Detekcija i praćenje ljudi u radnom okruženju robota pomoću RGB-D kamere i YOLO algoritma

Svirac, Damian

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

Sveučilišni studij

DETEKCIJA I PRAĆENJE LJUDI U RADNOM OKRUŽENJU ROBOTA POMOĆU RGB-D KAMERE I YOLO ALGORITMA

Diplomski rad

Damian Svirac

Osijek, 2021.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit

Osijek, 12.07.2021.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za diplomski ispit

Ime i prezime studenta:	Damian Svirac
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D-1086R, 06.10.2019.
OIB studenta:	14210449326
Mentor:	Prof.dr.sc. Robert Cupec
Sumentor:	Dr. sc. Petra Đurović
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Izv. prof. dr. sc. Emmanuel-Karlo Nyarko
Član Povjerenstva 1:	Prof.dr.sc. Robert Cupec
Član Povjerenstva 2:	Izv. prof. dr. sc. Damir Filko
Naslov diplomskog rada:	Detekcija i praćenje ljudi u radnom okruženju robota pomoću RGB-D kamere i YOLO algoritma
Znanstvena grana rada:	**Automatizacija i robotika (zn. polje elektrotehnika)**
Zadatak diplomskog rada:	RGB-D kamerom postavljenom u radni prostor robota potrebno je prepoznati čovjeka i odrediti brzinu i orijentaciju njegova kretanja primjenom YOLO algoritma. Na temelju brzine i orijentacije kretanja čovjeka, potrebno je mijenjati način rada robota u stvarnom vremenu. Sustav treba implementirati u ROS-u. Tema rezervirana za: Damian Svirac Sumentor s FERIT-a: 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: 3 razina
Datum prijedloga ocjene mentora:	12.07.2021.

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, 20.07.2021.

Ime i prezime studenta:	Damian Svirac
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D-1086R, 06.10.2019.
Turnitin podudaranje [%]:	5

Ovom izjavom izjavljujem da je rad pod nazivom**: Detekcija i praćenje ljudi u radnom okruženju robota pomoću RGB-D kamere i YOLO algoritma**

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

i sumentora Dr. sc. 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:

Zahvalu upućujem svojem mentoru prof.dr.sc. Robertu Cupecu i sumentorici dr.sc. Petri Đurović koji su svojim znanstvenim i stručnim savjetima pružili pomoć u pisanju diplomskog rada. Posebno hvala sumentorici koja je pratila i usmjeravala izradu cijeloga rada u pravome smjeru, a pritom imala strpljenja i razumijevanja.

Zahvalu upućujem i stručnom sumentoru iz Danieli Systeca Ferdi Bošnjaku koji je potaknuo izradu diplomskog rada i omogućio suradnju s tvrtkom Danieli Systec koja je financirala projekt iz kojega je nastao ovaj rad.

Zahvaljujem se kolegama Dariju i Valentinu koji su zajedno sa mnom sudjelovali na projektu, motivirali me i dijelili lijepe i manje lijepe trenutke studiranja.

Ogromno hvala mojoj zaručnici Valentini koja je bila velika podrška i motivacija u mome studiranju i jedan od razloga zbog kojih sam uspješno završio svaku akademsku godinu.

Na kraju, najveću zahvalnost iskazujem svojoj obitelji i roditeljima koji su mi pružili mogućnost studiranja. Od srca im hvala na neizmjernoj podršci i riječima ohrabrenja u teškim trenucima. Hvala im na strpljivosti i razumijevanju.

SADRŽAJ

1. UVOD

Robotika i umjetna inteligencija rastuće su grane računarstva. Razvojem novih tehnologija rastu i mogućnosti primjene robotskih sustava u raznim industrijama poput automobilske i kemijske industrije. Isto tako, povećanjem integracije robotskih sustava, ljudi su sve više u međusobnoj interakciji s robotskim sustavima, bilo da samo prolaze pokraj robota, ili su u direktnom kontaktu s istim. Robotski sustav trebao bi biti siguran za sebe i svoje okruženje. Dok su robot i njegovo okruženje materijalne i zamjenjive stvari, ljudski život je neprocjenjiv i nezamjenjiv. Stoga je najveći naglasak na sigurnost čovjeka.

Cilj ovog diplomskog rada je razvoj sustava u ROS-u (*Robot Operating System*) koji pomoću kamere prepoznaje čovjeka u okruženju robotskog manipulatora, te prilagođava rad robotskog manipulatora s obzirom na brzinu i smjer kretanja čovjeka. Sustav se temelji na YOLO (*You Only Look Once*) algoritmu za prepoznavanje ljudi i SORT Deep (*Simple Online and Realtime Tracking with Deep Association Metric*) algoritmu za praćenje ljudi. Prvi korak je istraživanje postojećih rješenja koja se koriste za slične probleme. Zatim slijedi implementacija prikladnih metoda za prepoznavanje i praćenje ljudi. Treći korak je integracija razvijenog algoritma s industrijskim robotskim manipulatorom ABB IRB-2400L i RGB-D kamerom. Zadnji korak čini eksperimentalna evaluacija sustava na skupu podataka koji je kreiran direktno za potrebe ovoga rada.

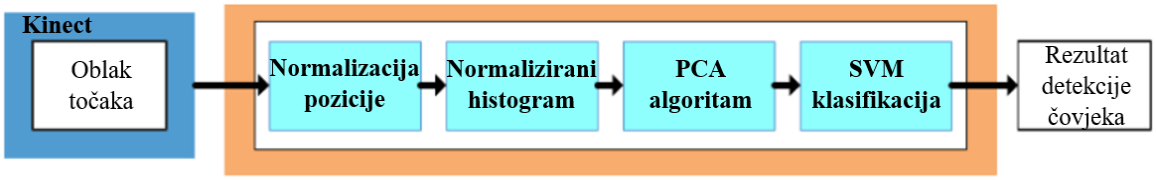
Diplomski rad nastao je kao dio projekta *Humans Detected by Robots* (HDR) [1], koji je pokrenut u suradnji FERIT[1]-a i tvrtke Danieli Systec koja je u partnerstvu s Danieli Automation S.p.A. Robotski sustavi Danieli Automation-a povremeno zahtijevaju ljudsku intervenciju kako bi ispravno radili. Stoga postoje određeni zahtjevi ove tvrtke na funkcionalnost i sigurnost robotskog sustava prema kojima je izrađen ovaj rad.

[1] Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek

2. PREGLED RADOVA U PODRUČJU

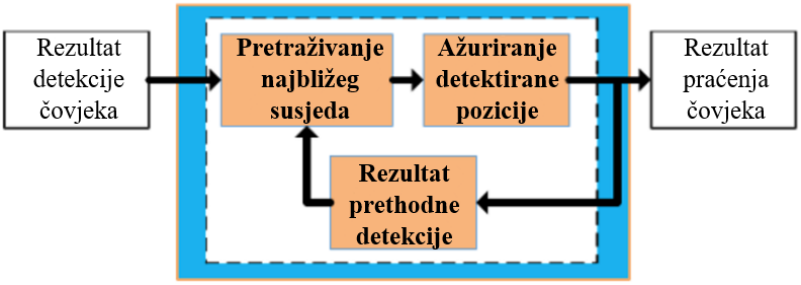
Autori rada [2] predlažu sigurnosne indikatore temeljene na energiji za robotske sustave koji dijele svoj radni prostor s ljudima. Na temelju tih indikatora, postavljaju se određena sigurnosna ograničenja u algoritmu upravljanja robotskim sustavom. Prvo ograničenje postavlja se na kinetičku energiju robotskog sustava kako bi se ograničila energija koja se gubi prilikom kolizije. Ovo ograničenje ovisi o udaljenosti između robota i čovjeka. Udaljenost se računa algoritmom koji koristi oblak točaka dobiven dubinskim senzorima (Kinect RGB-D kamerama). Drugo ograničenje je količina potencijalne energije koja je dozvoljena u interakciji čovjeka i robota tijekom fizičkog kontakta. Koristi se za moduliranje sile kontakta. Krajnji cilj je osigurati sigurnost za čovjeka koji ulazi u radno područje robota i ostvaruje fizički kontakt s robotom.

Rad [3] bavi se problemima navigacije mobilnog robota u kompleksnoj okolini gdje je Kinect kamera postavljena na mobilnog robota. Autori rada prezentiraju daljinsko upravljani sustav za detekciju, praćenje i sigurnost ljudi u stvarnom vremenu. Daljinsko upravljani sustav napravljen je u ROS-u, a implementiran na mobilnog robota s četiri kotača. Blok dijagram procesa detekcije ljudi prikazuje Slika 2.1.



Slika 2.1. Blok dijagram procesa detekcije ljudi

Rezultati detekcije ljudi služe kao ulazna vrijednost za algoritam praćenja ljudi. Blok dijagram procesa praćenja ljudi prikazuje Slika 2.2.



Slika 2.2. Blok dijagram procesa praćenja ljudi

U radu [4] implementiran je sustav koji estimira 3D položaj ljudske glave. Sustav najprije detektira ljudsku glavu i lice koristeći detektor pokreta, Houghovu transformaciju i statistički model boje. Nakon toga, pomoću pomaka i promjene nagiba na slici kamere, lice čovjeka postavlja se u sredinu vidnog polja kamere. Koristeći informaciju iz fokusiranja prstena autofokusa kamere, mjeri se udaljenost između kamere i ljudskog lica. Imajući informaciju o apsolutnoj poziciji kamere te njenim kutovima nagiba i pomaka, može se izračunati apsolutna pozicija čovjeka u prostoriji. U slučaju malih pomaka čovjeka, sustav estimira udaljenosti čovjeka s pogreškom manjom od 10 cm. Raspon udaljenosti na kojem sustav efektivno radi je od 90 cm do 340 cm.

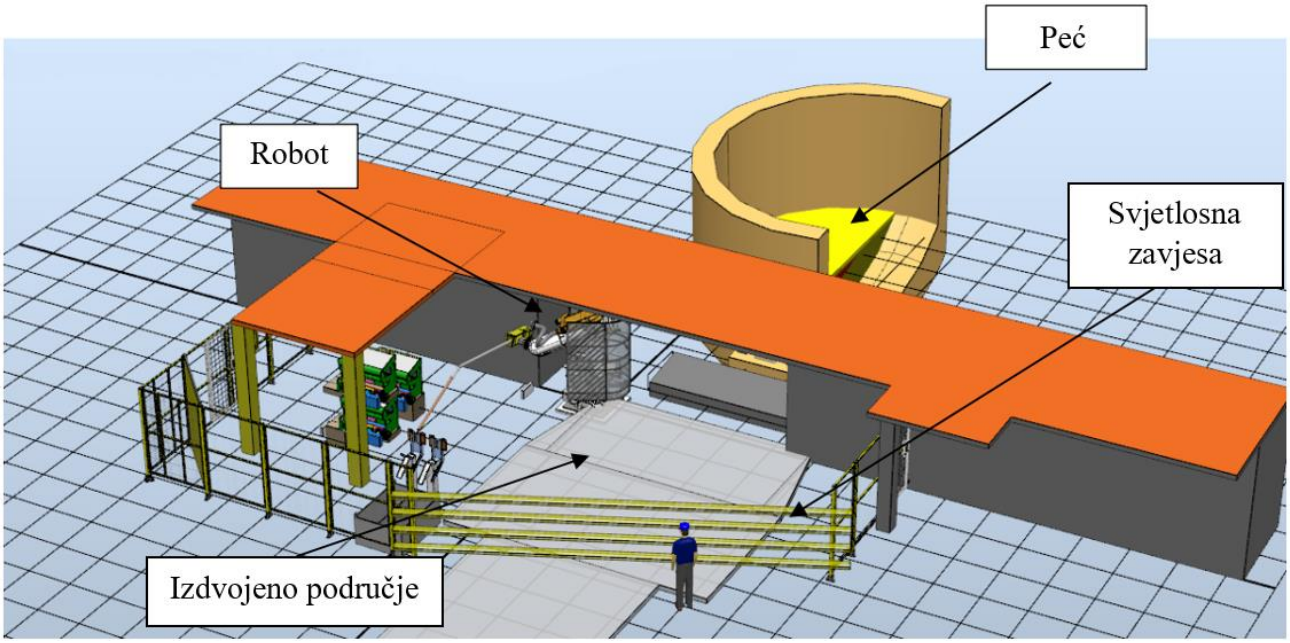
YOLO (*You Only Look Once*) algoritam predstavljen u radu [5] daje novi pristup u detekciji objekata. Dotadašnji radovi u području detekcije objekata prilagođavaju klasifikatore za izvođenje detekcije. YOLO s druge strane pristupa detekciji objekata kao regresijskom problemu za prostorno razdvajanje graničnih okvira (engl. *bounding boxes*), a svakom graničnom okviru pridružuje odgovarajuće vjerojatnosti pripadanja nekoj klasi. Za vršenje predikcije graničnih okvira i vjerojatnosti pripadanja klasi koristi se jedna neuronska mreža. S obzirom da je cijeli cjevovod detekcije (engl. *detection pipeline*) jedna neuronska mreža, algoritam se može optimizirati za poboljšanje performansi detekcije. YOLO algoritam radi dosta brzo pa je prikladan za korištenje u stvarnom vremenu. U usporedbi s drugim suvremenim (engl. *state-of-the-art*) metodama, YOLO algoritam radi više greški u lokalizaciji graničnih okvira, ali zato pravi puno manje lažno pozitivnih greški[2] (engl. *false positives*).

Rad [6] istražuje sistematične pristupe u praćenju više objekata na sceni. U radu se prezentira SORT (*Simple Online and Realtime* Tracking) algoritam kao prijedlog rješenja. Budući da algoritam praćenja koristi vanjski detektor objekata, kvaliteta samog detektora znatno utječe i na kvalitetu algoritma za praćenje objekata. Unatoč korištenju osnovnih tehnika za algoritam praćenja poput Kalmanovog filtera i Mađarskog algoritma, ovaj pristup postiže rezultate približne suvremenim metodama. Jednostavnost ovakvog pristupa omogućuje algoritmu vrlo brzo izvođenje, čak do dvadeset puta brže u odnosu na suvremene algoritme. Nadogradnja ovog pristupa predstavljena je u radu [7] algoritmom SORT Deep (*SORT with Deep Association Metric*) čime se unaprjeđuje sposobnost praćenja objekata na dužim periodima zaklonjenosti objekata. Time se efektivno smanjuje promjena identifikatora (engl. *identity switch*) istog objekta do 45%.

[2] Pogreška kada algoritam detektira objekt koji ne postoji na toj poziciji na slici

3. SIGURNOSNI SUSTAV

Sigurnosni sustav trebao bi biti siguran za samog sebe, svoje okruženje i čovjeka koji se kreće u blizini robota. S obzirom da je rad nastao u suradnji s Danieli Systecom, sigurnosni sustav razvijen je u skladu sa zahtjevima Danieli Automationa. Potreba Danieli Automation tvrtke za sigurnosne sustave neprestano raste, jer se sve više u njenim postrojenjima koriste robotski sustavi za razne zadaće. Primjer jednog sigurnosnog sustava u Danieli Automation postrojenju prikazuje Slika 3.1.



Slika 3.1. Primjer sigurnosnog sustava u Danieli Automation postrojenju

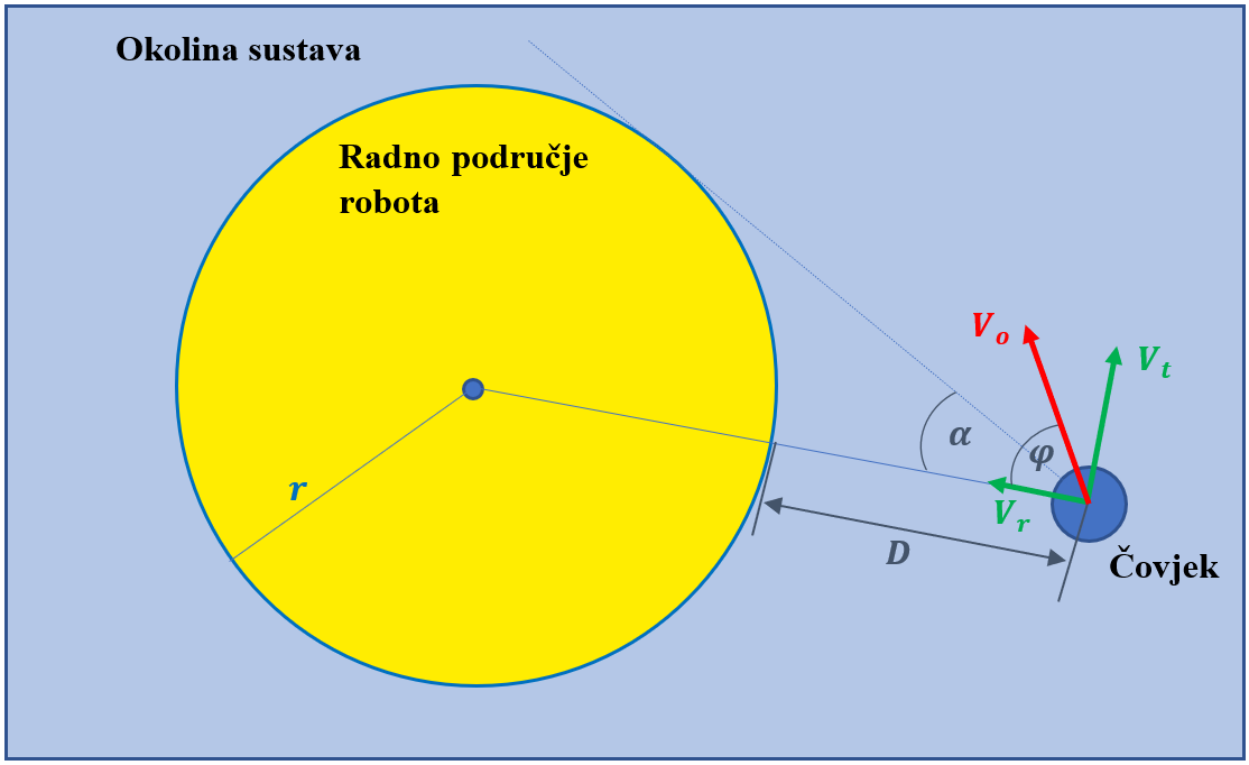
Sigurnosni sustav na slici sastoji se od robotskog sustava kojeg čine industrijski robotski manipulator i svjetlosna zavjesa. Zadaća robotskog manipulatora je izvođenje određenih mjerenja u peći, a svjetlosna zavjesa služi za otkrivanje ljudi koji ulaze u radno područje robota. Svjetlosna zavjesa je jednostavno i sigurno rješenje za zaštitu ljudi, ali kao takva ne može dati više informacija o objektu koji je ušao u izdvojeno područje. Samim time, svjetlosna zavjesa ne može raspoznati je li uopće čovjek taj koji je ušao u izdvojeno područje, niti dati informaciju o položaju čovjeka, njegovoj orijentaciji i brzini kretanja. Ovakav sustav nije efikasan, jer svaki put kada se dogodi bilo kakav ulaz u izdvojeno područje, jedina reakcija robota je zaustavljanje. Stoga se javlja potreba za naprednijim

sigurnosnim sustavima koji će omogućiti efikasniji rad robotskog sustava uz očuvanje sigurnosti za ljude.

Napredni sigurnosni sustavi temelje se na dva ključna faktora:

- promašena detekcija (engl. *missed detection*) – obavezna vrijednost je 0, jer svaka promašena detekcija dovoljna je da ugrozi sigurnost čovjeka
- lažno pozitivna detekcija – vrijednost bi trebala biti što manja jer se inače smanjuje efikasnost sustava.

Takvi sustavi koristili bi senzore (npr. RGB-D kamere) kojima se nadzire cijela okolina robotskog sustava i detektira prisutnost čovjeka. Prijedlog jednog takvog sustava predstavlja Slika 3.2.



Slika 3.2. Shematski prikaz radnog okruženja robotskog sustava

Radno područje robota modelirano je kao kružnica (u 3D prostoru je to valjak) čije je središte u bazi robota. Polumjer kružnice odgovara maksimalnom dosegu robota u slučaju kada nema ljudi u okolini

robota ili je brzina kretanja čovjeka dovoljno mala. Veće brzine kretanja čovjeka utječu na polumjer kružnice tako da se polumjer kružnice povećava prema sljedećoj jednadžbi:

$$r = \begin{cases} r_o, & ||V_o|| < 0.1\ [m/s] \\ 1.1r_o, & 0.1\left[\frac{m}{s}\right] \leq ||V_o|| < 1\ [m/s] \\ 1.5r_o, & ||V_o|| \geq 1\ [m/s] \end{cases} \tag{2.1}$$

gdje r_o predstavlja doseg robota, r predstavlja polumjer radnog područja (kružnice), a V_o vektor brzine čovjeka (operatora). Vektor brzine čovjeka može se izračunati kao $V_o = V_r + V_t$, gdje V_r predstavlja vektor radijalne brzine, a V_t vektor tangencijalne brzine čovjeka. Ovakva promjena polumjera kružnice osigurava sustavu više vremena za reakciju u slučaju brzog ili neočekivanog kretanja čovjeka.

Također, upravlja se brzinom alata robota sljedećim izrazom:

$$V_{rez} = V_{ee} \cdot V_n \tag{2.2}$$

gdje je V_{rez} rezultanta brzina alata robota, V_n nominalna brzina alata robota kada nema čovjeka u blizini te V_{ee} koeficijent brzine alata robota. Upravljanje brzinom alata robota određuje se koeficijentom brzine alata robota V_{ee} koji se mijenja ovisno o brzini i smjeru kretanja čovjeka. Jednadžba upravljanja koeficijentom brzine alata robota dana je sljedećom formulom:

$$V_{ee} = \begin{cases} 0\%, & \begin{cases} ||V_o|| \geq 2\left[\frac{m}{s}\right] \\ |\varphi| \leq \alpha \\ D \leq 0 \end{cases} \\ 50\%, & 0.1\left[\frac{m}{s}\right] \leq ||V_o|| < 2\left[\frac{m}{s}\right] \\ 75\%, & ||V_o|| < 0.1\left[\frac{m}{s}\right] \\ 100\%, & inače \end{cases} \tag{2.3}$$

gdje je $|\varphi|$ apsolutna vrijednost kuta između vektora radijalne brzine čovjeka V_r i vektora brzine čovjeka V_o, a α je kut između vektora radijalne brzine V_r i tangente na kružnicu u točki pozicije čovjeka. Ovime se dodatno osiguralo potrebno vrijeme sustavu za odgovarajuću reakciju.

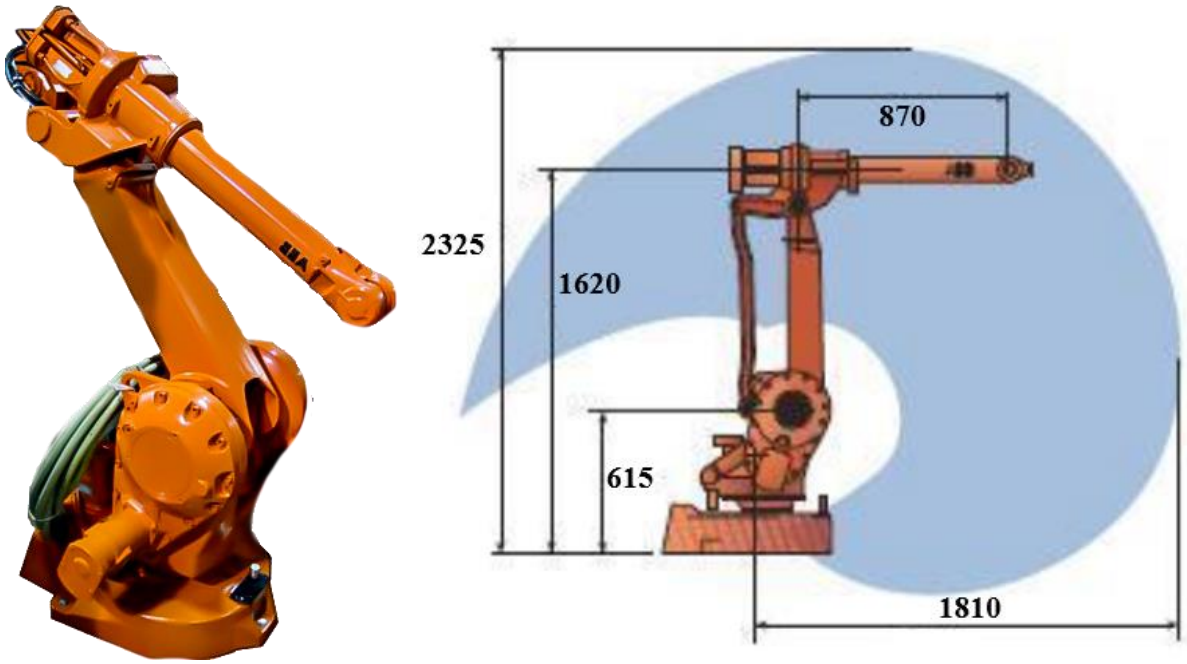
Općenito, sigurnosni sustav trebao bi se uključiti kada se čovjek nađe u okolini robotskog sustava. Sustav upravljanja trebao bi povećati polumjer radnog područja robota i smanjiti brzinu alata robota kako bi se dalo više vremena algoritmu upravljanja za reakciju, a samim time i smanjila mogućnost kolizije čovjeka s robotskim manipulatorom.

4. DIJELOVI ROBOTSKOG SUSTAVA

Razvijeni sigurnosni sustav u ovom radu implementiran je na stvarnom robotskom sustavu koji se sastoji od robotskog manipulatora i jedne RGB-D kamere. Sustav je izrađen u ROS-u čime se omogućuje komunikacija između svih elemenata robotskog sustava.

4.1. Robotski manipulator

Robotski manipulator korišten u ovome radu je industrijski robotski manipulator tvrtke ABB. ABB je svjetski poznati proizvođač industrijskih robotskih manipulatora s više od 300 000 proizvedenih robotskih manipulatora. Model korištenog manipulatora je IRB 2400L [8]. Prikaz manipulatora i njegovih dimenzija (u milimetrima) nalazi se na Slici 4.1.



Slika 4.1. Industrijski robotski manipulator IRB 2400L (lijevo); Dimenzije manipulatora IRB 2400 L (desno)

Prikazani robotski manipulator dostupan je na FERIT-u u zgradi na kampusu, a ustupila ga je na korištenje tvrtka Danieli Systec.

Specifikacije navedenog robotskog manipulatora su sljedeće:

- broj osi: 6
- nosivost: 7 [kg]
- dohvat: 1810 [mm]
- ponovljivost: 0.05 [mm]
- kontroler: IRC5

Navedeni manipulator svojim dosegom obuhvaća veliki radni prostor, ali zbog toga posjeduje manju nosivost. Ovakve postavke čine ga prikladnim za primjenu u industrijskim poslovima kao što su lučno zavarivanje (engl. *arc wielding*), sastavljanje (engl. *assembly*) i obrada materijala (engl. *material processing*).

4.2. RGB-D kamera

RGB-D kamere općenito su kamere koje uz RGB sliku[3] sadrže i dubinsku sliku. Pod dubinskom slikom podrazumijeva se slika na kojoj je svaki piksel, umjesto RGB vrijednostima boje, predstavljen vrijednošću udaljenosti (dubine) od kamere, iz čega se može dobiti 3D oblak točaka.

RGB-D kamera korištena u ovom radu je Microsoft Kinect v1 kamera [9] (Slika 4.2. lijevo). Tvrtka Microsoft počela je s proizvodnjom Kinect kamera u sklopu Xbox 360 igraćih konzola pri čemu se Kinect kamera koristila kao senzor koji prati pokrete igrača i imitira ih u video igri. Microsoft Kinect kamera koristi PrimeSense dubinski senzor koji projicira strukturirani infracrveni uzorak svjetlosti (Slika 4.2 desno) za estimaciju dubina pojedinog piksela.

Slika 4.2. Microsoft Kinect v1 kamera (lijevo); Strukturirani infracrveni uzorak svjetlosti (desno)

[3] Slika u boji gdje se svaki piksel prikazuje vektorom od tri elementa: R (*red*) – vrijednost crvene boje, G (*green*) – vrijednost zelene boje i B (*blue*) – vrijednost plave boje.

Specifikacije Microsoft Kinect v1 kamere su sljedeće:

- vidno polje: 43° vertikalno i 57° horizontalno
- 30 slika po sekundi (engl. *frames per seconds*)
- rezolucija slike: 640 x 480 [px][4]
- 24-bitni raspon boje RGB kamere
- 16-bitni raspon dubine dubinskog senzora
- efektivni domet: 0.5 – 5 [m]

Odabir dobre lokacije za RGB-D kameru znatno utječe na rezultate sigurnosnog sustava, jer će dobro pozicionirana kamera obuhvatiti puno veći prostor u radnoj okolini robota, a smanjiti broj slučajeva zaklonjenosti čovjeka. Takvim postupkom daje se algoritmu upravljanja više vremena za reakciju, a samim time se povećava i sigurnost čovjeka. Lokacija kamere u sustavu trebala bi biti takva da vidno polje kamere „vidi" radno područje robota i što je više moguće prostora oko radnog područja robota u kojemu se mogu kretati ljudi.

4.3. Robotski operacijski sustav (*Robot Operating System* - ROS)

Iako se u imenu nalazi „operacijski sustav" (engl. *operating system*), ROS nije operacijski sustav nego *open-source* razvojni okvir (engl. *framework*) namijenjen razvoju programske podrške za robote [10]. Pruža servise za apstrakciju sklopovlja, upravljanje sklopovljem na niskoj programskoj razini, implementaciju često korištenih funkcionalnosti, slanje poruka između procesa te upravljanje paketima. Također pruža alate i biblioteke za dohvaćanje, izgrađivanje, pisanje i izvršavanje programskog koda preko više računala. U određenim aspektima, ROS je sličan drugim robotskim razvojnim okvirima kao što su *Player*, *YARP*, *Orocos*, *CARMEN*, *Orca*, *MOOS* i *Microsoft Robotics Studio*. ROS je primarno razvijen za Linux operacijske sustave (naročito Ubuntu) i nešto manje za Mac OS X sustave, dok opcija za uspostavu ROS-a na Windows sustavima nije do kraja realizirana. ROS podržava programske jezike *Python*, *C++* i *Lisp*. Glavni cilj ROS-a je podržavanje ponovnog korištenja programskih kodova (engl. *code reuse*) u istraživanju i razvoju robotike. Razvoj

[4] Piksel (engl. *pixel*)

programske podrške grupira se u pakete koji se jednostavno mogu dijeliti i distribuirati u zajednici, čime se postiže neovisnost u razvoju i implementaciji tih paketa za neke druge primjene.

Koncept ROS okvira podijeljen je na tri razine: datotečni sustav, računalni graf i zajednica.

Koncept datotečnog sustava u ROS-u čine ROS resursi koji se nalaze na disku računala kao što su: paketi, metapaketi, manifesti, repozitoriji te datoteke poruka i servisa. Paketi su osnovne organizacijske jedinice programske podrške u ROS-u. Paketi mogu sadržavati programske kodove (čvorove), biblioteke, konfiguracijske i razne druge datoteke koje su smisleno organizirane u jednu cjelinu. Metapaketi su posebni paketi koji služe da predstavljaju grupu drugih povezanih paketa. Manifest sadrži metapodatke o paketu kao što su ime paketa, verzija i opis paketa. Repozitorij je kolekcija više paketa koji dijele istu verziju i mogu se objavljivati zajedno. Datoteke poruka i servisa sadrže opise poruka i servisa koji se mogu izmjenjivati u ROS okviru.

Računalni graf je mreža ROS procesa koji zajedno obrađuju podatke. Osnovni koncepti računalnog grafa čine čvorovi (engl. *nodes*), *Master*, poslužitelj parametara (engl. *parameter server*), poruke, servisi, akcije, teme (engl. *topics*) i *rosbag* datoteke. Čvorovi su procesi koji izvode program. Svaki čvor izvršava određenu zadaću, a istovremeno se može izvršavati više čvorova. Primjerice jedan čvor služi za dohvaćanje slike s kamere, drugi čvor za upravljanje motorima kotača mobilnog robota, a treći čvor za lokalizaciju robota. *Master* omogućuje čvorovima da međusobno komuniciraju. Poslužitelj parametara je dio *Mastera*, a služi za posluživanje podataka koje koriste određeni čvorovi. Poruke su podatkovne strukture koje je moguće objavljivati ili ih koristiti. Podatkovna struktura poruke može biti raznolika, od jedne cjelobrojne (engl. *integer*) varijable, pa do složenih struktura koje se mogu sastojati od drugih podatkovnih struktura. Primjerice geometrijski tip poruke naziva Položaj (engl. *Pose*) je podatkovna struktura koja se sastoji od podatkovnih struktura Točke (engl. *Point*) i Orijentacije (engl. *Orientation*), a podatkovna struktura Točka se dalje sastoji od tri decimalna elementa *x, y* i *z*. Sve poruke koje se izmjenjuju u sustavu nalaze se na temama. Čvorovi koji objavljuju poruke na temu nazivaju se izdavači, a čvorovi koji koriste te podatke su pretplatnici. Primjerice prvi čvor objavljuje sliku s kamere na jednu temu, a drugi čvor je pretplaćen na tu temu i dohvaća sliku s kamere na kojoj onda vrši određene izračune. Servis je način komunikacije između čvorova gdje je jedan čvor poslužitelj (engl. *server*), a drugi klijent. Komunikacija se odvija tako da klijent šalje zahtjev i čeka odgovor od poslužitelja. *Rosbag* datoteke su strukture u kojima je moguće snimiti poruke s tema, te ih ponovno reproducirati. Korisne su kod testiranja, jer je moguće snimiti skup

podataka (engl. *dataset*) koji sadrži RGB slike, dubinsku sliku, oblak točaka i druge podatke, nakon čega se isti podaci mogu reproducirati za testiranje algoritama.

Koncept zajednice čine ROS resursi koji omogućuju različitim članovima zajednice dijeljenje i izmjenjivanje programske podrške i znanja.

5. IZRADA SIGURNOSNOG SUSTAVA

Izrada sigurnosnog sustava podijeljena je na nekoliko dijelova. Prvi dio predstavlja inicijalizacija sustava iza kojega slijede detekcija YOLO algoritmom i praćenje SORT Deep algoritmom. Nakon toga se određuje pozicija čovjeka u 3D prostoru, brzina i orijentacija kretanja čovjeka. Na kraju se računa koeficijent brzine alata robota.

Pri izradi, korišteno je svojstvo razvoja programske podrške u ROS-u, a to je podjela programa na više paketa. Svaki paket ima svoju ulogu u cijelom sustavu, a čvorovi svakog paketa međusobno komuniciraju.

5.1. Inicijalizacija scene

Inicijalizacija scene predstavlja spremanje statične dubinske slike pozadine. Pri inicijalizaciji, na sceni ne smiju biti ljudi niti drugi objekti u pokretu koji ne čine pozadinu. Inicijalizaciju je potrebno provesti samo jednom, pri čemu se spremljena dubinska slika statične scene učitava prilikom svakog uključivanja sigurnosnog sustava. U slučaju da se pozadina scene promijeni, potrebno je ponovno provesti inicijalizaciju. Spremljena slika statične dubinske slike pozadine dalje se koristi za određivanje dubina detektiranih ljudi tako da se uspoređuje dubinska slike na kojoj su ljudi sa slikom pozadine.

Čvor za inicijalizaciju scene pretplaćen je na temu dubinske slike Kinect kamere. Promatranjem toka dubinske slike (engl. *depth image stream*) kamere može se uočiti da slika „treperi“, odnosno na jednoj slici (engl. *frame*), određenim pikselima nije određena dubina, nego se vrijednost dubine postavi u *NaN* (*Not a Number*) vrijednost. Na idućoj slici dubine tih piksela su određene, a nekim drugima nisu. Zbog toga, inicijalizacija se ne izvodi samo na jednoj, nego na 51 slici. Dubinske slike spremaju se kao matrice, odnosno pomoću *OpenCV* biblioteke kao *cv::Mat* elementi. Nakon što se popunio vektor matrica dubinskih slika, prolazi se kroz svaki piksel slike te se za svaki piksel stvara vektor od 51 elementa koji sadrži vrijednosti dubine tog piksela na 51 slici. Zatim se traži medijan element iz tog vektora koji se sprema kao točna dubina tog piksela. Postupak određivanja medijana vrijednosti dubine svakog piksela prikazuje Slika 5.1.

```
for(int y = 0; y < height; y++){
	for(int x = 0; x < width; x++){
		for (int j = 0; j < maxFrames; j++) {
			array[j] = depthImageArray[j].at<float>(y,x);
		}
		value = findMedian(array);
		medianDepthImage.at<float>(y,x) = value;
	}
}
```

Slika 5.1. Određivanje medijana vrijednosti dubine

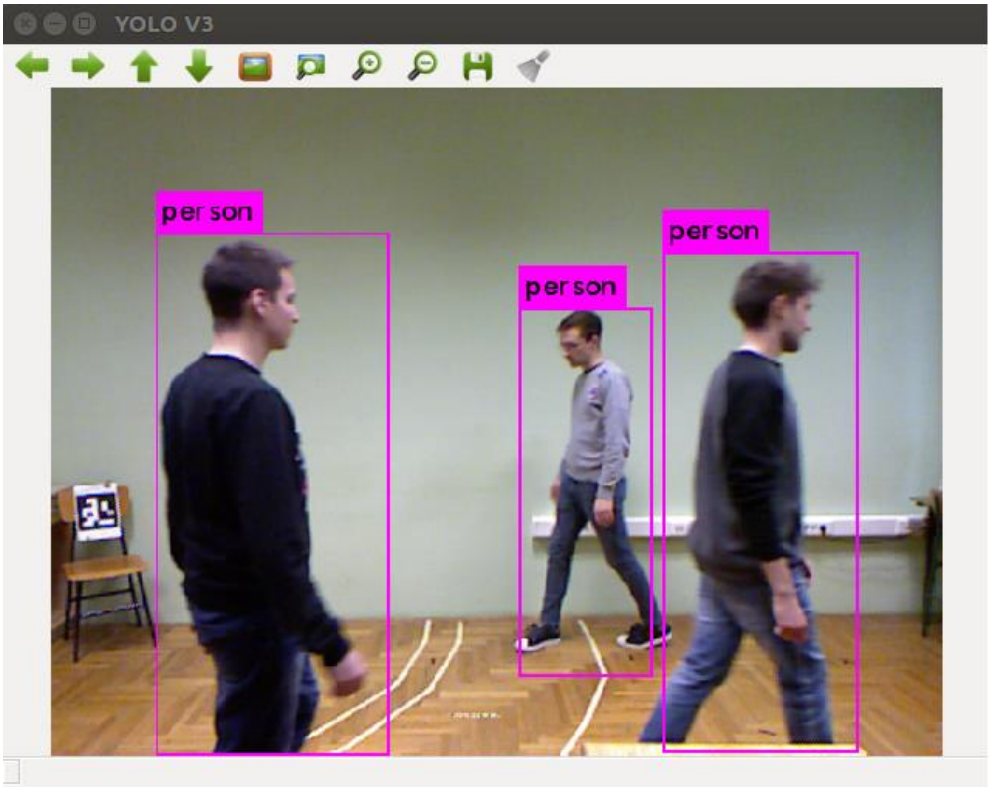
Rezultat je matrica dubina statične scene s puno manje šumova i smetnji, a spremljena je u *csv* formatu.

5.2. Detekcija i praćenje ljudi

Detekciju ljudi odrađuje YOLO algoritam. Na GitHubu postoji dostupan repozitorij čiji je autor YOLO algoritam oblikovao kao ROS paket.[5] Time je omogućeno da se YOLO izvodi u obliku čvora u ROS-u i komunicira s drugim čvorovima. Kako YOLO ima mogućnost detekcije raznih objekata, YOLO paket modificiran je tako da detektira samo ljude. Još je potrebno definirati ROS temu na koju će se YOLO pretplatit. Budući da YOLO radi detekciju ljudi na slikama u boji, pretplaćen je na temu s Kinect kamere koja daje RGB sliku. Rezultat YOLO detekcije su granični okviri oko detektiranih ljudi, što se može vidjeti na Slici 5.2. YOLO čvor objavljuje na tri teme: slika detekcija (Slika 5.2.), koordinate graničnih okvira te broj graničnih okvira na slici detekcije.

Za detekciju u ovome radu korištena je inačica YOLO algoritma nazvana *YOLOv2 tiny*. Ova verzija ima malo lošiju točnost, ali je mnogo brža od standardne YOLO verzije i pruža mogućnost detekcije u stvarnom vremenu. Međutim, sam YOLO algoritam nije dovoljan jer ne pruža mogućnost praćenja ljudi. Stoga je potrebno implementirati dodatni algoritam koji svakom detektiranom objektu pridružuje određeni identifikator.

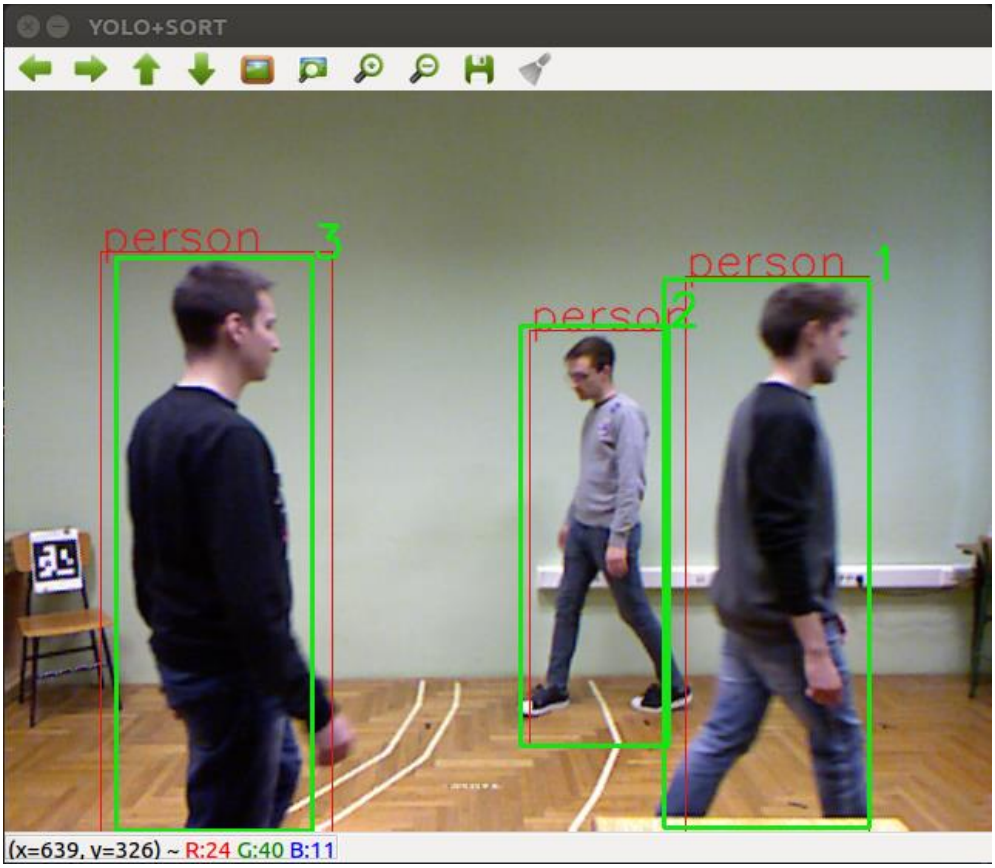
[5] https://github.com/leggedrobotics/darknet_ros



Slika 5.2. Detekcija YOLO algoritmom

Za praćenje ljudi korišten je SORT Deep algoritam. Kao i YOLO, SORT Deep algoritam također je dostupan na GitHub repozitoriju u obliku ROS paketa.[6] Potrebno je definirati teme na koje će se SORT Deep pretplatiti, a to su RGB slika i koordinate graničnih okvira dobivenih YOLO detekcijom. Rezultat SORT Deep algoritma prikazan je na Slici 5.3. Na slici se mogu vidjeti po dva granična okvira oko svakog čovjeka, od čega je jedan (crveni) rezultat YOLO, a drugi (zeleni) rezultat SORT Deep algoritma. Također, moguće je uočiti da sada postoje jedinstveni identifikatori za svaki pojedini granični okvir čime je omogućeno pratiti ljude. Čvor SORT Deep algoritma objavljuje na temu pozicije novih graničnih okvira uz pripadnu identifikaciju svakog.

[6] https://github.com/ilyasmg/sort-deepsort-yolov3-ROS



Slika 5.3. Praćenje ljudi SORT Deep algoritmom

5.3. Određivanje pozicije čovjeka

Određivanje pozicije čovjeka napravljeno je u jednom čvoru. Ovaj čvor objedinjuje spremljenu inicijalnu dubinsku sliku, detekciju ljudi YOLO algoritmom i praćenje ljudi SORT Deep algoritmom na temelju kojih računa poziciju i brzinu kretanja čovjeka, a na kraju i samu vrijednost koeficijenta brzine alata robota. Čvor je ujedno i izdavač i pretplatnik na određene teme, a strukturiran je kao klasa nazvana *BoundingBoxDepthClass*. Deklaracija klase, njenih metoda i atributa nalazi se u datoteci zaglavlja (engl. *header file*) *BoundingBoxDepthClass.hpp*. Osim toga, u datoteci zaglavlja još se nalazi i uvoz (engl. *include*) drugih potrebnih biblioteka. Dio kôda koji se bavi uvozom potrebnih biblioteka prikazuje Slika 5.4.

```
#ifndef BOUNDINGBOXDEPTHCLASS_HPP
#define BOUNDINGBOXDEPTHCLASS_HPP

#include <stdio.h>
#include <iostream>
#include <string>
#include <fstream>
#include <algorithm>
#include <boost/algorithm/string.hpp>
#include <vector>
#include <math.h>
#include <time.h>
#include <chrono>

/* OpenCV */
#include <cv_bridge/cv_bridge.h>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/core.hpp>

/* ROS */
#include <ros/ros.h>
#include "sensor_msgs/Image.h"
#include <image_transport/image_transport.h>
#include <std_msgs/String.h>
#include <std_msgs/Float32.h>
#include <geometry_msgs/Point.h>
#include <std_msgs/Int32.h>
#include <std_msgs/Int64.h>
#include <boost/thread.hpp>
#include <ros/package.h>

/* Yolo package */
#include "darknet_ros_msgs/BoundingBoxes.h"
#include "darknet_ros_msgs/BoundingBox.h"
#include "darknet_ros_msgs/ObjectCount.h"

/* Deep SORT package */
#include "sort_track/IntList.h"
#include "sort_track/TrackBoundingBox.h"
#include "sort_track/TrackBoundingBoxes.h"
#include "sort_track/TrackCount.h"

/* BBox depth package */
#include "bbox_depth/Regulator.h"
```

Slika 5.4. Uvoz potrebnih biblioteka

Uvoz biblioteka grupiran je u više dijelova kako bi čitanje kôda bilo jednostavnije. Biblioteke koje se uvoze su opće biblioteke koje koristi C++ programski jezik, kao što su *string, fstream, math.h* i *time.h* bibilioteke. Zatim, uvoze se i datoteke potrebne za rad s *OpenCV* i ROS bibliotekama. Na kraju se uvoze i datoteke pojedinih paketa o kojima ovisi rad ovog čvora (engl. *dependencies*). Te datoteke

uglavnom predstavljaju datoteke poruke koje će se izmjenjivati na temama u ROS okviru tijekom rada cijelog sustava.

U datoteci zaglavlja deklarirana je struktura *BBox* koja predstavlja granični okvir. Deklaracija *BBox* strukture prikazuje Slika 5.5.

```
struct BBox{
	int id, xmin, xmax, ymin, ymax, center_x, center_y;
	float depth, depth_prev;
	float x, y, z;
	float x_prev, y_prev, z_prev;
	ros::Time time_current, time_prev;
};
```

Slika 5.5. Deklaracija BBox strukture

Granični okvir definiran je:

- identifikatorom (*id*)
- 2D koordinatama rubova (*xmin*, *xmax*, *ymin*, *ymax*) i središta (*center_x*, *center_y*) na slici
- dubinom središta na trenutnoj slici (*depth*) i na prethodnoj slici(*depth_prev*)
- 3D koordinatama središta na trenutnoj (*x*, *y*, *z*) i prethodnoj slici (*x_prev*, *y_prev*, *z_prev*)
- vremenima pojave graničnog okvira na trenutnoj i prethodnoj slici

3D koordinate ustvari predstavljaju koordinatni sustav čovjeka u odnosu na koordinatni sustav kamere kojim je estimirana pozicija čovjeka.

U datoteci zaglavlja deklarirana je još jedna struktura *Robot* (Slika 5.6.).

```
struct Robot{
	float x, y, z;
	float r;
} robot;
```

Slika 5.6. Deklaracija Robot strukture

Struktura *Robot* sadrži parametre kojima je opisan robot:

- pozicija koordinatnog sustava baze robota u odnosu na koordinatni sustav kamere (*x*, *y*, *z*)
- polumjer radnog područja robota *(r)*

Pokretanje čvora napravljeno je u *BoundingBoxDepth_node.cpp* datoteci. Pokretanje čvora prikazuje Slika 5.7.

```
#include "BoundingBoxDepthClass.hpp"

int main(int argc, char** argv)
{
	std::string node_name = "bbox_depth";
	ros::init(argc, argv, node_name);
	ros::NodeHandle nh("~");
	BoundingBoxDepthClass node(nh);
	ROS_INFO("Initialized single-thread class node.");
	ros::spin();

	return 0;
}
```

Slika 5.7. Pokretanje bbox_depth čvora

Naziv čvora je *bbox_depth*. Nakon inicijalizacije čvora, stvara se instanca *BoundingBoxDepthClass* klase kojoj se kao argument predaje referenca čvora (engl. *node handle*).

Glavni programski kôd i definicije metoda klase nalaze se u datoteci *BoundingBoxDepthClass.cpp*. Stvaranjem instance klase poziva se konstruktor. U konstruktoru se primi referenca čvora i definiraju se određene vrijednosti od kojih su najznačajnije dimenzija slike (*width, height*), intrinzični parametri Kinect kamere (*fx, fy, cx, cy*) te zadani polumjer radnog područja robota definiran dohvatom robota (*envelope_default_radius*). Još se inicijaliziraju i elementi *Robot* strukture.

```
BoundingBoxDepthClass::BoundingBoxDepthClass(const ros::NodeHandle &node_handle):
	nh(node_handle), width(640), height(480), fx(554.545), fy(552.145), cx(327.799),
	cy(236.814), envelope_default_radius(1.81)
{
	(…)
	nh.param("robot_position/x", robot.x, float(0.0));
	nh.param("robot_position/y", robot.y, float(0.0));
	nh.param("robot_position/z", robot.z, float(0.0));
	robot.r = envelope_default_radius;
	(…)
	loadInitialDepths(width, height);
	this->init();
}
```

Slika 5.8. Definicija konstruktora

Nakon definicije određenih vrijednosti u konstruktoru, pozivaju se dvije metode: *loadInitialDepths()* i *init()*.

LoadInitialDepths() metoda radi učitavanje spremljene inicijalne dubinske slike na disku iz *csv* formata u *initial_depths* matricu čije su dimenzije definirane veličinom slike (*width, height*).

```
void BoundingBoxDepthClass::loadInitialDepths(const int width, const int height)
{
	ifstream file;
	file.open(initial_depths_filename);
	for(int row = 0; row < height; row++) {
		string line;
		getline(file, line);
		if ( !file.good() )
			break;
		stringstream iss(line);
		for (int col = 0; col < width; col++) {
			string val;
			getline(iss, val, ',');
			if ( !iss.good() )
				break;
			stringstream convertor(val);
			convertor >> initial_depths[col][row];
		}
	}
	file.close();
}
```

Slika 5.9. Učitavanje inicijalne dubinske slike u matricu

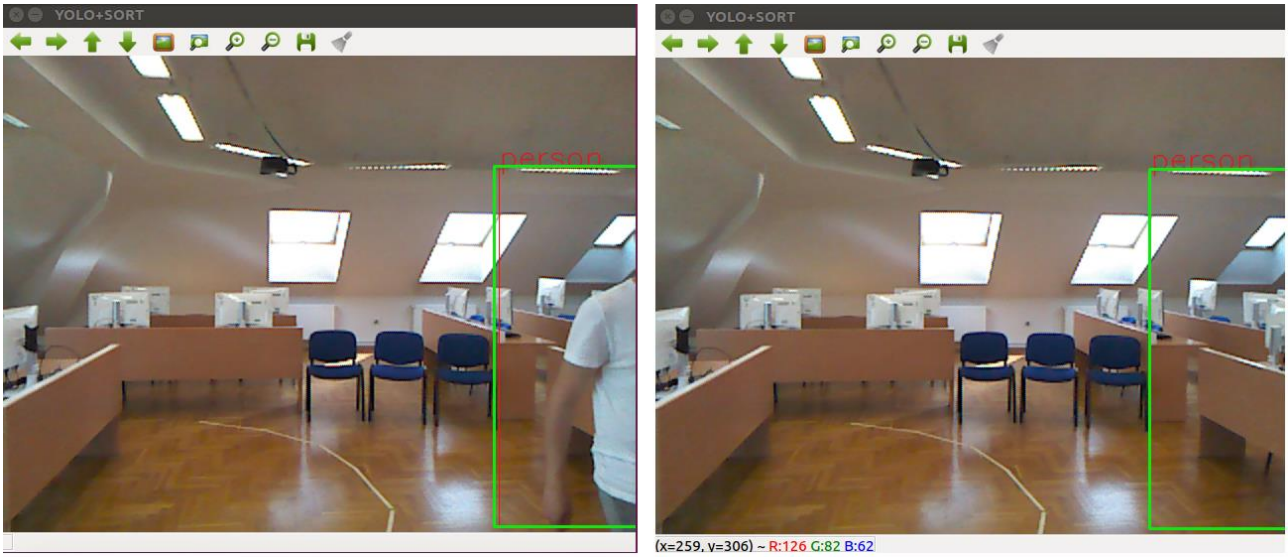
U *init()* metodi odvija se inicijalizacija nekih drugih parametara. Najvažnije elemente koji se postavljaju u *init()* metodi prikazuje Slika 5.10.

Kao što se može vidjeti na slici, u *init()* metodi dohvaćaju se imena tema na koje će se čvor pretplatiti, odnosno na koje će čvor objavljivati. Čvor je pretplaćen na dubinsku sliku Kinect kamere, brojač detektiranih ljudi na slici YOLO algoritmom te koordinate graničnih okvira dobivenih SORT Deep algoritmom, a objavljuje na temu kojom će se upravljati brzinom robota.

```
void BoundingBoxDepthClass::init()
{
	// Get topic names
	nh.param("subscribers/depth_img_topic", depth_img_topic,
			std::string("/camera/depth/image"));

	nh.param("subscribers/yolo_count_topic", yolo_count_topic,
			std::string("/darknet_ros/found_object"));

	nh.param("subscribers/tracker_bbox_topic", tracker_bbox_topic,
			std::string("/sort_track/bounding_boxes"));

	nh.param("publishers/regulator_topic", regulator_topic,
			std::string("/bbox_depth/regulator_value"));

	regulator_publisher = nh.advertise<bbox_depth::Regulator>(regulator_topic, 1);

	sub_yolo_count = nh.subscribe(yolo_count_topic,1,
				&BoundingBoxDepthClass::yoloCountCallback, this);

	sub_tracker_bbox = nh.subscribe(tracker_bbox_topic,1,
				&BoundingBoxDepthClass::trackerBboxCallback, this);

	image_transport::ImageTransport it(nh);

	sub_depth_image = it.subscribe(depth_img_topic, 1,
				&BoundingBoxDepthClass::depthImageCallback, this);
	(…)
}
```

Slika 5.10. Definicija init() metode

Dohvaćanje broja detektiranih ljudi na slici odvija se u *yoloCountCallback()* metodi (Slika 5.12). SORT Deep algoritam ima tendenciju ostaviti lažno pozitivan granični okvir kada čovjek izađe sa slike. U tom slučaju se pozicija graničnog okvira zaustavlja na zadnjoj poziciji gdje je čovjek bio detektiran prije nego je izašao. Ovaj problem događa se samo kada sa slike izađe čovjek koji je bio jedini na slici. Primjer ovog problema može se vidjeti na Slici 5.11.



Slika 5.11. Problem lažno pozitivne detekcije SORT Deep algoritma

Zbog toga, u metodi *yoloCountCallback()* dodatno se provjerava koliko je slika (engl. *frames*) proteklo od zadnje detekcije YOLO algoritmom.

```
void BoundingBoxDepthClass::yoloCountCallback(const
                                darknet_ros_msgs::ObjectCount::ConstPtr& msg)
{
	current_frame = frame_counter;
	if(msg->count) {
		last_frame = current_frame;
	}
	frame_diff = abs(current_frame - last_frame);
	if(frame_diff > frame_threshold && !msg->count) {
		frame_counter = 0;
	}
}
```

Slika 5.12. Dohvaćanje broja detektiranih ljudi

U *trackerBboxCallback()* metodi dohvaćaju se koordinate graničnih okvira dobivenih SORT Deep algoritmom (Slika 5.13.). Najprije se provjerava koliko je vremena prošlo od zadnje detekcije YOLO algoritmom što je izračunato u prethodnoj metodi. Ako je prošlo više vremena od definirane granice, tada se postavlja broj detektiranih ljudi na 0. Time je riješen problem lažno pozitivne detekcije SORT Deep algoritma kada čovjek izađe sa scene.

U slučaju da su ljudi na slici, popunjavaju se elementi *BBox* strukture parametrima koji su dobiveni SORT Deep algoritmom. Tu se podrazumijevaju identifikatori graničnih okvira, 2D koordinate te centri graničnih okvira na slici. Dodatno je definiran poseban uvjet koji služi da ograniči koordinate pozicije graničnih okvira unutar dimenzija slike.

```
void BoundingBoxDepthClass::trackerBboxCallback(const
                            sort_track::TrackBoundingBoxes::ConstPtr& msg)
{
	if(frame_diff <= frame_threshold)
		bbox_count = msg->bounding_boxes.size();
	else
		bbox_count = 0;
	human_idx_array.clear();
	for(int i = 0; i < bbox_count; i++) {
		int idx = msg->bounding_boxes[i].id;
		human_idx_array.push_back(idx);

		bbox[idx].id = msg->bounding_boxes[i].id;
		bbox[idx].ymin = msg->bounding_boxes[i].ymin;
		bbox[idx].ymax = msg->bounding_boxes[i].ymax;
		bbox[idx].xmin = msg->bounding_boxes[i].xmin;
		bbox[idx].xmax = msg->bounding_boxes[i].xmax;

		if(bbox[idx].xmin <= 0) bbox[idx].xmin = 0;
		if(bbox[idx].xmax >= 640) bbox[idx].xmax = 639;
		if(bbox[idx].ymin <= 0) bbox[idx].ymin = 0;
		if(bbox[idx].ymax >= 480) bbox[idx].ymax = 479;

		bbox[idx].center_x = (int) bbox[idx].xmin + (bbox[idx].xmax -
                                                        bbox[idx].xmin) / 2;

		bbox[idx].center_y = (int) bbox[idx].ymin + (bbox[idx].ymax -
                                                        bbox[idx].ymin) / 2;
	}
}
```

Slika 5.13. Dohvaćanje pozicija graničnih okvira

Glavna metoda ovog čvora je *depthImageCallback()* metoda koja se poziva za svaku dubinsku sliku iz toka snimanja dubinske slike Kinect kamerom. U ovoj metodi se odvija određivanje pozicije, računa se brzina kretanja čovjeka i vrijednost koeficijenta brzine alata robota.

Slika 5.14. prikazuje algoritam koji određuje je li određeni piksel na dubinskoj slici predstavlja čovjeka ili pozadinu.

```
for(const int& idx : human_idx_array) {
	human_depth_array.clear();
	float initial_depth = 0.0;
	float current_depth = 0.0;
	float previous_depth = 0.0;
	int j_previous = 0;
	for(int j = bbox[idx].ymin; j < bbox[idx].ymax; j++) {
		int i_previous = 0;
		for(int i = bbox[idx].xmin; i < bbox[idx].xmax; i++) {
			bool human_point = false;
			if( depth_image.at<float>(j,i) != 0 &&
					!isnan(depth_image.at<float>(j,i)) ) {
				current_depth = findAverage(i,j,1);
				initial_depth = findAverage(i,j,0);

				float diff = initial_depth - current_depth;
				if(diff > 0.2) {
					human_point = true;
				}
				if (initial_depth == 0) {
					if(abs(previous_depth-current_depth) < 0.1) {
						human_point = true;
					}
					else if ( abs(previous_depth-current_depth) < 0.3 &&
						(i-i_previous == 0 || j-j_previous == 0)) {
						human_point = true;
			}}}
			if(human_point)
			{
				i_previous = i;
				j_previous = j;
				previous_depth = current_depth;
				human_depth_array.push_back(depth_image.at<float>(j, i));
	}}}
```

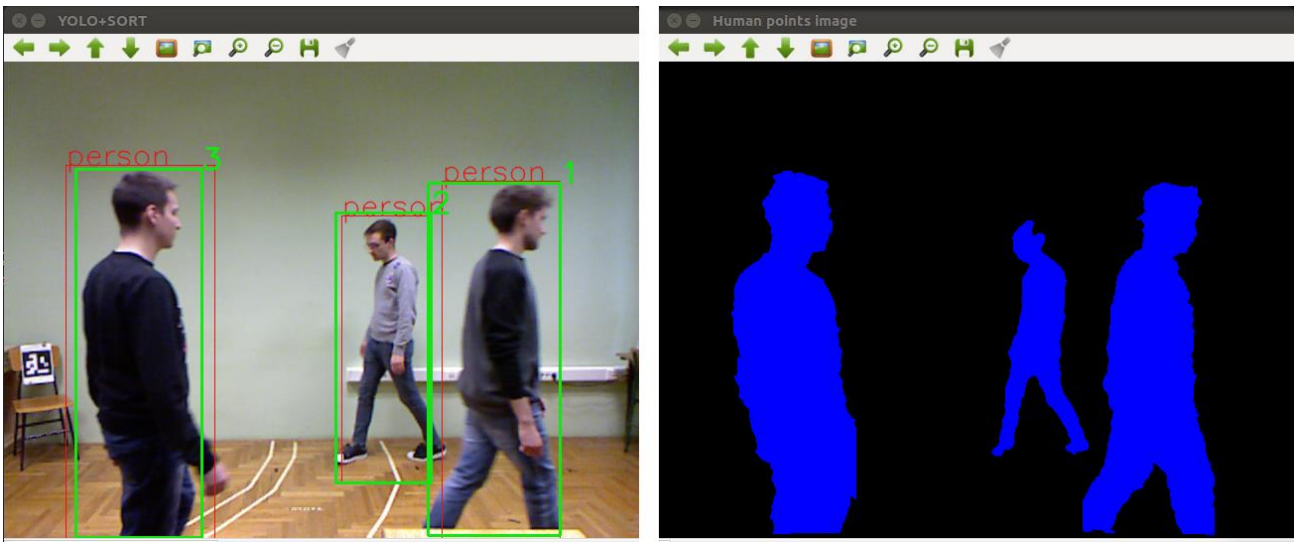
Slika 5.14. Određivanje piksela na dubinskoj slici koji predstavljaju čovjeka

Algoritam pretražuje piksele za svaki granični okvir, odnosno pretražuje samo one piksele koji se nalaze unutar graničnog okvira. Na slici se može vidjeti da se poziva funkcija *findAverage()* prilikom dohvaćanja trenutne i inicijalne dubine za jedan piksel. Funkcija *findAverage()* usrednjava dubinu piksela s obzirom na okolinu tog piksela. Radi se o okolini veličine bloka 3x3. Ovaj postupak se koristi kako bi se izbjegle smetnje i šumovi Kinect dubinskog senzora na način da se koristi informacija i o susjednim pikselima. Zatim se provjerava razlika trenutne i inicijalne dubine na temelju koje se određuje pripada li taj piksel čovjeku.

Kako je Kinect dubinski senzor ograničen na efektivni domet od 5 metara i ne može odrediti dubine na mjestima gdje se događa svjetlosni odsjaj na površini, takve vrijednosti dubine budu 0 ili *NaN*. Ako se radi o pikselu pozadine, tada se javlja problem kada na inicijalnom pikselu nije određena vrijednost

dubine, a na trenutnom pikselu je određena. Onda će razlika između inicijalne i trenutne dubine biti velika. Primjerice da trenutni piksel na pozadini ima vrijednost dubine 5 metara, a na inicijalnoj slici nije određena dubina tog piksela, tada će razlika biti |0-5| = 5 metara. Međutim, može se dogoditi da na tom istom pikselu na trenutnoj slici nije pozadina već čovjek. Tu se onda javlja problem kako odrediti je li takva razlika dubine pozadina ili čovjek. Taj problem je riješen tako da se uvijek oduzima dubina trenutnog piksela od inicijalnog, što znači da piksel koji predstavlja čovjeka ne može imati dubinu veću od pozadine. U ovome slučaju, ako postoji razlika veća od 0.2 metra, onda taj piksel pripada čovjeku. Ako dubina piksela inicijalne slike nije definirana (iznosi 0), tada se provjerava zadnja dubina piksela koja ja predstavljala čovjeka. To znači ako je trenutna dubina piksela za 0.1 metar različita od prethodne dubine piksela čovjeka, onda će se smatrati da i trenutna dubina pripada čovjeku. Idući uvjet povećava granicu prethodne razlike na 0.3 metra, ali tada se mora raditi o pikselima koji su susjedni. Ovaj uvjet obuhvaća situacije kada je čovjek ispružio ruku ispred sebe, pa je razlika dubine između njegovog tijela i ruke veća.

Slika 5.15 prikazuje rezultat ovog postupka. Na desnoj strani slike plavom bojom označeni su pikseli koji predstavljaju čovjeka, dok je crnom bojom prikazana pozadina.



Slika 5.15. Detekcija ljudi na RGB slici (lijevo) i estimacija čovjeka na dubinskoj slici (desno)

Nakon što je algoritam prošao kroz sve piksele unutar graničnog okvira i spremio sve dubine koje predstavljaju čovjeka u vektor, traži se jedna vrijednost dubine koja će estimirati dubinu cijelog čovjeka. To je napravljeno tako da se sortira cijeli vektor dubina i uzme srednji element sortiranog vektora. U slučaju da je pronađena samo jedna dubina, tada se ona uzima kao estimirana vrijednost

dubine čovjeka, a u slučaju da nije određena niti jedna dubina, odnosno ako je vektor dubina prazan, tada se dubina postavlja na prethodnu vrijednost. Ovaj postupak prikazuje Slika 5.16.

```
if(human_depth_array.size()>1) {
	sort(human_depth_array.begin(), human_depth_array.end());
	int middle_index = (int) human_depth_array.size() / 2;
	bbox[idx].depth = human_depth_array.at(middle_index-1);
}
else if (human_depth_array.size() == 1) {
	bbox[idx].depth = human_depth_array.at(0);
}
else {
	bbox[idx].depth = bbox[idx].depth_prev;
}
bbox[idx].depth_prev = bbox[idx].depth;
```

Slika 5.16. Estimacija dubine čovjeka jednom vrijednošću

Nakon toga, poznavanjem koordinata središta graničnog okvira na slici i njegove dubine, mogu se odrediti koordinate točke u koordinatnom sustavu kamere prema sljedećim formulama:

$$x = \frac{z(u - c_x)}{f_x}, y = \frac{z(v - c_y)}{f_y} \tag{5.1}$$

gdje su (u, v) koordinate središta graničnog okvira na slici, (f_x, f_y, c_x, c_y) intrinzični parametri dubinskog senzora Kinect kamere, a (x, y, z) koordinate koje predstavljaju poziciju čovjeka u koordinatnom sustavu kamere.

Kada je određena pozicija čovjeka na svakoj slici, može se izračunati njegova udaljenost od robota, te njegova brzina na temelju više slika.

5.4. Određivanje brzine kretanja čovjeka

Izračun brzine kretanja čovjeka (Slika 5.17.) vrlo je jednostavan. Pamti se prethodna pozicija čovjeka, a kada prođe dovoljno vremena (0.3 sekunde), računa se Euklidska udaljenost prethodne i trenutne pozicije čovjeka. Izračunata udaljenost se podijeli s proteklim vremenom čime se dobije brzina čovjeka.

```
double time_d = bbox[idx].time_current.toSec() - bbox[idx].time_prev.toSec();
if(time_d > 0.3) {
	// Human (operator) velocity
	double human2human_prev_distance =
		calculateDistance(bbox[idx].x,bbox[idx].y,bbox[idx].z,
				bbox[idx].x_prev,bbox[idx].y_prev,bbox[idx].z_prev,0);

	double velocity = calculateVelocity(human2human_prev_distance, time_d);
```

Slika 5.17. Izračun brzine kretanja čovjeka

Zatim se provjerava uvjet (2.1) koji se zasniva na brzini čovjeka (Slika 5.18.) i mijenja polumjer radnog područja robota s obzirom na brzinu kretanja čovjeka.

```
if (velocity < 0.1)
	robot.r = envelope_default_radius;
else if (velocity >= 0.1 && velocity < 1)
	robot.r = envelope_default_radius * 1.1;
else
	robot.r = envelope_default_radius * 1.5;
```

Slika 5.18. Provjera uvjeta (2.1)

Nakon toga, slijedi izračun tangencijalne i radijalne brzine. Radijalna brzina računa se tako da se odredi udaljenost čovjeka od robota u trenutnom i prošlom trenutku. Ta udaljenost se dijeli s proteklim vremenom i dobije se radijalna brzina. Ako je ova brzina pozitivnog predznaka, to znači da se čovjek kreće prema robota, i obratno.

Ako se radijalna V_r, tangencijalna V_t i ukupna brzina V_o čovjeka predstave vektorima, tada se može vidjeti da je vektor ukupne brzine ustvari zbroj vektora tangencijalne i radijalne brzine: $V_o = V_r + V_t$. Budući da su vektori radijalne i tangencijalne brzine međusobno okomiti, vektori čine pravokutni trokut u kojem su V_r i V_t katete, a V_o hipotenuza. Prema tome, iznos tangencijalne brzine može se izračunati kao: $V_t = \sqrt{|V_o|^2 - |V_r|^2}$. Slika 5.19. prikazuje izračune ovih brzine te izračun udaljenosti čovjeka od radnog područja robota (udaljenost D sa slike 3.2.) koja je u kôdu na slici označena kao *human2envelope_distance*.

```
double human2robot_distance = calculateDistance(bbox[idx].x,bbox[idx].y,bbox[idx].z,
                                                robot.x,robot.y,robot.z,1);

double human2robot_distance_prev = calculateDistance(bbox[idx].x_prev,bbox[idx].y_prev,
                                            bbox[idx].z_prev,robot.x,robot.y,robot.z,1);

double human2envelope_distance = human2robot_distance - robot.r;
double radial_distance = human2robot_distance_prev - human2robot_distance;
double radial_velocity = calculateVelocity(radial_distance, time_d);
double tangential_velocity = sqrt(pow(velocity,2) - pow(radial_velocity,2));
```

Slika 5.19. Izračun radijalne i tangencijalne brzine čovjeka

5.5. Određivanje koeficijenta brzine alata robota

Idući korak je provjera uvjeta (2.2)(2.2). Uvjeti su temeljeni na brzini kretanja čovjeka. Veća brzina kretanja čovjeka uzrokuje manje vrijednosti koeficijenta brzina alata robota (V_{ee}) koja je u kôdu na Slici 5.20. označena kao *regulator_value*. U uvjetu za potpuno zaustavljanje robota mjeri se kut između vektora radijalne i ukupne brzine čovjeka (kut φ) te kut između vektora radijalne brzine V_r i tangente na kružnicu u točki pozicije čovjeka (kut α). Ako je kut $\varphi \leq \alpha$ to znači da se čovjek kreće prema radnom području robota (zorniji prikaz kuteva α i φ može se vidjeti na Slici 3.2.).

```
int regulator_value = 100;
if(human2envelope_distance < 0) {
	regulator_value = 0;
}
else {
	if(velocity < 0.1) {
		regulator_value = 50;
	}
	else if(velocity >= 0.1 && velocity < 2.0) {
		regulator_value = 25;
	}
	else if(velocity >= 2.0) {
		regulator_value = 0;
	}
	else {
		regulator_value = 100;
	}

	if(radial_velocity > 0) {
		// * Angle between velocity vector and radial velocity vector
		double phi = acos((velocity_vector[0]*radial_velocity_vector[0] +
					velocity_vector[1]*radial_velocity_vector[1] +
					velocity_vector[2]*radial_velocity_vector[2])
				/ (sqrt(pow(velocity_vector[0],2)+pow(velocity_vector[1],2) +
				  pow(velocity_vector[2],2))
			* sqrt(pow(radial_velocity_vector[0],2) +
			  pow(radial_velocity_vector[1],2) +
			  pow(radial_velocity_vector[2],2))));

		// * Angle between |human to robot| line and
		// * tangent from human point to robot's envelope
		double alpha = asin(robot.r / human2robot_distance);
		if(phi <= alpha) {
			regulator_value = 0;
		}
	}
}

regulator_array.push_back(regulator_value);
```

Slika 5.20. Provjera uvjeta (2.2)

Nakon što je određena vrijednost koeficijenta brzine alata robota prema navedenim uvjetima, ta vrijednost se sprema u vektor. Spremanje u vektor radi se jer se koeficijent brzine alata robota računa za svakog čovjeka. U slučaju kada je više ljudi na sceni, tada se u obzir treba uzeti najstroži uvjet, odnosno najmanji koeficijent brzine robota. Zbog toga se koeficijenti spremaju u vektor pa se na temu objavljuje samo najmanja vrijednost koeficijenta. Dio kôda koji objavljuje na temu ovu vrijednost prikazuje Slika 5.21.

```
if(!regulator_array.empty()) {
	int min_regulator_value = regulator_array.at(0);
	for(int i=0;i<regulator_array.size();i++) {
		if(regulator_array.at(i) < min_regulator_value)
			min_regulator_value = regulator_array.at(i);
	}
	regulator.value = min_regulator_value;
	regulator.header.stamp = ros::Time::now();
	regulator.header.frame_id = "Regulator value";
	regulator_publisher.publish(regulator);
```

Slika 5.21. Objavljivanje na temu najmanje vrijednosti koeficijenta brzine alata robota

Uz vrijednost koeficijenta, objavljuju se još i dodatne informacije porukama zaglavlja.

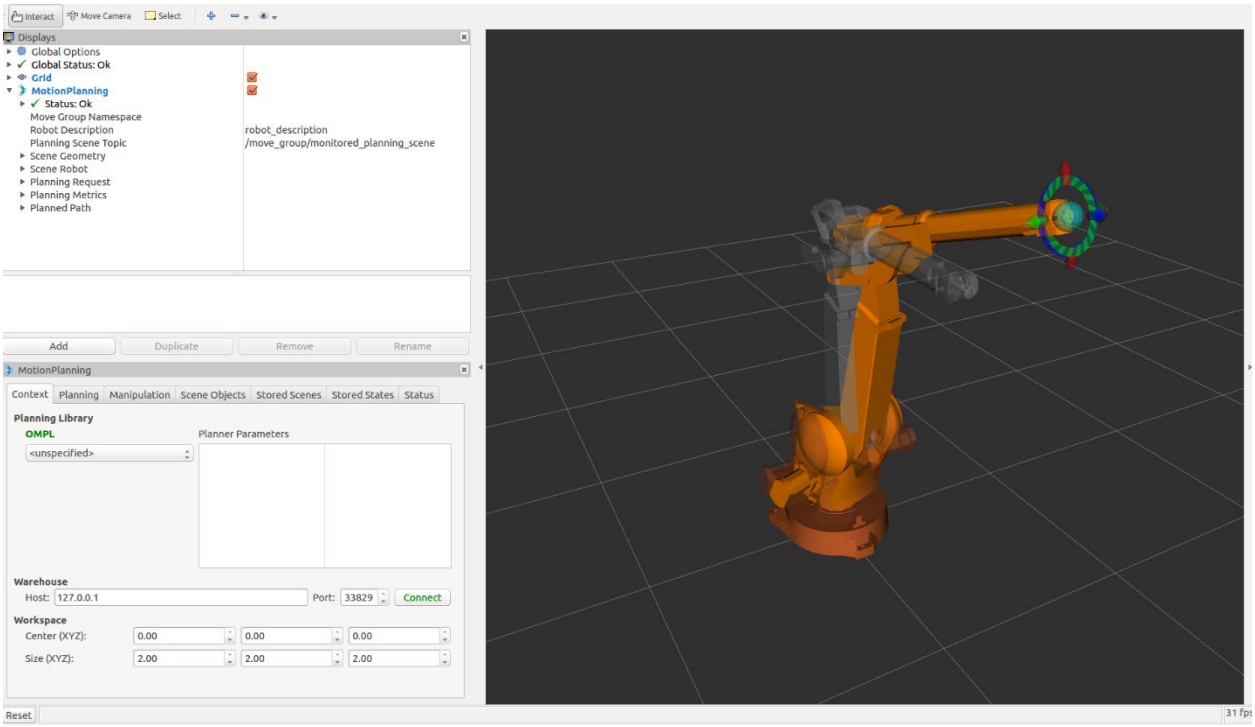
5.6. Upravljanje robotskim manipulatorom

Upravljanje robotskim manipulatorom u ROS-u omogućeno je korištenjem ROS-Industrial paketa. ROS-Industrial povezuje industrijske robote s drugim sklopovljem i omogućuje komunikaciju industrijskog robota s ROS čvorovima. Komunikacija se izvodi preko konfiguracijskog paketa koji je jedinstven za svakog robota. Konfiguracijski paket predstavlja skup sistemskih datoteka, pogonskih programa i drugih značajnih datoteka koje služe za upravljanje robotom računalnim programom napisanim u ROS okviru. Za većinu robotskih manipulatora postoje gotovi konfiguracijski paketi na Internetu. Za robotski manipulator korišten u ovome radu ABB IRB 2400L ne postoji gotovi konfiguracijski paket, ali postoji paket za njemu sličan model ABB IRB 2400. Autor rada [11] napravio je modifikaciju dostupnog paketa i prilagodio ga za korišteni model robotskog manipulatora u ovome radu.

Pokretanje paketa za upravljanje robotskim manipulatorom radi se u terminalu naredbom:

```
roslaunch abb_irb2400_moveit_config moveit_planning_execution.launch sim:=false robot_ip:=192.168.1.42
```

Na slici 5.22. može se vidjeti prikaz robota u *Rviz* alatu ROS-a nakon što se pokrenuo konfiguracijski paket. U alatu je moguće ručno pomicati alat robota ili izabrati nasumični položaj. Međutim, moguće je upravljati robotom i putem ROS čvora.



Slika 5.22. Vizualizacija robotskog manipulatora u Rviz alatu

ROS čvor za upravljanje robotskim manipulatorom napravljen je u novom paketu nazvanom *abb_control*. Paket se sastoji od jednog čvora *infinite_movement* čija je zadaća upravljati kretanjem robota beskonačno u par definiranih točaka. Čvor je pretplaćen na temu na kojoj se nalazi podatak o koeficijentu brzine alata robota izračunatom u *bbox_depth* čvoru.

Kretanje robota odvija se uz pomoć *MoveIt* sučelja *MoveGroupInterface*, a sučelje *PlanningSceneInterface* služi za planiranje scene. Postavljanje ovih sučelja i drugih parametara prikazuje Slika 5.23.

```
static const std::string PLANNING_GROUP = "manipulator";
moveit::planning_interface::MoveGroupInterface move_group(PLANNING_GROUP);
moveit::planning_interface::PlanningSceneInterface planning_scene_interface;
const robot_state::JointModelGroup* joint_model_group =
       move_group.getCurrentState()->getJointModelGroup(PLANNING_GROUP);
moveit::planning_interface::MoveGroupInterface::Plan my_plan;
moveit::core::RobotStatePtr current_state = move_group.getCurrentState();
std::vector<double> joint_group_positions;
current_state->copyJointGroupPositions(joint_model_group, joint_group_positions);
```

Slika 5.23. Postavljanje sučelja za upravljanje kretanjem robota

Nakon toga, mogu se definirati kretnje robotskog manipulatora. Na Slici 5.24. može se vidjeti dio kôda koji definira osam koraka u kretanju robota.

```
int step = 0;
while(ros::ok())
{
	ros::spinOnce();
	if(ros::Time::now().toSec() - ::regulator_time.toSec() > 5.0){
		regulator_value = 100;
	}

	if(regulator_value == 0)
		continue;
	else if(regulator_value == 25)
		move_group.setMaxAccelerationScalingFactor(0.5);
	else if(regulator_value == 50)
		move_group.setMaxAccelerationScalingFactor(0.75);
	else
		move_group.setMaxAccelerationScalingFactor(1.0);


	if(step==0)
	{
		joint_group_positions[0] = -0.8;
		joint_group_positions[1] = -0.2;
	}
	(…)
	}
	else if(step==7)
	{
		joint_group_positions[0] = -0.4;
		joint_group_positions[1] = 0.1;
	}
	move_group.setJointValueTarget(joint_group_positions);
	move_group.plan(my_plan);
	move_group.move();
	step++;
	if(step >=8) step = 0;
}
```

Slika 5.24. Definiranje pokreta robota

Prije svakog planiranja i izvršavanja kretnje robota, poziva se *callback* funkcija koja s teme učitava vrijednost koeficijenta brzine alata robota i sprema tu vrijednost u varijablu *regulator_value*. Ta varijabla onda određuje kojom će se brzinom kretati vrh alata robota. U svakom koraku pomiču se samo prva dva zgloba robota čija je kretnja zadana u radijanima. Time je napravljeno beskonačno kretanje robota u definiranih osam točaka, čime je jednostavno testirati izrađeni sustav prateći usporavanje ili zaustavljanje robota u određenim uvjetima.

5.7. Pokretanje sustava

ROS omogućuje pokretanje više čvorova istovremeno pomoću tzv. *launch* datoteka. *Launch* datoteke koriste XML opisni jezik. Čvorovi koji se pokreću mogu biti iz različitih paketa. Na Slici 5.25. prikazana je *launch* datoteka koja pokreće cijeli sustav.

```
<launch>

	<include file="$(find openni_launch)/launch/openni.launch" />

	<include file="$(find darknet_ros)/launch/darknet_ros_custom.launch" />

	<include file="$(find sort_track)/launch/sort_deep.launch" />

	<node pkg="bbox_depth" type="bbox_depth" name="bbox_depth"  output="screen">
		<rosparam command="load" file="$(find bbox_depth)/config/bbox_depth.yaml"/>
	</node>

</launch>
```

Slika 5.25. Launch datoteka za pokretanje sustava

U *launch* datotekama moguće je uključiti i druge *launch* datoteke koje se nalaze u različitim paketima. U ovom sustavu pokreću se sljedeće *launch* datoteke:

- *openni.launch* – pokretanje Kinect kamere
- *darknet_ros_custom.launch* – pokretanje modificiranog YOLO algoritma
- *sort_deep.launch* – pokretanje SORT Deep algoritma

Uz to se još pokreće i čvor *bbox_depth* koje je prethodno objašnjen. Pokretanjem *launch* datoteke moguće je učitati poslužitelj parametara u ROS okviru naredbom *rosparam command="load"*. Parametri mogu biti zapisani u samoj *launch* datoteci ili se mogu zapisati u zasebnoj datoteci *YAML* formata. Na Slici 5.25. može se vidjeti da se poziva poslužitelj parametara za *bbox_depth* čvor koji će moći koristiti te parametre u izvođenju čvora. Pri tome se poziva datoteka *bbox_depth.yaml*. Sadržaj ove datoteke prikazan je na Slici 5.26.

```
init_depths_file: /init/kinect_depth_init.csv

robot_position:
       x: -2.22
       y: 0.49
       z: 4.06

# Input topics
subscribers:
       depth_img_topic: /camera/depth_registered/image
       yolo_count_topic: /darknet_ros/found_object
       tracker_bbox_topic: /sort_track/bounding_boxes
       queue_size: 1

# Output topic
publishers:
       regulator_topic: /bbox_depth/regulator_value
```

Slika 5.26.YAML datoteka s potrebnim parametrima

YAML datoteka sadrži putanju i ime spremljene dubinske slike statične scene, koordinate pozicije robota u koordinatnom sustavu kamere te nazive tema na koje se čvor pretplaćuje i na koje objavljuje podatke.

Uključivanje ovako definiranog sustava za pokretanje sustava radi se u terminalu naredbom:

```
roslaunch bbox_depth bbox_depth.launch
```

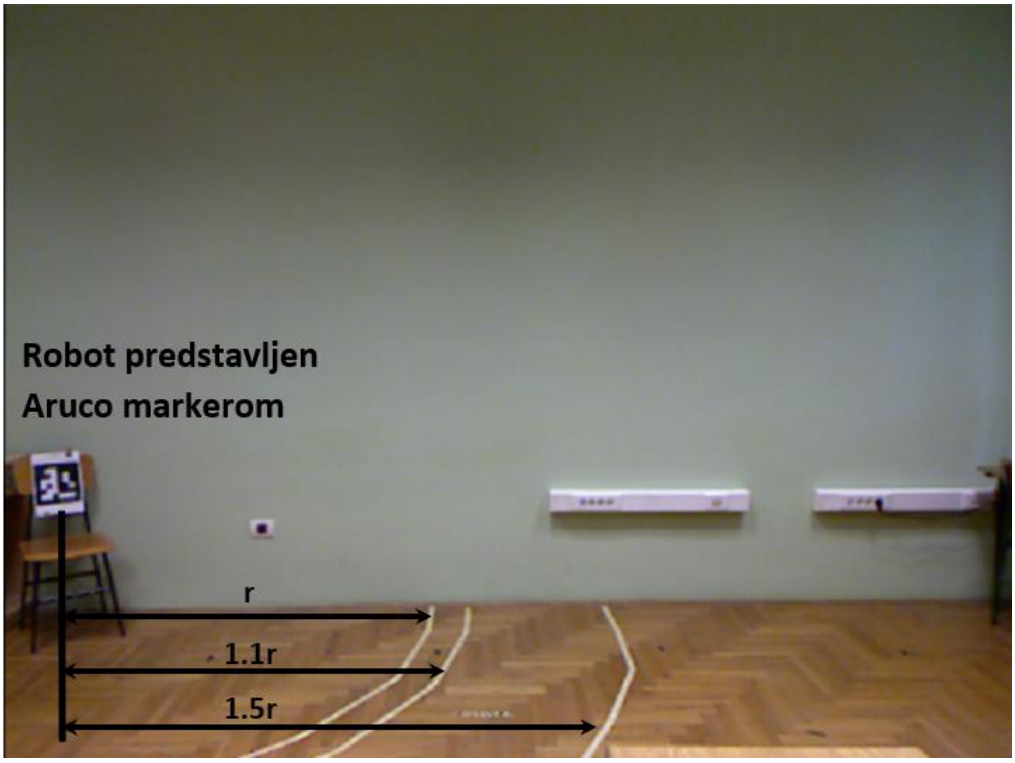
6. Eksperimentalna analiza

Testiranje izrađenog sigurnosnog sustava napravljeno je na skupu podataka sa zamišljenim (virtualnim) robotom. Na Internetu su dostupni razni skupovi podataka koji sadrže snimke kretanja ljudi, ali zbog specifičnosti izrađenog sigurnosnog sustava, dostupni skupovi podataka nisu primjenjivi. Stoga je napravljen skup podataka (engl. *dataset*) u sklopu HDR projekta za temeljito ispitivanje i testiranje različitih slučajeva i uvjeta prema kojima je izrađen sigurnosni sustav. Više o izrađenom skupu podataka i opisu pokusa slijedi u idućim poglavljima.

6.1. Skup podataka

Kreirani skup podataka nazvan je HDR skup podataka i čine ga različiti videozapisi snimljeni Kinect kamerom. Scene koje su snimljene simuliraju stvarne scene u industrijskim postrojenjima u kojima postoji interakcija ljudi s robotskim sustavima. Primjeri scena obuhvaćenih HDR skupom podataka su kretanje jednog čovjeka ili više ljudi različitim brzinama i smjerovima, trčanje čovjeka prema robotskom sustavu te situacije u kojima je čovjek djelomično zaklonjen od kamere objektom kojeg čovjek nosi. Takve situacije korisne su za testiranje algoritama prepoznavanja i praćenja ljudi, procjene brzine kretanja ljudi te procjene udaljenosti između čovjeka i robota. Videozapisi skupa podataka spremljeni su u obliku *rosbag* datoteka. Snimanje skupa podataka u *rosbag* formatu omogućuje snimanje podataka s različitih tema koje objavljuje Kinect kamera. U HDR skupu podataka snimljeni su podaci sa sljedećih tema: RGB slika, dubinska slika, oblak točaka te informacije o RGB kameri, dubinskom senzoru i infracrvenom projektoru.

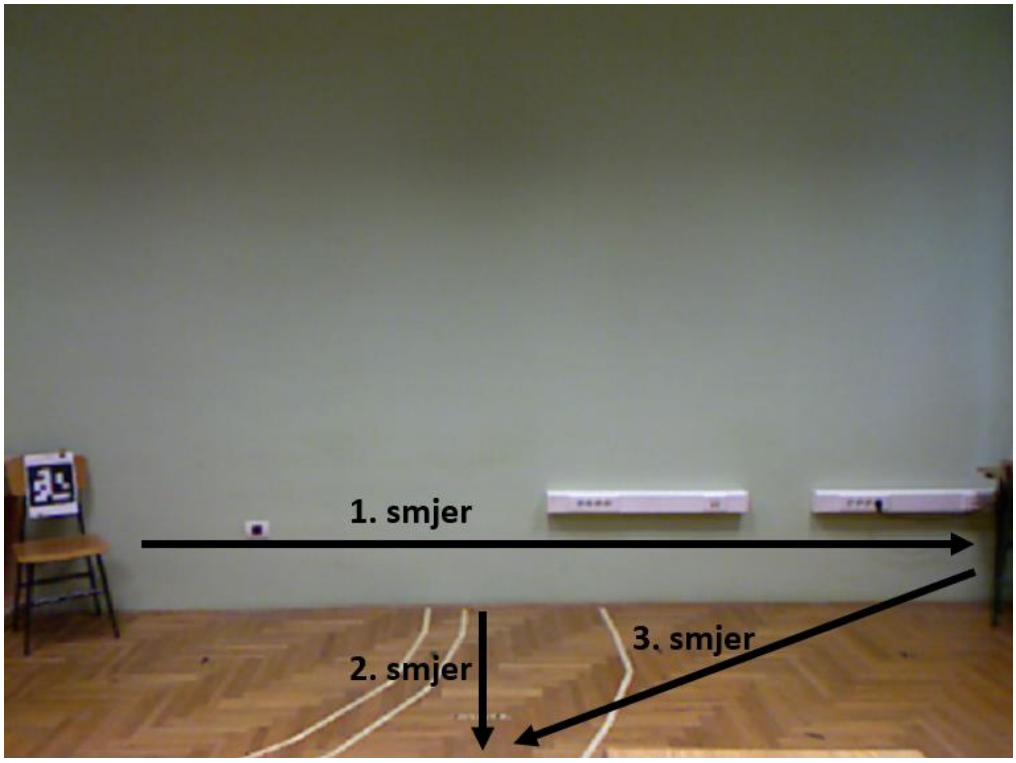
Skup podataka snimljen je na FERIT-u u zgradi na kampusu. Prostorija koja se koristila za snimanje označena je montažnim trakama i Aruco markerom kako bi se lakše evaluirali rezultati dobiveni sigurnosnim sustavom. Prikaz prostorije nalazi se na Slici 6.1.



Slika 6.1. Prostor za snimanje skupa podataka

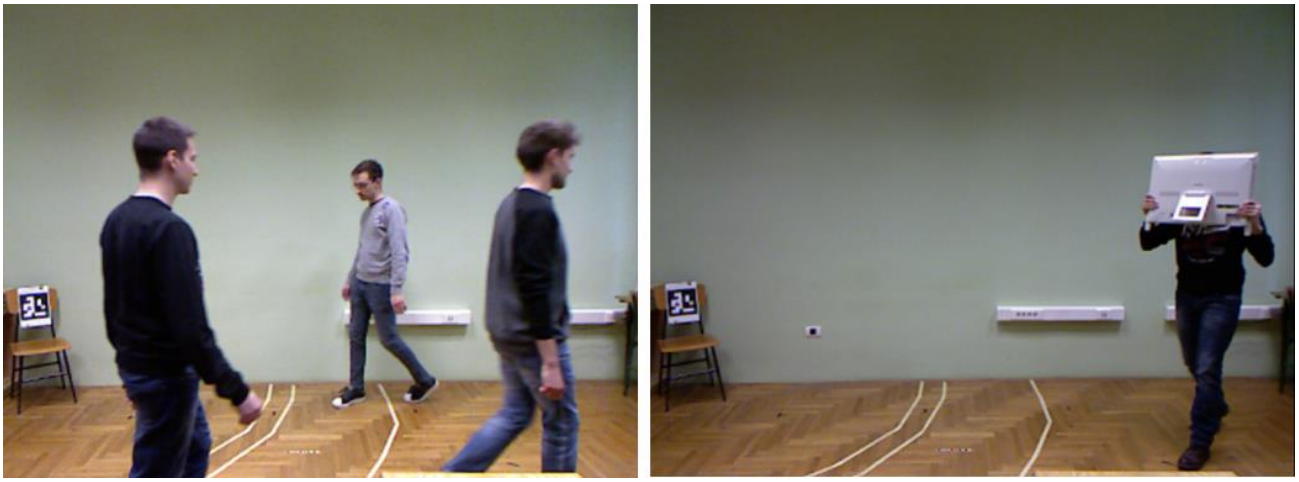
Robot je predstavljen Aruco markerom pomoću kojeg je moguće točno odrediti poziciju njegovog koordinatnog sustava u odnosu na koordinatni sustav kamere. Tri montažne trake koje se nalaze na podu predstavljaju polumjer radnog područja robota u različitim uvjetima opisanima u (2.1). Zadani polumjer radnog područja robota, *r*, predstavlja doseg ABB IRB 2400L robotskog manipulatora koji iznosi 1.81 metar. Druge dvije trake označavaju polumjer radnog područja oko robota uvećanog za 10% i 50%.

Snimke za testiranje brzine čovjeka raspoređene su na više kraćih snimaka. U svakoj snimci čovjek se kreće točno određenom brzinom, a za svaku brzinu napravljene su po tri različite snimke u kojima je različit smjer kretanja čovjeka. Smjerovi kretanja su paralelno, okomito i dijagonalno u odnosu na kameru. Smjerovi su prikazani na Slici 6.2.



Slika 6.2. Smjerovi kretanja čovjeka kod testiranja brzine

Skup snimaka za testiranje algoritama za detekciju i praćenje ljudi također se sastoji od više slučaja. Ovi slučajevi uključuju kretanje više ljudi istovremeno na slici i situacije u kojima je čovjek od kamere zaklonjen objektom kojeg nosi. Primjer ovakvih snimaka može se vidjeti na Slici 6.3.



Slika 6.3. Primjer snimke s više ljudi (lijevo); primjer snimke sa zaklonjenim čovjekom (desno)

Skup podataka sadrži označene referentne podatke koje čine informacije o broju ljudi na slici, brzini kretanja ljudi te udaljenosti između robota i čovjeka.

6.2. Opis snimaka

Snimke koje su snimljene za HDR skup podataka napravljene su specifično za testiranje metoda izrađenih u HDR projektu, stoga su pogodne za testiranje izrađenog sigurnosnog sustava. Testiranje sigurnosnog sustava podijeljeno je na tri vrste testova: test algoritma detekcije i praćenja ljudi, test estimacije brzine čovjeka i test zaustavljanja robota.

Uspješnost algoritama detekcije i praćenja ljudi testira se na četiri snimke:

- *Više ljudi* – snimka u kojoj se kreće više ljudi
- *Kutija 1* – snimka u kojoj je glava čovjeka zaklonjena od početka snimke
- *Kutija 2* i *Kutija 3* – snimke u kojima je glava čovjeka zaklonjena u određenim vremenskim intervalima snimke

Uspješnost estimacije brzine testira se na više snimaka u kojima se čovjek kreće sljedećim brzinama:

- $v \geq 2.0$ [m/s]
- $1.0 < v \leq 2.0$ [m/s]
- $0.1 < v \leq 1.0$ [m/s]
- $v \leq 0.1$ [m/s]

pri čemu je za svaku brzinu snimljeno kretanje u tri različita smjera (Slika 6.2.), osim za brzinu veću od 2.0 m/s. Za brzinu veću od 2.0 m/s snimljen je samo paralelan smjer kretanja u odnosu na kameru.

Zaustavljanje robota testira se na tri snimke u kojima čovjek ulazi i izlazi iz radnog područja robota.

6.3. Rezultati

U svakom testu izračunate su vrijednosti: *TP* (točno pozitivno; engl. *true positive*), *TN* (točno negativno; engl. *true negative*), *FP* (lažno pozitivno; engl. *false positive*) i *FN* (lažno negativno; engl. *false negative*).

- Ukupan broj primjera $= TP + TN + FP + FN$
- Broj točno klasificiranih primjera $= TP + TN$
- Broj pozitivnih primjera $= TP + FN$
- Broj negativnih primjera $= TN + FP$

Oznaka *T* (točno) znači da je algoritam točno klasificirao primjer, a oznaka *F* (lažno) da je algoritam netočno klasificirao primjer.

Pomoću tih vrijednosti mogu se izračunati mjere za vrednovanje sustava: odziv (engl. *recall*), preciznost (engl. *precission*) i točnost (*accuracy*). Odziv predstavlja udio točno klasificiranih primjera u skupu svih pozitivnih primjera, a računa se formulom: $R = \frac{TP}{TP+FN}$. Preciznost je udio točno klasificiranih primjera u skupu svih pozitivno klasificiranih primjera, a računa se formulom: $P = \frac{TP}{TP+FP}$. Točnost predstavlja udio točno klasificiranih primjera u skupu svih primjera, a računa se kao: $A = \frac{TP+TN}{TP+FP+TN+FN}$. Najznačajnija mjera je odziv jer se radi o sigurnosnom sustavu koji treba prepoznati čovjeka u svim točkama radnog prostora robota kako bi se vjerojatnost kolizije čovjeka i robota svela na minimum. Samo jedna propuštena detekcija čovjeka može prouzrokovati katastrofalne posljedice, čime se ugrožava sigurnost čovjeka. Sigurnosni sustav trebao bi biti 100% siguran za čovjeka da bi bio upotrebljiv u praksi.

Rezultati testiranja algoritama detekcije i praćenja ljudi prikazani su u Tablici 6.1.

Tablica 6.1. Rezultati detekcije i praćenja ljudi

Test	Odziv	Preciznost	Točnost
Više ljudi	0.9266	1.0	0.9323
Kutija 1	0.9333	1.0	0.9333
Kutija 2	0.9492	1.0	0.9492
Kutija 3	0.9074	1.0	0.9178

U slučaju testiranja detektora na snimci *Više ljudi* pokazalo se da najveći problem stvara međusobno zaklanjanje ljudi, odnosno kad se ljudi mimoilaze ispred kamere. Detektor u takvom slučaju detektira samo jednog čovjeka, pa se tu gubi na mjeri odziva i točnosti. Testiranjem detektora na snimkama gdje je čovjek zaklonjen pokazalo se da je detektor u mogućnosti detektirati čovjeka bez problema ako mu je vidljiva glava ili barem polovica tijela. Visoka vrijednost preciznosti od 100% pokazuje da algoritam neće nikada neki drugi objekt detektirati kao čovjeka. Iako je to dobro, bitnije je postići što veću vrijednost odziva, pošto odziv utječe na sigurnost čovjeka, a preciznost utječe na efikasnost sustava.

Rezultati testiranja estimacije brzine kretanja čovjeka prikazani su Tablicom 6.2.

Tablica 6.2. Rezultati estimacije brzine kretanja čovjeka

	Odziv	Preciznost	Točnost
$v \geq 2.0$ [m/s]			
Smjer 1	1.0	1.0	1.0
$1.0 \leq v < 2.0$ [m/s]			
Smjer 1	0.8333	0.8333	0.75
Smjer 2	1.0	1.0	1.0
Smjer 3	0.8333	1.0	0.875
$0.1 \leq v < 1.0$ [m/s]			
Smjer 1	0.8462	0.7333	0.7
Smjer 2	0.7143	0.9091	0.6875
Smjer 3	0.7	0.875	0.6364
$v < 0.1$ [m/s]			
Smjer 1	0.3628	1.0	0.3628
Smjer 2	0.3303	1.0	0.3303
Smjer 3	0.2785	1.0	0.2784

Estimacija brzine najbolje rezultate postiže kada se čovjek kreće brzinom većom od 2.0 m/s rezultatom od 100% za svaku mjeru vrednovanja. Relativno dobro radi i za brzine kretanja između 0.1 i 2.0 m/s. Najgori rezultati postižu se za brzinu manju od 0.1 m/s. Razlog tako lošeg rezultata može biti brže kretanje čovjeka na snimci od 0.1 m/s, zbog greške u snimci iz skupa podataka jer je teško hodati tako malom brzinom. Drugi razlog uzrokovan je radom algoritma koji poziciju čovjeka estimira sredinom graničnog okvira. Zbog šumova u detekciji, događa da čovjek stoji na mjestu, a granični okvir oko čovjeka konstantno mijenja svoju poziciju na slici. Time se stvara udaljenost između trenutne i prethodne pozicije središta graničnog okvira, pa algoritam pretpostavlja da se čovjek kreće.

Rezultate testova zaustavljanja robota prikazuje Tablica 6.3.

Tablica 6.3. Rezultati testova zaustavljanja robota

	Odziv	Preciznost	Točnost
Test 1	1.0	1.0	1.0
Test 2	0.9167	1.0	0.9412
Test 3	0.6667	1.0	0.7857

U tablici se može vidjeti da je prvi test postigao vrijednost odziva od 100%. Međutim, da bi sigurnosni sustav u potpunosti bio siguran za čovjeka, potrebno je na svim testovima postići takav rezultat. Nezaustavljanje robota na vrijeme može uzrokovati koliziju robota s čovjekom i dovesti do štetnih posljedica za čovjeka.

7. ZAKLJUČAK

Ovaj diplomski rad opisuje razvoj sigurnosnog sustava u ROS-u. Sigurnosni sustav pomoću RGB-D kamere nadgleda okolinu robotskog sustava u kojemu prepoznaje čovjeka te prilagođava rad robotskog manipulatora s obzirom na brzinu i smjer kretanja detektiranog čovjeka. Sigurnosni sustav zasnovan je na YOLO algoritmu za detekciju ljudi te SORT Deep algoritmu za praćenje ljudi.

U diplomskom radu predstavljeni su radovi koji se bave sličnom tematikom. Opisan je i shematski prikazan prijedlog rješenja sigurnosnog sustava te su opisani elementi korišteni za testiranje takvog sustava. Prikazana je izrada sustava te na kraju i postignuti rezultati izrađenog sustava.

Diplomski rad nastao je kao dio projekta *Humans Detected by Robots* (HDR), koji je pokrenut u suradnji FERIT-a i tvrtke Danieli Systec. U HDR projektu izrađene su tri metode koje koriste različite algoritme za detekciju i praćenje ljudi, ali su zasnovane na istim zahtjevima za sigurnosni sustav. U sklopu HDR projekta napravljen je skup podataka koji se koristio za testiranja rada sigurnosnog sustava razvijenog u ovom diplomskom radu.

Daljnjom nadogradnjom algoritama detekcije i praćenja ljudi mogu se poboljšati rezultati izrađenog sustava. Također, implementacijom dodatnih senzora u sustavu, npr. još jedna RGB-D kamera, mogu se smanjiti slučajevi zaklonjenosti čovjeka i dodatno poboljšati rezultati detekcije.

LITERATURA

[1] V. Šimundić, D. Mihelčić, D. Svirac, P. Đurović, and R. Cupec, "Safety System for Industrial Robots Based on Human Detection Using an RGB-D Camera,", MIPRO, 2021. prihvaćeno za objavljivanje

[2] A. Meguenani, V. Padois, J. Da Silva, A. Hoarau, and P. Bidaud, "Energy based control for safe human-robot physical interaction," in *2016 International Symposium on Experimental Robotics*, vol. 1, D. Kulić, Y. Nakamura, O. Khatib, and G. Venture, Eds. Springer, 2017. Pristupljeno: 10.03.2020. [Online]. Dostupno: https://hal.archives-ouvertes.fr/hal-01398790

[3] S. Sidagam and C.-Y. Tsai, "ROS-Based Human Detection and Tracking from a Wireless Controlled Mobile Robot Using Kinect," *Applied System Innovation*, vol. 2, p. 5, Jan. 2019, doi: 10.3390/asi2010005.

[4] S. S. Ghidary, Y. Nakata, T. Takamori, and M. Hattori, "Human detection and localization at indoor environment by home robot," in *Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics. "cybernetics evolving to systems, humans, organizations, and their complex interactions" (cat. no.0*, Oct. 2000, vol. 2, pp. 1360–1365 vol.2. doi: 10.1109/ICSMC.2000.886043.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *arXiv:1506.02640 [cs]*, May 2016, Pristupljeno: 14.03.2020. [Online]. Dostupno: http://arxiv.org/abs/1506.02640

[6] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple Online and Realtime Tracking," *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 3464–3468, Sep. 2016, doi: 10.1109/ICIP.2016.7533003.

[7] N. Wojke, A. Bewley, and D. Paulus, "Simple Online and Realtime Tracking with a Deep Association Metric," *arXiv:1703.07402 [cs]*, Mar. 2017, Pristupljeno: 20.03.2020. [Online]. Dostupno: http://arxiv.org/abs/1703.07402

[8] "ABB IRB 2400/L PRODUCT MANUAL Pdf Download | ManualsLib." https://www.manualslib.com/manual/1723921/Abb-Irb-2400-L.html

[9] M. Tenney, "Microsoft Kinect – Hardware," *CAST GeoSpatial Methods & Visualization*, Jul. 05, 2012. https://gmv.cast.uark.edu/scanning/hardware/microsoft-kinect-resourceshardware/ (Pristupljeno: 19.6.2020.).

[10] “ROS/Introduction - ROS Wiki.” http://wiki.ros.org/ROS/Introduction (Pristupljeno: 04.05.2020.).

[11] M. Meisel, “SUSTAV UPRAVLJANJA ROBOTSKIM MANIPULATOROM POMOĆU 3D KAMERE U ROS OKVIRU”, diplomski rad, FERIT, 2018.

SAŽETAK

Robotski sustavi općenito bi trebali biti sigurni za okolinu u kojoj rade. Ako se u toj okolini nalaze i ljudi, tada se javlja potreba za sigurnosnim sustavima koji će osigurati zaštitu čovjeka i spriječiti koliziju čovjeka i robota koji se kreće. U ovome radu prikazan je prijedlog rješenja jednog sigurnosnog sustava. Sigurnosni sustav sastoji se od RGB-D kamere i robotskog manipulatora. Pomoću RGB-D kamere detektiraju se ljudi koji se nalaze u okolini robotskog sustava te se estimira njihova pozicija, smjer i brzina kretanja. S obzirom na poziciju, smjer i brzinu kretanja čovjeka, upravlja se radom robotskom manipulatora. Sustav je implementiran u ROS-u te je integriran s Microsoft Kinect RGB-D kamerom i ABB IRB 2400L robotskim manipulatorom. Uspješnost razvijenog sigurnosnog sustava ispitana je na posebno izgrađenom skupu podataka kojim se ispituju zahtjevi prema kojima je izrađen sigurnosni sustav. Prikazani su rezultati testiranja te mogućnosti poboljšanja rezultata i samog sigurnosnog sustava.

Ključne riječi: detekcija ljudi, estimacija brzine ljudi, estimacija pozicije ljudi, robotski sustav, sigurnosni sustav

ABSTRACT

Human detection and tracking in the robotic environment using an RGB-D camera and YOLO algorithm.

In general, robotic systems should be safe for the environment they are working in. If humans are working in such environments, safety systems are needed which will ensure human protection and prevent collision between human and moving robot. In this paper, a solution proposal of a safety system is represented. The safety system includes an RGB-D camera and a robot manipulator. With the RGB-D camera, the safety system can detect humans located near the robot manipulator and estimate their position, walking speed and direction. With respect to the human position, speed and direction, working mode of the robot manipulator is regulated. The safety system is implemented in ROS and integrated with Microsoft Kinect RGB-D camera and ABB IRB 2400L robot manipulator. Performance of the developed safety system is tested on a dataset designed for testing the conditions according to which the safety system is created. The results of the developed safety system are shown with adduced possibilities for improvement of both the results and the safety system.

Key words: human detection, human speed estimation, human position estimation, robotic system, safety system

ŽIVOTOPIS

Damian Svirac rođen je 1.9.1997. godine u Slavonskom Brodu. Pohađao je osnovnu školu „Đuro Pilar“ u Slavonskom Brodu nakon koje upisuje „Prirodoslovno-matematičku gimnaziju Matija Mesić“ također u Slavonskom Brodu. Uz školske obveze aktivno je sudjelovao u Kulturno umjetničkom društvu Luka Lukić Brodski Varoš, te nekoliko godina svirao u Brodskom tamburaškom orkestru. Nakon završetka srednje škole, 2016. godine upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija u Osijeku, smjer Računarstvo. Tijekom 2019. godine odrađivao je stručnu praksu u Danieli Systecu., gdje je i pisao završni rad. U 2019. godini završava preddiplomski studij Računarstva i upisuje diplomski studij Računarstva, izborni blok Robotika i umjetna inteligencija. Od 1.3.2020. sudjeluje na projektu *Humans Detected by Robots* (HDR) koji je prihvaćen na MIPRO konferenciji te za koji dobiva rektorovu nagradu.