB-D camera as the only sensor, Automatika, 8:4, 495-505, DOI: 10.1080/00051144.2018.1461771, June 2018.

[11] F. Chaumette, S. Hutchinson, Visual Servo Control Part I: Basic Approaches, IEEE Robotics & Automation Magazine, 1070-9932/06, 82-90, December 2006.

[12] ArUco: a minimal library for Augmented Reality applications based on OpenCV, dostupno na: https://www.uco.es/investiga/grupos/ava/node/26 [Lipanj 2018.]

[13] Eigen: a C++ template library for linear algebra: matrices, vectors, numerical solvers, and related algorithms, dostupno na: http://eigen.tuxfamily.org [Rujan 2018.]

SAŽETAK

Opisan je sustav koji omogućuje pozicioniranje alata robotske ruke i hvatanje objekata pomoću informacija dobivenih RGB-D kamerom. Korištena je metoda kalibracije kamere i robotske ruke koja se provodi u dvije faze; prva je određivanje dominantne ravnine, a druga uključuje pokret prvog rotacijskog zgloba za poznati kut. Ovi parametri dovoljni su za vizualni servoing pomoću kojeg se alat pozicionira na željenu poziciju, odnosno točku hvatanja. Odgovarajućim programom za prepoznavanje objekata određuje se prisustvo objekta od interesa na sceni te njegov položaj u odnosu na kameru. Točka hvatanja definirana je u odnosu na referentni koordinatni sustav modela objekta od strane korisnika. Nakon određivanja točke hvatanja, računa se kut rotacije alata kako bi se objekt mogao uhvatiti i premjestiti na željeno odredište. Funkcionalnost razvijenog sustava ispitana je pokusima na stvarnoj robotskoj ruci.

Ključne riječi: visual servoing, kalibracija robotske ruke i kamere, planiranje hvatanja, prepoznavanje objekata, RGB-D kamera

ABSTRACT

A system for vision-based tool positioning and grasp planning is described. The camera calibration method is carried out in two phases: the first phase includes detection of the dominant plane, and the second involves the movement of the first rotational joint for a known angle difference. These parameters are sufficient for visual servoing. The target position of the tool is determined by detecting object of interest in an image using an object recognition program. The grasp point is defined by the user in relation to the reference frame of the object model. After determining the grasping point, the rotation angle of the tool is calculated so that the object can fit in and can be moved to a desired destination. The developed system is experimentally tested using a real robot arm.

Keywords: visual servoing, hand-eye calibration, grasp planning, object recognition, RGB-D camera

ŽIVOTOPIS

Maja Mihaljević rođena je 18. travnja 1993. godine u Đakovu, Hrvatska. Osnovnu školu Mate Lovraka u Vladislavcima pohađa od 2000. do 2008. godine. Srednjoškolsko obrazovanje započinje 2008. godine upisom u Isusovačku klasičnu gimnaziju s pravom javnosti u Osijeku. Srednju školu završava 2012. godine kada polaže državnu maturu i ostvaruje izravan upis na Preddiplomski sveučilišni studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. Na drugoj godini sudjeluje na međunarodnom sveučilišnom natjecanju studenata elektrotehničkih fakulteta iz predmeta Matematika 1. Godine 2015. završava Preddiplomski studij te na istom fakultetu upisuje Diplomski sveučilišni studij računarstva, izborni blok Procesno računarstvo.

Maja Mihaljević