Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Jernej Poles

Implementacija enostavnega mobilnega robota za varno premikanje po omejenem prostoru

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: izr. prof. dr. Danijel Skočaj

Ljubljana, 2016

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Da bi lahko roboti sobivali s človekom, se morajo temu primerno tudi obnašati. Zato postaja socialna robotika vse bolj pomembna veja robotike. V diplomski nalogi implementirajte funkcionalnosti enostavnega avtonomnega mobilnega robota z zametki socialnega obnašanja. Mobilni robot naj se tako prosto giblje po območju, ki ga na tleh omejimo z dvobarvno črto. Izogiba naj se vsem oviram, tako statičnim kot dinamičnim. Poleg tega naj detektira obraze, se vsakemu od njih približa in ga ogovori. Razviti sistem ter njegove funkcionalnosti tudi ustrezno ovrednotite.

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani Jernej Poles,

z vpisno številko 63130367,

sem avtor diplomskega dela z naslovom:

Implementacija enostavnega mobilnega robota za varno premikanje po omejenem prostoru

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom

  izr. prof. dr. Danijel Skočaj

- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela

- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 12.09.2016 Podpis avtorja:

Kazalo

Povzetek

Naslov: Implementacija enostavnega mobilnega robota za varno premikanje po omejenem prostoru

Cilj diplomske naloge je implementacija enostavnega mobilnega robota za varno premikanje po omejenem prostoru. Ta implementirani sistem bo v prostoru tudi zaznaval obraze in izvajal pozdrav s sintetičnim govorom. Sistem je bil implementiran z uporabo računalniškega ogrodja ROS in programskega jezika Python. V prvem delu bomo predstavili mobilno enoto in računalniško ogrodje ROS, v osrednjem delu pa metode in evalvacijo implementiranega sistema. Z evalvacijo bomo preverili delovanje metod in celotnega sistema. V zaključku bomo predstavili ugotovitve evalvacij in morebitne možnosti za nadgradnje sistema.

Ključne besede: računalniško ogrodje ROS, sistem, metoda, mobilna robotska enota.

Abstract

Title: Implementation of a simple mobile robot that safely moves in a confined area

The aim of the thesis is the implementation of a simple mobile robotic unit that can safely move around in a confined area. This system will also perceive faces in the space and greet them with synthetic speech. The system has been implemented using computer framework ROS and the Python programming language. In the first part we will present the mobile unit and the computer framework ROS, and in the middle part the methods and evaluation of the implemented system. The evaluation will be used to check the functioning of the operating methods and of the entire system. In conclusion, we will present the findings of the evaluations and possible options for the future system upgrades.

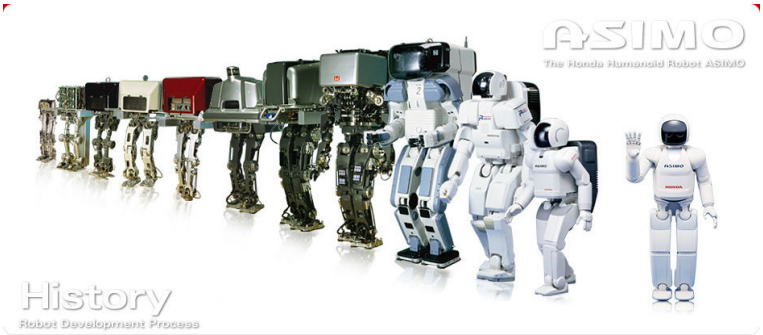
Keywords: computer framework ROS, system, method, mobile robotic unit.

Poglavje 1

Uvod

1.1 Motivacija

Robotika [12] je eno izmed področij strojništva, elektrotehnike in računalništva, ki se ukvarja z načrtovanjem, konstrukcijo in računalniškimi sistemi za nadzor, senzorične povratne informacije in procesiranje informacij. Te tehnologije se ukvarjajo z avtomatskimi stroji (na kratko roboti), ki lahko nadomestijo mesto človeka v nevarnih okoljih ali v proizvodnih procesih ali spominjajo na človeka v videzu, obnašanju in spoznavanju. Veliko od današnjih robotov se zgleduje po naravi, kar je prispevalo k področju bio-robotike (ang. Bio-inspired robotics).



Slika 1.1: Robotska zgodovina Asimo ([18]).

Koncept o načrtovanju naprav, ki lahko deluje samostojno, sega že iz klasične antike, vendar se pred 20. stoletjem funkcionalnost in možnost uporabe ni bistveno povečala [6]. Skozi zgodovino je bilo pogosto domnevano, da bodo roboti nekega dne sposobni oponašati človeško obnašanje in opravljati naloge na človeški način. Primer zgodovine razvoja robota Asimo je razvidno na sliki 1.1.

Robotika je veja več področij z različnimi nameni, nas pa najbolj zanima aspekt računalniškega sistema za robota. Želeli smo implementirati sistem za robota, ki bi imel sposobnost interakcije z ljudmi.

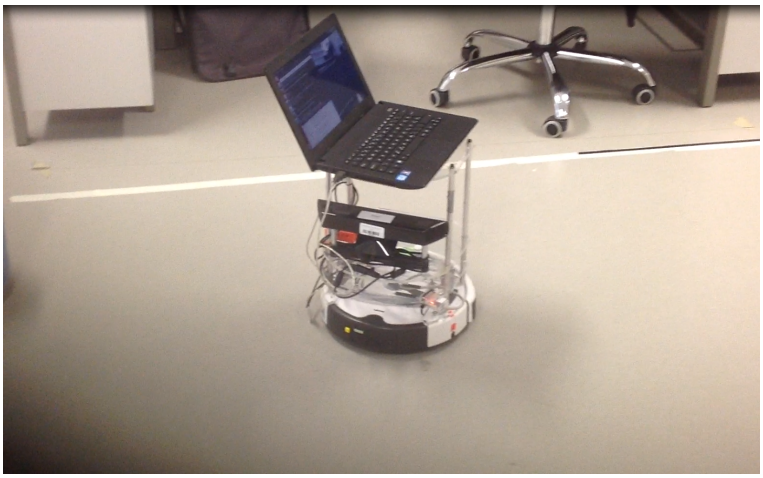
1.2 Cilj

Zastavljen problem diplomske naloge je implementacija avtonomnega mobilnega robota s funkcionalnostjo varnega premikanja in zaznavanja obrazov. V omejenem prostoru naj bi se naključno premikal, ne da bi si izrisal načrt prostora. Omejeni prostor, ki ga ne sme zapustiti, predstavlja črta na tleh. Ta črta je črno-bele barve. Predmete v prostoru naj bi robot zaznal kot ovire ter se jim izognil. V primeru trka naj bi se ustavil in se začel premikati vzvratno ter nadaljeval pot v drugo smer. Sistem mora tudi pravilno zaznati previse in se jim pravilno izogniti. V primeru, da bi zaznal človeški obraz, naj se mu skuša približati in ga pozdraviti. Ko pozdravi obraz, naj bi se premaknil vzvratno in obrnil v drugo stran ter nadaljeval s premikanjem v prostoru. Primer uporabe se ponazarja na sliki 1.2.

Zahtevane sposobnosti, ki jih potrebujemo za robotski sistem so:

- Zaznavanje in izogibanje oviram.
- Zaznavanje in odziv na trk.
- Zaznavanje in odziv na previs.

- Zaznavanje in odziv na črno-bele črte.
- Zaznavanje obrazov.
- Premikanje do obrazov.
- Pozdrav obraza s sintetičnim govorom.



Slika 1.2: Izvajanje sistema na robotski enoti.

1.3 Metodologija

Za implementacijo sistema smo zasnovali sistem v računalniškem ogrodju *ROS* [8] in ga implementirali v programskem jeziku *Python* [14]. Za upravljanje z računalniškim vidom smo uporabili metode iz knjižnice *OpenCV* [7]. Sistem smo izvajali ter testirali z uporabo robotske enote *Turtelbot* [22] in simulatorjem *Gazebo* [17].

V sistem smo implementirali sledeče metode, ki pomagajo pri reševanju problemov.
Metoda zaznave in odziva na črte, ki lahko razlikuje med črto črno-bele

barve in navadno črto. S to metodo sistem zna ločevati med črtami in se izogibati le specifičnim črtam.
Metoda zaznave in odziva na obraz, ki omogoča zaznavo obrazov ter pravilno premikanje do njih. S tem se robotska enota premika do njih in sproži sintetični govor.
Metoda za odziv na ovire, ki zaznava razdaljo do ovir. S to metodo se robotska enota izogiba oviram, ki so preblizu.
Metode za odziv na trk in previs, ki zazna trk ali previs. S to metodo se robotska enota pri trku ali previsu ustavi in začne premikati v nasprotno smer.

Končani sistem smo evalvirali in preverili pravilnost delovanja.

1.4 Zgradba diplomske naloge

Diplomska naloga je sestavljena iz petih poglavij.
Prvo poglavje je uvod.
Drugo poglavje je poglavje strojne in programske opreme in vsebuje podpoglavja, ki opišejo izbrano programsko in strojno opremo, ki jo uporabljamo za načrtovanje in testiranje sistema.
Tretje poglavje je poglavje implementiranih metod za robotski sistem in vsebuje podpoglavja, ki opišejo delovanje: celotnega sistema, metod zaznavanja in odziva na obraz, metod zaznavanja in odziva na črto, metod zaznavanja in odziva na oviro in metod odziva na trk ali previs.
Četrto poglavje je poglavje evalvacije ustvarjenega sistema in vsebuje podpoglavja, ki opišejo evalvacijo: metode zaznavanja in odziva na obraz, metode zaznavanja in odziva na črto in delovanje celotnega sistema na testnem poligonu.
Peto poglavje je zaključek diplomske naloge in vsebuje zaključni sklep ter predloge za morebitno izboljšanje sistema.

Poglavje 2

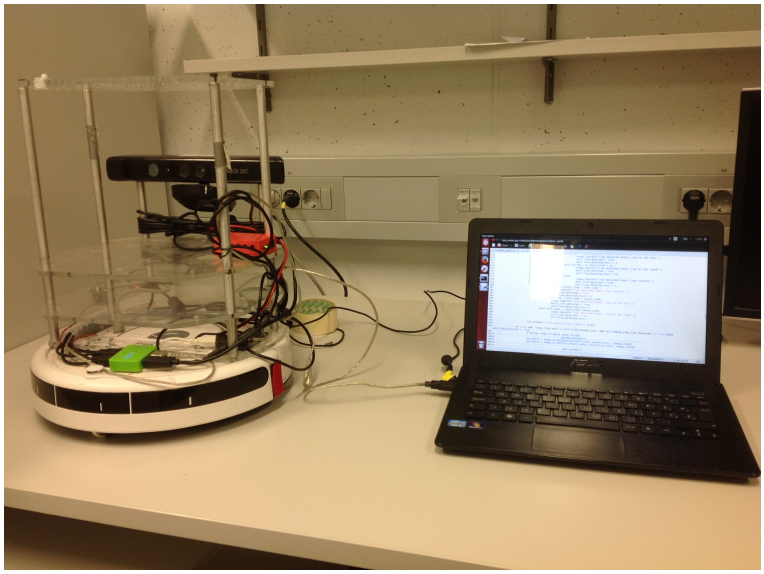
Strojna in programska oprema

V tem poglavju bomo predstavili strojno in programsko opremo, ki smo jo uporabljali za načrtovanje ter preizkušanje implementiranega sistema. V prvem delu poglavja bomo opisali robotsko enoto, ki je bila potrebna za testiranje implementiranega sistema. V drugem delu poglavja bomo opisali, katero programsko opremo smo uporabljali za načrtovanje sistema.

2.1 Strojna oprema

Za strojno opremo smo imeli na voljo dve računalniški delovni postaji in eno robotsko enoto:

- Robotsko enoto *Turtelbot*, ki je vidna na sliki 2.1.
- Računalniško delovno postajo, na kateri smo načrtovali ter implementirali celotni sistem in testirali sistem na simulatorju *Gazebo*.
- Prenosni računalnik; z njim smo preizkusili sistem na robotski enoti Turtelbot.



Slika 2.1: Robotska enota *Turtelbot* in prenosni računalnik.

2.1.1 Robotska enota

Robotska enota je sestavljena iz dveh delov, ki jih bomo na kratko opisali.

- **iRobot Roomba 530** robot, ki je zadolžen za premikanje.
- **Barvno globinska kamera Kinect 360**, s katero zaznavamo sliko.

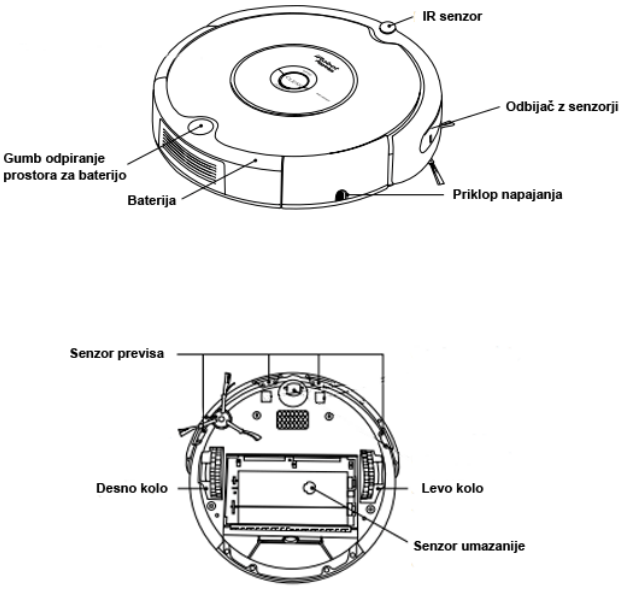
Roomba [10] je serija avtonomnih sesalnih robotov za čiščenje. Proizvajalec je *iRobot*. Prva serija je izšla septembra 2002. Robot, ki ga uporabljamo v našem primeru, je izšel avgusta 2007. Ta *Roomba* je vidna na sliki 2.2. *Roomba* je opremljena s preprostimi senzorji: odbijač za zaznavanje trkov, IR senzor na spodnjem delu roombe za zaznavanje previsov, senzor za zaznavanje umazanije, senzor za navidezni zid. Ti so vidni na sliki 2.3. Uporablja dve neodvisni kolesi, ki omogočata obračanje za 360° na mestu.

Roomba je predelana, tako da ima namesto sesalne enote baterijo. Ta dodatek podaljša delovno dobo *Roombe*. Senzorja *Roombe*, ki jih uporabljamo za sistem, sta senzor odbijača za kolizije ter IR senzor za zaznavane previsov.

Slika 2.2: Roomba 530 ([11]).

Odbijač se nahaja na sprednji strani v smeri premikanja in zazna dotik ob trku z oviro. Odbijač lahko trk zazna na treh straneh: levo, desno in spredaj. IR senzor za previs se nahaja v sprednjem spodnjem delu pod odbijačem, ki zazna previs na talni površini. Strani zaznave previsa z IR senzorjem: previsi na levi, desni, levi spredaj in desni spredaj.
Uporabljamo tudi funkcijo premikanja in obračanja za 360°. To funkcijo robotska enota uporablja za premikanje naprej in vzvratno ter obračanje na mestu.

Barvno globinska kamera Kinect 360 [9] vidna na sliki 2.4 je zasnovana po vzoru spletne kamere. V Kinectu so vgrajeni RGB kamera, senzorji gibanja in senzorji globine ter mikrofoni za zaznavo zvoka. Zasnoval ga je *Microsoft* za konzolo *Xbox 360* in kasneje operacijski sistem *Windows*. Senzor globine sestoji iz IR laserskega projektorja in monochrome *CMOS* senzorja, ki dobi podatke za ustvarjanje virtualnega 3D prostora v katerikoli osvetlitvi. Barvno globinsko kamero *Kinect* uporabljamo za sprejem navadne RGB slike ločljivosti 640x480 slikovnih pik in razdaljo do predmetov, ki jo dobi z globinskim senzorjem za več višin.



Slika 2.3: Lokacije senzorjev na Roombi. (povzeto po [21]).

Slika 2.4: Barvno globinska kamera Kinect 360 ([9]).

2.2 Programska oprema

Za programsko opremo smo izbrali orodja in sisteme, ki podpirajo računalniško ogrodje *ROS*. S temi bomo implementirali, načrtovali in upravljali sistem. Vsako uporabljeno programsko opremo bomo na kratko opisali, razen *ROS-a*, ki ga bomo podrobneje opisali ter pokazali njegove glavne elemente, ki smo jih uporabili.

Ubuntu 14.04 [15] je prosto dostopen operacijski sistem z veliko podpore za znanstvene programe in zelo obširno dokumentacijo.

Python [14] je visokonivojski, široko uporaben, splošno namenski, interpretiran, dinamičen programski jezik. Ima veliko podprtih knjižnic in dokumentacij. Sintaksa omogoča implementacijo kode z manj vrsticami kot kateri drugi jeziki, npr. *C++*. Ta lastnost nam zmanjša čas in morebitne težave, ko moramo popravljati sistem in dodajati nove metode. Podpira tudi računalniško ogrodje *ROS*, čeprav ima manj dokumentacije kot *C++*.

OpenCV [7] je odprtokodna knjižnica, ki je v glavnem namenjena za delovanje računalniškega vida v realnem času. Je pomemben del našega sistema, ker jo potrebujeta dve metodi. Ima veliko dokumentacije ter podpira tudi programski jezik *Python*.

Numpy [16] je knjižnica, ki razširi programski jezik *Python*, tako da omogoča delovanje z večjimi in večdimenzionalnimi matrikami. Potreben je za učinkovito in lažje delovanje z računalniškim vidom.

2.2.1 ROS

ROS [8] je računalniško ogrodje, ki ponuja veliko knjižnic in orodij, z namenom, da bi pomagalo programskemu razvijalcu ustvariti robotske sisteme. Ponuja abstrakcije strojne opreme, gonilnike naprav, knjižnice, vizualizacije,

prenašanje sporočil, upravljanje paketov in več. Ima veliko dokumentacij in je dobro podprt. Verzija *ROS-a*, ki smo jo uporabili, je *Indigo*. Za delovanje z *ROS* delovnim prostorom(ang. *Workspace*) smo uporabljali *Catkin*, ki omogoča ustvarjanje in upravljanje *ROS* paketov. Uporabljali in delali smo z naslednjimi deli računalniškega ogrodja *ROS*.

Catkin je uraden sistemski graditelj *ROS-a* in naslednjik prejšnjega *ROS*-ovega sistemskega graditelja. *Catkin* je zadolžen z ustvaritvijo cilja iz surove programske kode, ki omogoči njeno uporabo končnemu uporabniku. Ti cilji so lahko v obliki knjižnice, izvedljivega programa, generirane skripte, izvoženih vmesnikov in česarkoli drugega, ki ni statična koda.

ROS paket (ang. *Packages*) se uporablja za organizacijo ROS programov. Ti lahko vsebujejo *ROS* vozlišča, samostojne knjižnice, nabor podatkov in razne uporabne module. Za sintetični govor smo uporabili naslednji paket:

- **Sound_play** [13] je del zbirke paketov *Audio_common*. Ponudi funkcionalnost, ki pretvori besedilo v sintetični govor. Uporabljamo ga za pozdravljanje obrazov.

ROS vozlišče (ang. *Node*) je proces, ki izvede računanje. Vozlišča medsebojno komunicirajo z uporabo sporočil. Ta vozlišča so namenjena za operacije v manjšem obsegu programa (npr. vsak senzor uporablja svojo vozlišče).

ROS tema (ang. *Topic*) je namenjena za enostranski tok komunikacij, med katerimi si lahko vozlišča izmenjujejo sporočila. Na temo se lahko vozlišče prijavi in dobi sporočilo ali objavi in pošilja sporočila temi. Teme, ki smo jih uporabljali:

- **mobile_base/sensors/core** je tema, ki pošilja status robotske enote in vsebuje status odbijača in IR senzorja za previs.
- **cmd_vel_mux/input/navi** je tema, na katero se objavi sporočila za premikanje robotske enote.

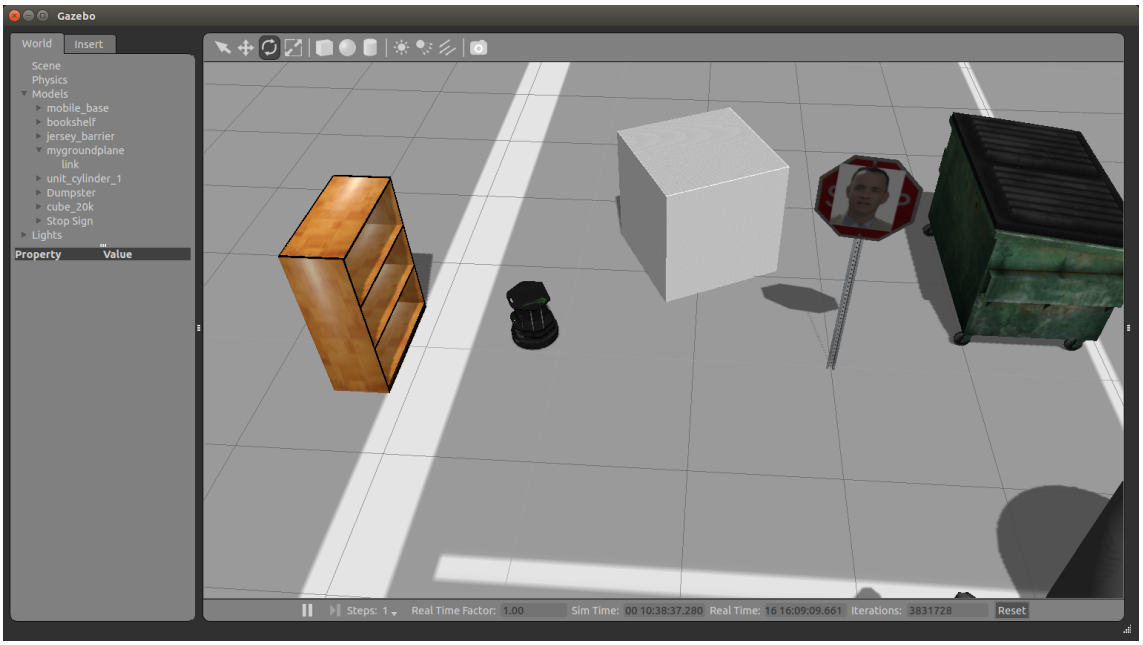
- **scan** je tema, ki pošilja vse razdalje v prostoru od robotske enote. Tema preverja IR senzor, ki je v našem primeru simuliran z *Kinekt* kamero.
- **camera/rgb/image_raw** je tema, ki pošilja navadno sliko RGB formata.

ROS Sporočilo (ang. *Message*) je potrebno za komunikacijo med *ROS* temami in *ROS* vozlišči. Ta sporočila so preprosta podatkovna struktura, ki vsebuje različne podatke. V našem primeru uporabljamo naslednja sporočila:

- **SoundRequest** vsebuje podatke besedila, ki ga nameravamo povedati s sintetičnim govorom.
- **Twist** vsebuje podatke o hitrosti in smeri premikanja.
- **LaserScan** vsebuje podatke o razdaljah v prostoru.
- **Image** vsebuje podatke o RGB sliki.
- **TurtlebotSensorState** vsebuje podatke o stanju *Roombe*.

2.2.2 Gazebo

Gazebo [17] je simulator za robotske sisteme, ki omogoča simulacijo delovanja robotskih sistemov v različnih okoljih pod različnimi pogoji. Tako lahko hitro testiramo nove algoritme, načrtujemo robote in izvajamo regresijske teste z uporabo realnih scenarijev. *Gazebo* nam omogoča natančno in efektivno simulacijo populacij robotov v kompleksnih zunanjih in notranjih prostorih. Vsebuje robustni fizikalni računalniški pogon, visokonivojsko grafiko ter preprost grafični vmesnik.
Uporabljali smo ga za simulacijo robotske enote *Turtelbot* v preprostem okolju z nekaj predmeti. To preprosto okolje je vsebovalo tudi simulirano črto in obraz. Ta uporaba je razvidna s slike 2.5.



Slika 2.5: Gazebo virtualno okolje.

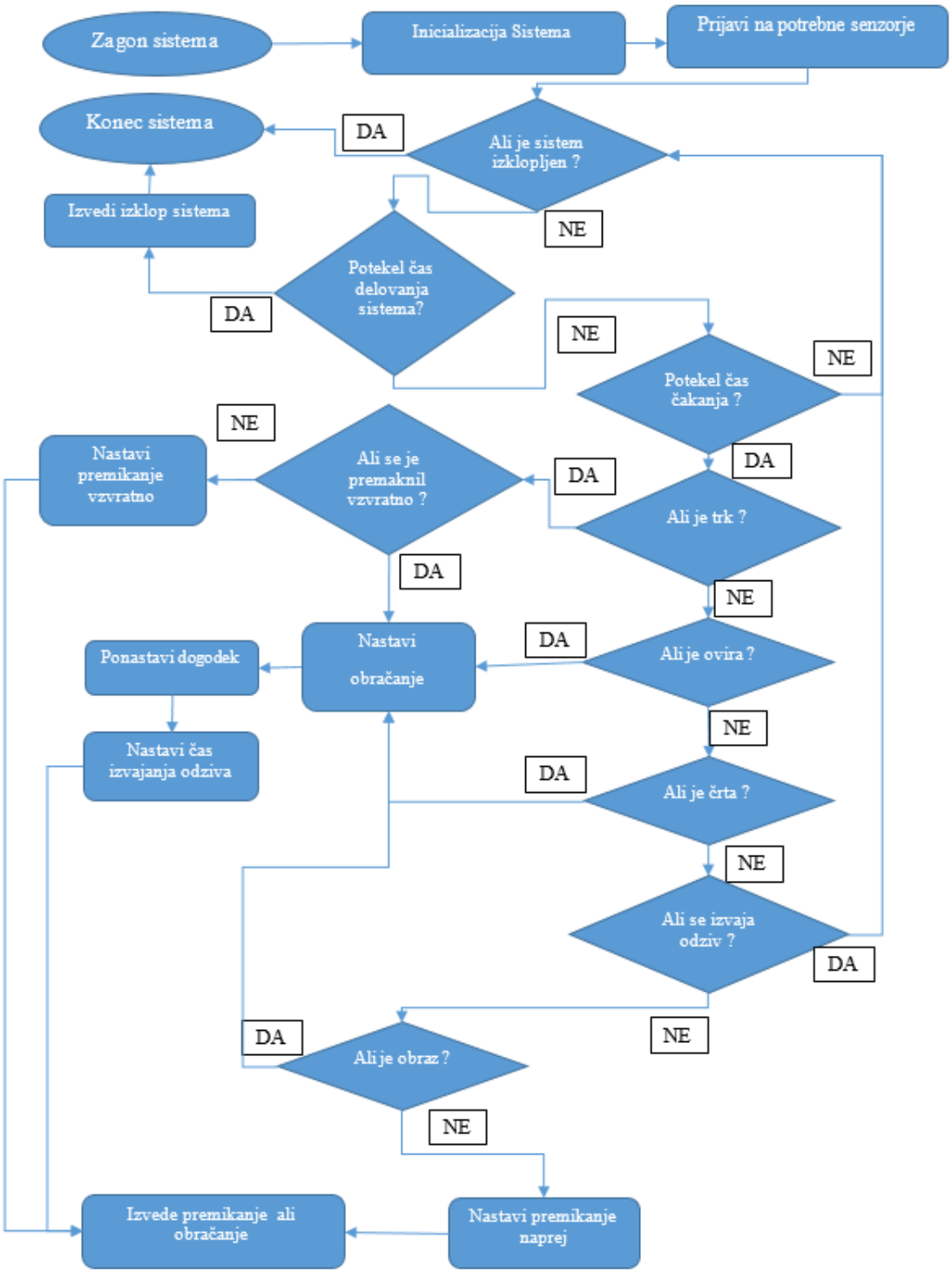
Poglavje 3

Implementirane metode

V tem poglavju bomo predstavili, kako smo zadoščali zahtevanemu cilju diplomske naloge. Opisali bomo delovanje celotnega sistema in metodo za premikanje in obračanje, metodo zaznavanja in odziva na obraz, ki je razdeljena na dva dela, metodo zaznavanja in odziva na črto, metodo odziva na ovire in metodo odziva na trke ali previse.

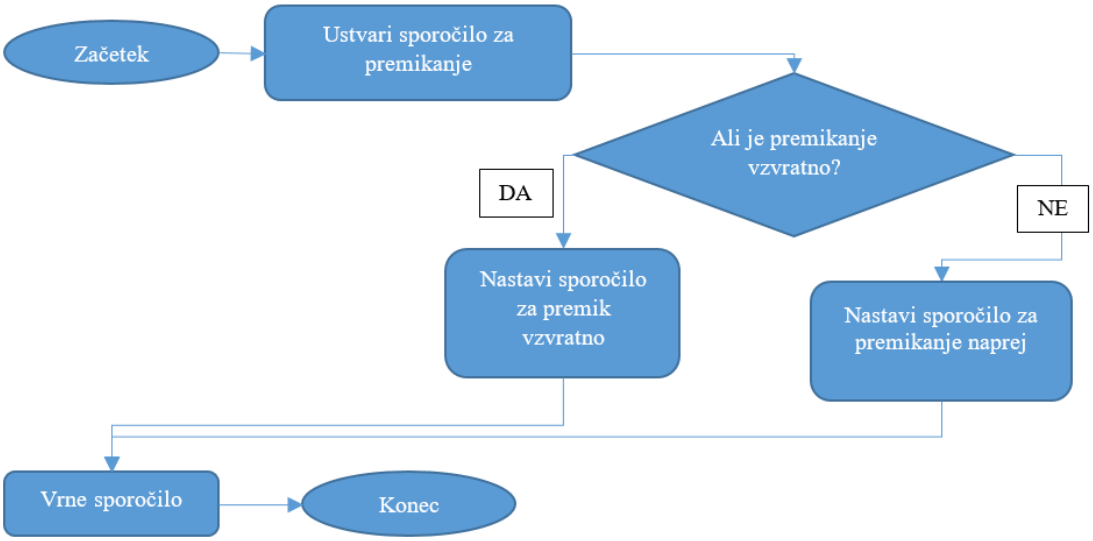
3.1 Delovanje celotnega sistema

Sistem ob zagonu počaka določen čas, preden se začne premikati. Sistem in metode sprejemajo sporočila od senzorjev, ki vsebujejo podatke glede na tip senzorja. Te podatke celotni sistem preveri in sproži temu primerne odzive ter jih izvrši. Premikati se začne naravnost. Kadar robotska enota pride pred črto, ki je črno-bele barve, bo metoda za zaznavanje in odziv na črto sprožila odziv, po katerem bo sistem začel z izvajanjem izogibanja črte. Ko robotska enota pride do ovire, ki je preblizu, bo metoda za odziv na oviro sprožila odziv, po katerem bo sistem začel z izvajanjem izogibanja ovir. Če se robotska enota oviri ne izogne in pride do trka, bo metoda za odziv na trk sprožila odziv, po kateri bo sistem prenehal premik naprej in začel izvajati premik vzvratno ter se bo obrnil v drugo smer.



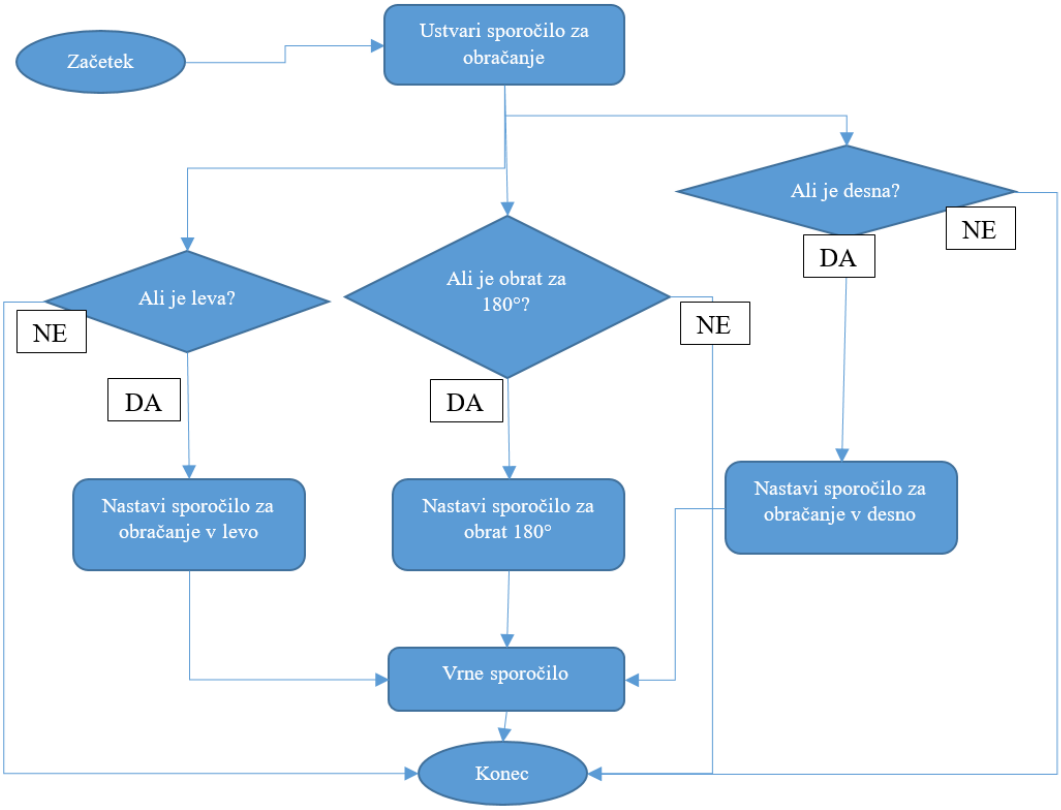
Slika 3.1: Diagram poteka celotnega sistema.

Če se robotska enota premika čez previs, bo metoda za odziv na previs sprožila odziv, po kateri bo sistem prenehal premik naprej in se začel premikati vzvratno ter se bo obrnil v drugo smer. Ob morebitnem obrazu pred robotsko enoto bo metoda za zaznavanje in odziv na obraz sprožila odziv, po kateremu se bo sistem skušal približati obrazu ter ga pozdraviti s sintetičnim govorom ob pravi razdalji. Po določenem času delovanja se bo sistem samodejno izključil in prekinil vse trenutne aktivnosti. Celotno delovanje sistema v algoritmični obliki je razvidno s slike 3.1.



Slika 3.2: Diagram poteka metode premikanja.

Preden sistem izvede premikanje, je treba sporočilo za premikanje, ki vsebuje zahtevano hitrost ter morebitno smer. To izvede metoda za premikanje ali obračaje, kateri ustvarita in nastavita sporočilo glede zahtevani način premikanja. Za premikanje naprej se nastavi le pozitivna hitrost, za premikanje vzvratno pa negativna. Pri obračanju se določi stran obračanja med levim, desnim in obratom za 180°, ki se zgodijo na mestu. Delovanje obeh metod se lepo vidi na slikah 3.2 in 3.3.

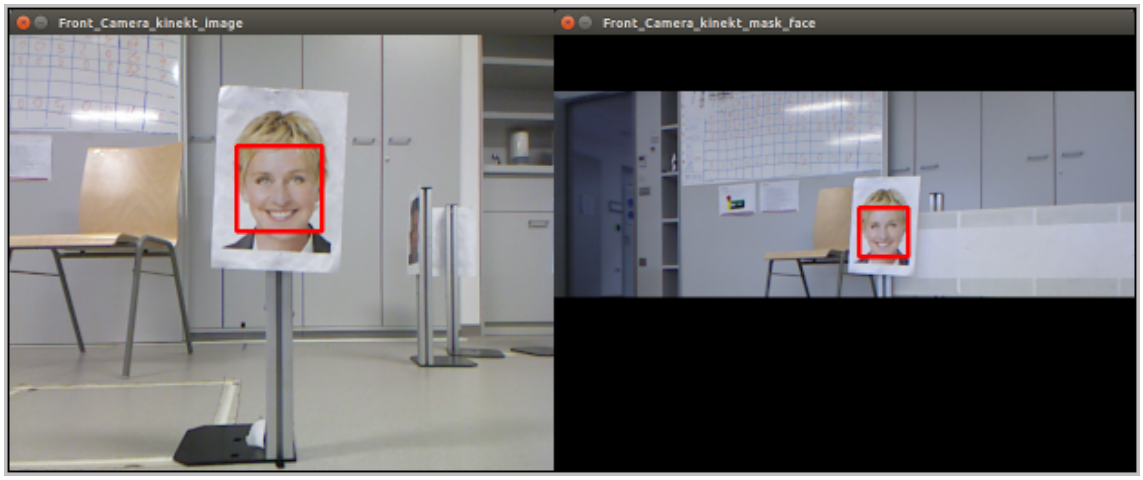


Slika 3.3: Diagram poteka metode obračanja.

3.2 Metoda zaznavanj in odziv na obraz

Metoda za odziv najprej sprejme sporočilo, ki vsebuje RGB sliko. To sliko nato maskira, kar zmanjša število možnih napak pri zaznavanju, saj predvidevamo, da se obrazi ne nahajajo na tleh. To je razvidno iz slike 3.4. Na sliki izvede metodo za zaznavanje obrazov. Ta metoda s pomočjo algoritma *AdaBoost* [3] najde vse možne obraze na sliki. Preden začne z odzivom, še počaka, da se je iztekel čas preverjanja, če je res obraz in ne napačno za-

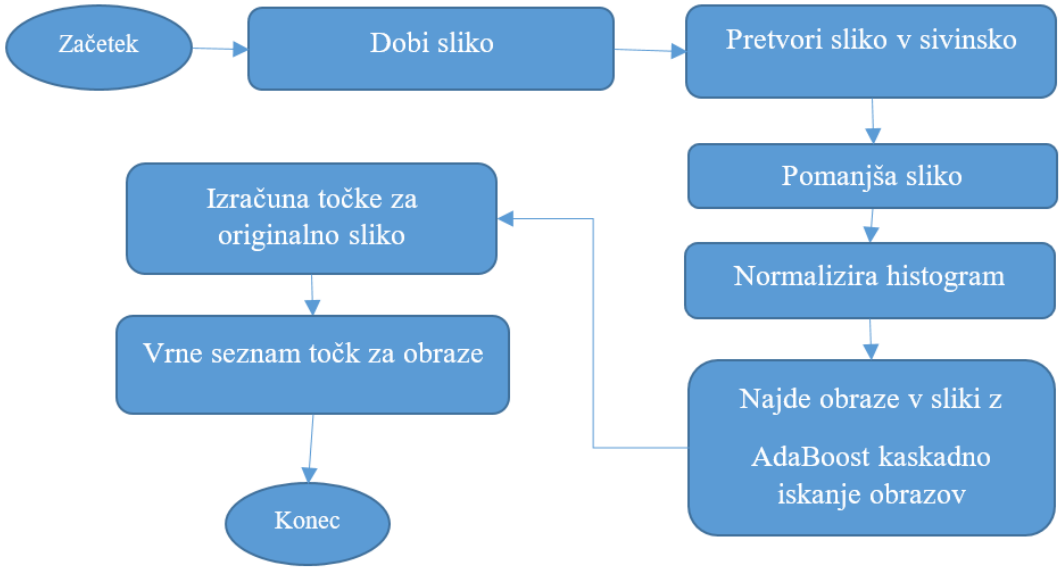
znan predmet. Po tem, glede na pozicijo obraza, sproži odziv za obračanje proti obrazu. Ko je zaznani obraz dovolj blizu, sproži pozdrav obraza. Ob zaključku pozdrava sproži lažen odziv za trk, kar povzroči, da se robotska enota ustavi in pelje vzvratno ter obrne v nasprotno smer od obraza. To delovanje je razvidno s slike 3.8.



Slika 3.4: Prikaz razlike med maskirano sliko desno in nemaskirano sliko levo.

Metoda za zaznavanje obraza dobi sliko ter jo pomanjša, kar pohitri iskanje. Izenači histogram slike, da normalizira svetlost, kar izboljša kontrast slike in poudari obraz. Potem se izvede glavni del metode iskanja obrazov v sliki z algoritmom *AdaBoost* [3]. V našem primeru uporabljamo že naučeni klasifikator z značilnicami *Haar*. Ta klasifikator je določen v zagonski datoteki. S klasifikatorjem začnemo iskati obraz v sliki. Iskanje poteka postopoma z večjimi šibkimi klasifikatorji. Algoritem najde obraz, ko se vse stopnje pravilno ujemajo na določeni regiji v sliki. Iskanje se izvede večkrat na sliki in tudi za različne velikosti obraza. Po končanem iskanju metoda dobi seznam obrazov. Delovanje metode je razvidno s slike 3.5.

AdaBoost [3] je algoritem strojnega učenja. Ta deluje tako, da izbira in kombinira več **šibkih klasifikatorjev**, da ustvari en bolj natančen in **močen klasifikator**, kar je razvidno na sliki 3.6.

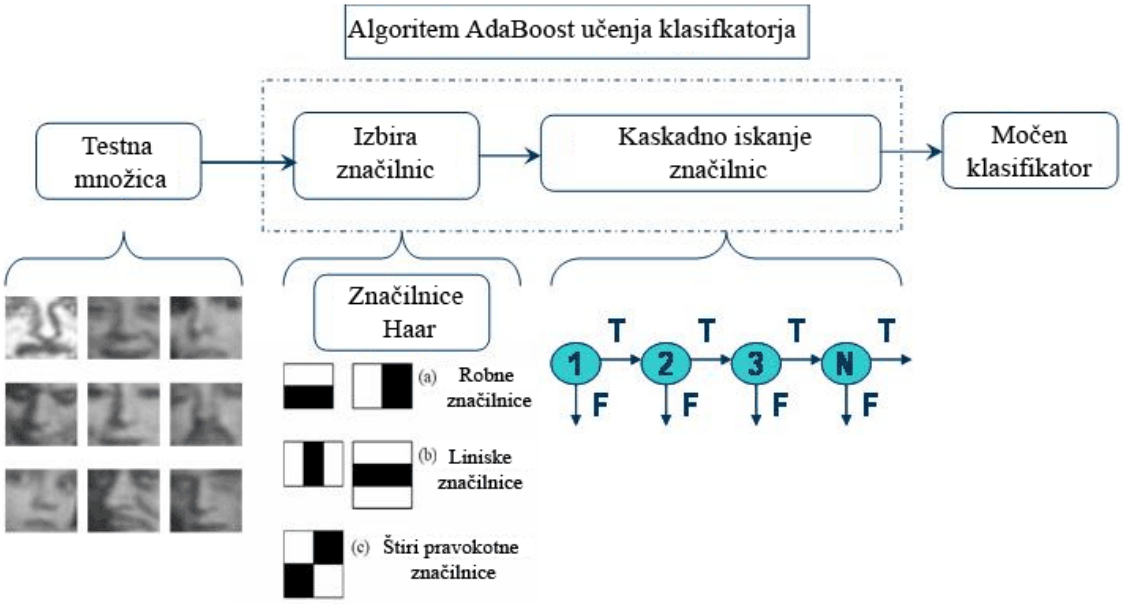


Slika 3.5: Diagram poteka metode zaznavanja obraza.

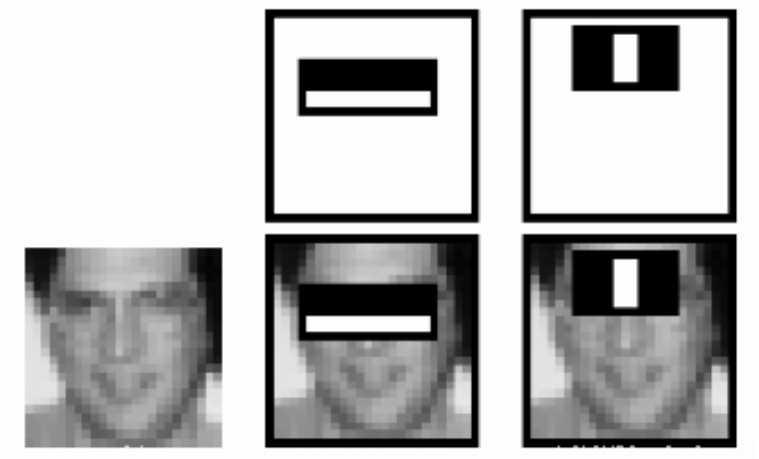
Imenujemo jih šibke klasifikatorje, ker en sam ne more določiti iskanega obraza. To kombiniranje je razvidno iz naslednje enačbe 3.1, kjer je: F močen klasifikator, f šibek klasifikator, x slika, α utež.

$$F(x) = \sum_{t=1}^{T} \alpha_t * h_t(x) \tag{3.1}$$

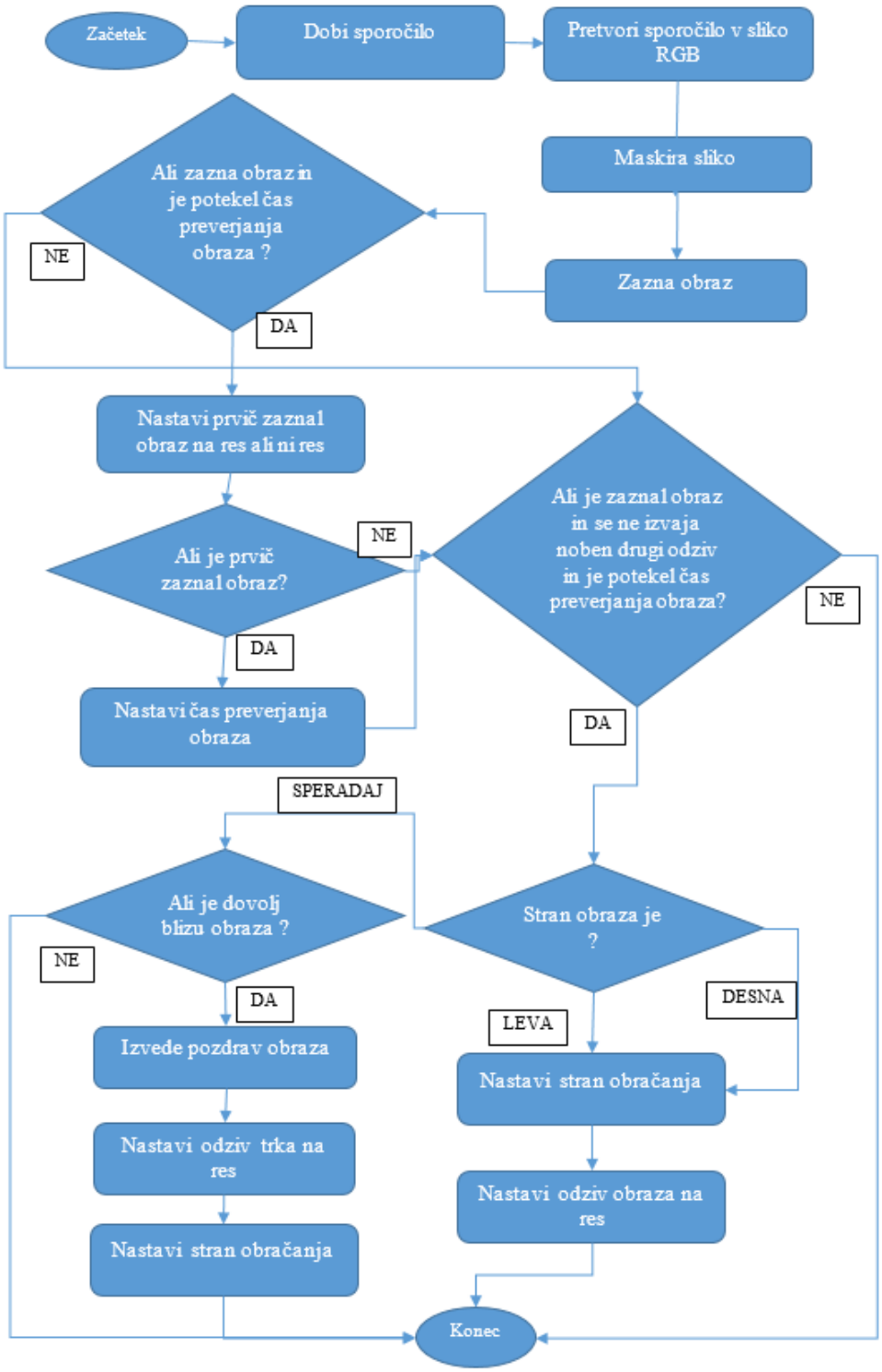
V našem primeru iskanja obraza uporabljamo klasifikatorje značilnice *Haar* [4], ki je vidna na sliki 3.7. Te značilnice uporabljamo na testni množici. Algoritem *AdaBoost* vsaki značilnici poišče najboljši prag, ki klasificira obraze na pozitivno - slika z obrazoma in negativno – slika brez obraza. Seveda pride do napak in napačnih klasifikacij, zato izberemo samo klasifikatorje z najmanjšimi napakami. To pomeni, da so najboljši klasifikatorji uporabljeni za zaznavo pravilne in nepravilne slike.



Slika 3.6: Adeboost učenje s značilnicami Haar (povzeto po [19]).



Slika 3.7: Iskanje obraza z AdeBoost, ki uporablja značilnice Haar ([5]).



Slika 3.8: Diagram poteka metode odziva na obraz.

3.3 Metode zaznavanj in odziv na črto

Metoda sprejme sporočilo, ki vsebuje RGB sliko. Dobljeno sliko maskira, kar zmanjša število zaznanih črt. Sliko pretvori v sivinsko, da lažje zazna črno-belo črto. Izvede iskanje robov z algoritmom *Canny* [2] na sliki. Na sliki robov izvede iskanje črt z algoritmom *Hough* [1], ki dobi seznam točk za vse črte na sliki. Iz dobljenih točk lahko začne preverjati, ali so preblizu. Pred tem normalizira histogram, kar normalizira svetlost in poveča kontrast slike in s tem poudari črto črno-bele barve ne glede na osvetlitev okolja. Črto, ki se nahaja preblizu robotske enote metoda preveri, če je črno-bele barve. Za te črte metoda sproži odziv in nastavi smer obračanja glede na stran črte. Delovanje metode je razvidno s slike 3.10 in postopek obdelave slike se vidi na sliki 3.11.

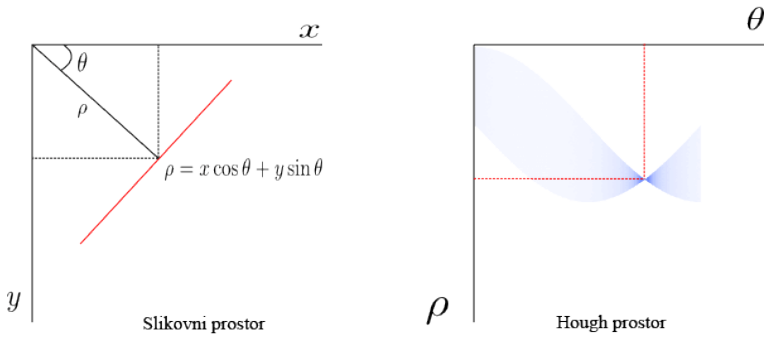
Hough transformacija [1] je algoritem za iskanje premic, elips in krogov v analizi slik, računalniškem vidu in digitalnem procesiranju slik. Namen tega algoritma je najti približni primer objekta, ki ga išče s postopkom glasovanja. Ta postopek glasovanja (ang. *voting*) se izvaja v parametričnem prostoru, iz katerega so pridobljeni kandidati objekta.

Hough transformacija uporablja dvodimenzionalni seznam imenovan akumulatorsko polje. Algoritem Hough definira obstoječe črte z enačbo 3.2. Prikaz premice v *Hougovem* polju se vidi na sliki 3.9. Dimenzija akumulacijskega polja se ujema s številom neznanih parametrov; za naš primer sta to dva približka od r in θ v paru (r,θ). Za vsako slikovno točko in njegove sosede bo algoritem Hough določil, ali je dovolj dokazov, da je na točki ravna premica. Če je premica ravna, ji izračuna parametre (r,θ) ter poišče celico v akumulatorskem polju in poveča vsebino celice za ena. Če *Hough* algoritem najde celice z največjimi vrednostmi v akumulatorskem polju, lahko dobimo najbolj verjetno premico in približek geometrijske definicije premice.

V naši implementaciji uporabljamo algoritem verjetne *Hough* transforma-

cije, ki deluje podobno kot navadna *Houghova* transformacija, le da namesto celotne množice točk robov vzame le del točk, kar tudi zmanjša kompleksnost glasovanja, kar pohitri delovanje algoritma.

$$r = x * sin * \theta + y * sin * \theta \tag{3.2}$$



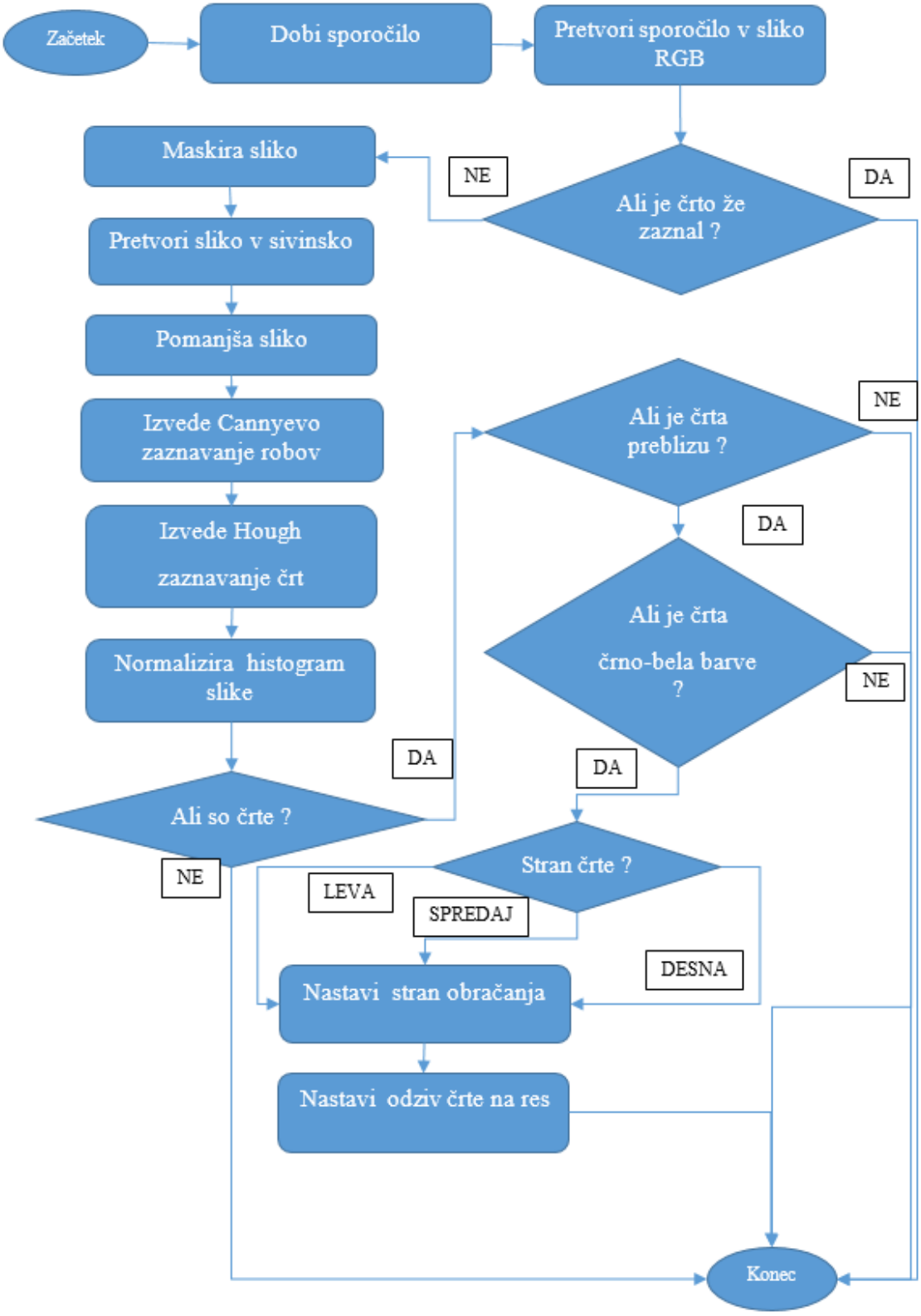
Slika 3.9: Prikaz premice v Houghovem prostoru (povzeto po [20]).

3.4 Metode odziv na oviro

Metoda sprejme sporočilo, ki vsebuje podatke oddaljenosti robotske enote v prostoru. Preveri, če je ovira preblizu. Če je, metoda sproži odziv na oviro in nastavi obračanje na 180°. Delovanje metode je razvidno s slike 3.12.

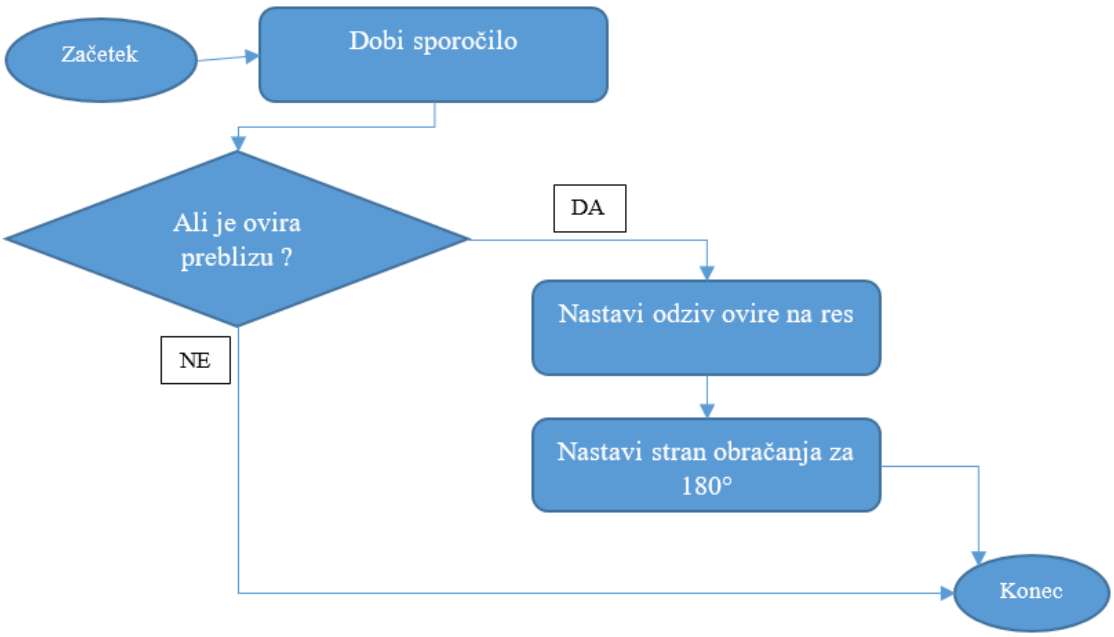
3.5 Metode odziv na trk in previs

Metoda sprejme sporočilo, ki vsebuje podatke o morebitnem trku zaznano z odbijača ali previsu zaznano z IR senzorjem. Če se je zgodil trk z oviro, metoda sproži odziv za trk in nastavi stran odziva glede na stran trka. Če se zgodi previs, metoda sproži odziv na previs in nastavi stran odziva glede na stran previsa. Delovanje metode je vidno na sliki 3.13.

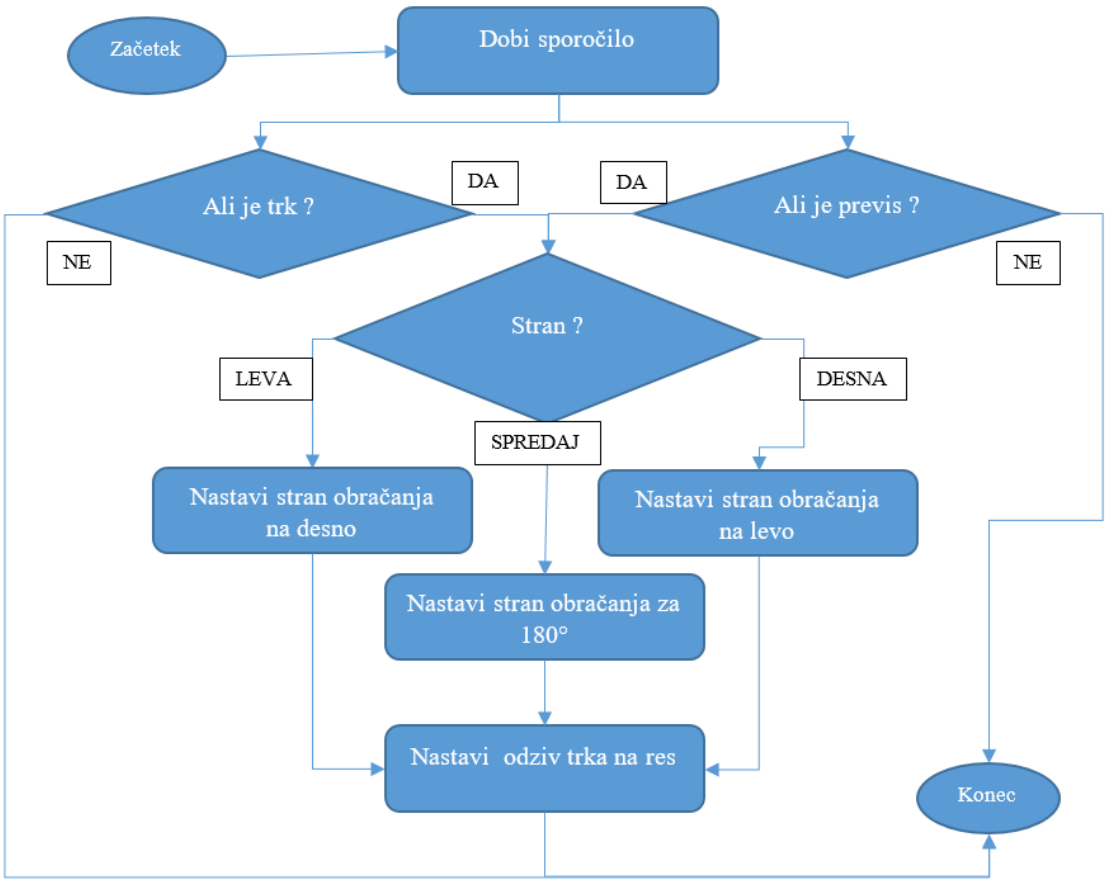


Slika 3.10: Diagram poteka metode zaznavanja in odziva na črto.

Slika 3.11: Potek obdelave slike z metodo odziv na črto: a) neobdelana slika b) maskirana slika c) sivinska slika d) pomanjšana slika e) slika robov f) slika najdenih črt h) slika z normaliziranim histogramom g) slika preverjenih slikovnih točk.



Slika 3.12: Diagram poteka metode odziva na oviro.



Slika 3.13: Diagram poteka metode odziva na trk in previs.

Poglavje 4

Evalvacija

V četrtem poglavju smo evalvirali delovanje implementiranega sistema. Da to dosežemo, smo izvedeli tri evalvacije. Prvi evalvira zaznavanje obraza. Drugi evalvira zaznavanje ter odziv na črto. Tretji in zadnji evalvira delovanje celotnega sistema na poligonu. Posamezne ugotovitve smo sklenili s pomočjo podatkov v obliki tabel, slik in grafov. Vsaka evalvacija vsebuje posamezen komentar za vsak del poskusa in na koncu še zaključni sklep celotne evalvacije.

4.1 Evalvacija zaznavanja obrazov

Metoda zaznavanja obraza je zelo pomembna za delovanje sistema, saj brez njega sistem ne more zagotoviti zahtev cilja o pozdravu obraza. Zato smo morali zagotoviti čimboljšo funkcionalnost delovanja metode za zaznavanje obraza.
Za poskus smo vzeli 5 testnih barvnih slik različnih obrazov, natisnjenih na belem papirju A4 formata, ki so bili pritrjeni na stojalu, slika 4.1. Ti obrazi so bili vsi na isti višini 35,5 cm in pred obraz smo na tla postavili 27 oznak za 9 kotov na 3 razdaljah. Ta postavitev se vidi iz slike 4.2. Pri tem smo upoštevali, da je bila najbližja razdalja 50 cm in najbolj oddaljena razdalja 150 cm, po kateri metoda ni več zaznala obrazov.

Slika 4.1: Testni obrazi od 1 do 5 iz leve proti desne.

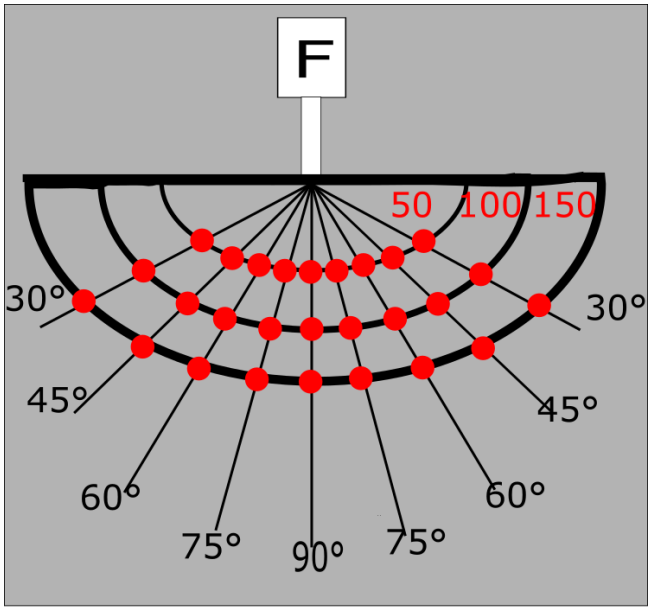
Za kote smo določili levo stran od L30° do L75°, sredino na C90° in desno stran D30° do D75°. Robotsko enoto smo postavili na oznake in preverili, če je zaznala obraz. Pri tem je robotska enota stala na mestu. Za vsak obraz smo to ponovili petkrat. Šteli smo le pravilno zaznane primere obraza. Skupaj smo dobili 45 primerov zaznav na posamezno razdaljo ter 25 primerov za posamezni kot. Skupaj smo imeli 225 primerov zaznav za analizo.

Metodi se je primer uspešnega zaznanja štel:

- če se je pravokotnik pravilno prikazal na obrazu,
- če se je v roku 2 sekund pravokotnik večkrat prikazal kot ne prikazal.

Metodi se primer uspešnega zaznanja ni štel:

- če se je v roku 2 sekund samo enkrat prikazal pravokotnik in potem izginil,
- če je zaznal napačno predmet kot obraz, prikazal pravokotnik tam, kjer ni obraza,
- če je narobe prikazal pravokotnik za obraz,
- če ni nič zaznal, pravokotnika ni prikazal.



Slika 4.2: Testno okolje za obraz.

4.1.1 Rezultati

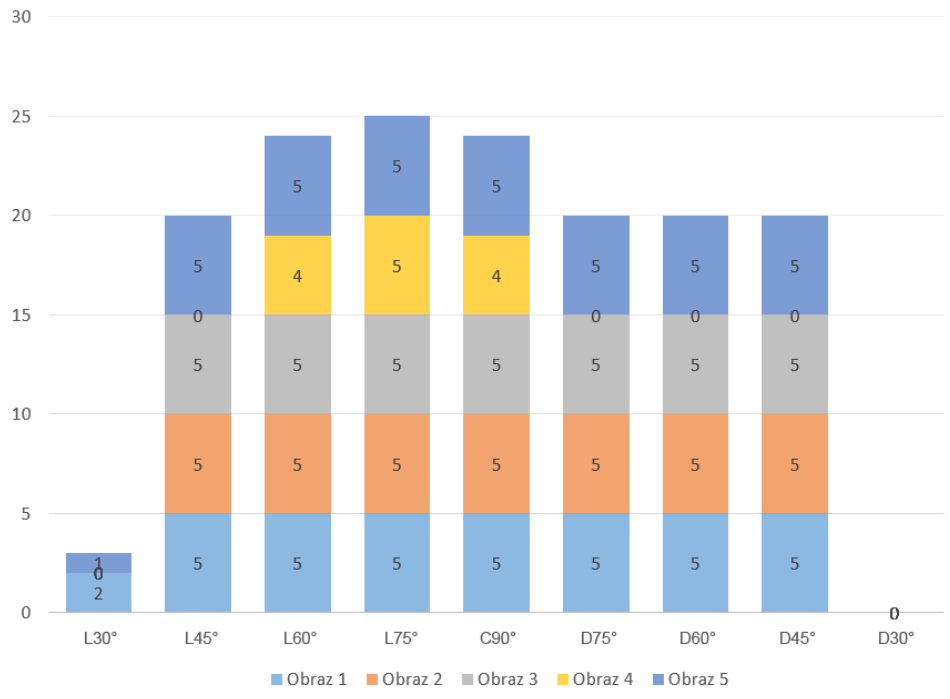
Z **razdaljo 50 cm** smo iz tabele 4.1 ugotovili, da je metoda najboljše delovala na kotu L75°, saj je metoda uspešno zaznavala vse testne obraze za vse primere. Metoda ni niti enega primera obraza uspešno zaznala na kotu D30°. Na kotu L30° pa je metoda imela samo tri primere obraza z uspešnim zaznavanjem.

Metoda je imela malo primerov uspešnih zaznavanj pri obrazu 4, kar je vidno na grafu 4.3. Predvidevamo, da je do tega prišlo, ker ima metoda težave z zaznavanjem obraznih struktur. Glede na ostale obraze je obraz 4 bolj zamegljen in ni dobro razviden prehod med obraznimi strukturami. Opazili smo tudi, da metoda zazna obraz 4 le na L60°, L75°ter C90°, ne pa na D 75° in D60°, kar so isti koti, le da so na drugi strani.

Ugotovitev za razdaljo 50 cm je, da je metoda v celoti dobro zaznalo vse obraze, razen obraza 4. V celoti je metoda uspešno zaznala kar 89% vseh primerov.

	Leva				Center	Desna				
Obraz	30°	45°	60°	75°	90°	75°	60°	45°	30°	Skupaj
1	2	5	5	5	5	5	5	5	0	37
2	0	5	5	5	5	5	5	5	0	35
3	0	5	5	5	5	5	5	5	0	35
4	0	0	4	5	4	0	0	0	0	13
5	1	5	5	5	5	5	5	5	0	36
Skupaj	3	20	24	25	24	20	20	20	0	156

Tabela 4.1: Število primerov zaznav na razdalji 50 cm.



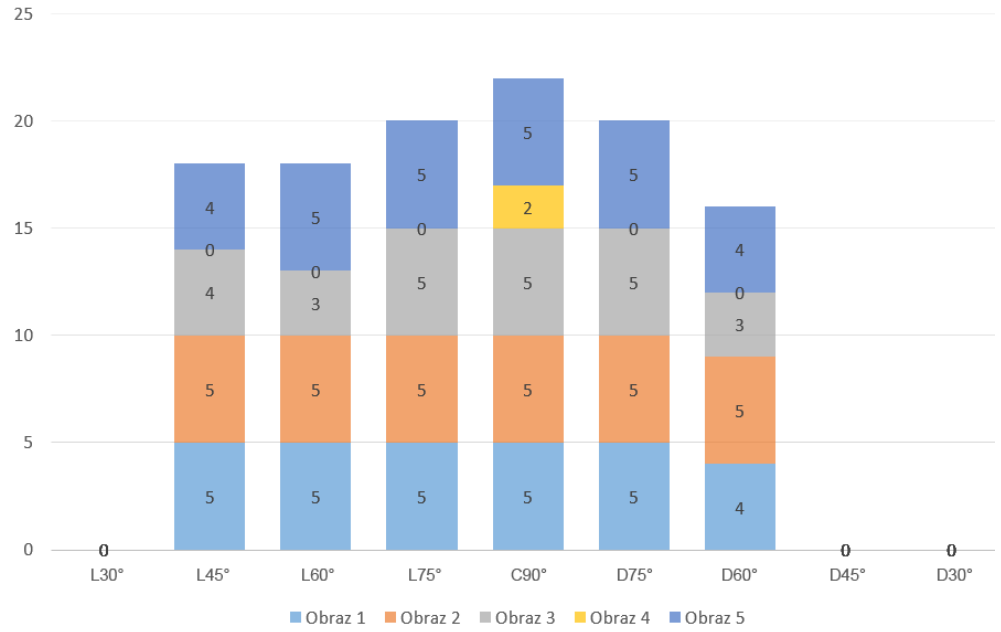
Slika 4.3: Primeri zaznav na razdalji 50 cm.

Za **razdaljo 100 cm** smo ugotovili iz tabele 4.2, da je metoda najboljše delovala na kotu C 90°. Tu je metoda uspešno zaznala štiri testne obraze za vse primere zaznavanj. Metoda je imela za obraz 2 najboljši uspeh z zaznavo. Z grafa 4.4 je razvidno, da metoda obraza 4 skoraj ne zazna več. Uspešno ga je zaznala le v dveh primerih za kot C 90°. Dodatno lahko opazimo, da metoda ne zazna več obrazov na kotu D45°.
Za razdaljo 100 cm ugotavljamo, da je število uspešnih primerov zaznav padlo s 156 na 114, kar je za 42 manj primerov. To pomeni, da metoda na povečani razdalji izgubi učinkovitost zaznavanja obraza.

	Leva				Center	Desna				
Obraz	30°	45°	60°	75°	90°	75°	60°	45°	30°	Skupaj
1	0	5	5	5	5	5	4	0	0	29
2	0	5	5	5	5	5	5	0	0	30
3	0	4	3	5	5	5	3	0	0	25
4	0	0	0	0	2	0	0	0	0	2
5	0	4	5	5	5	5	4	0	0	28
Skupaj	0	18	18	20	22	20	16	0	0	114

Tabela 4.2: Število primerov zaznav na razdalji 100cm.

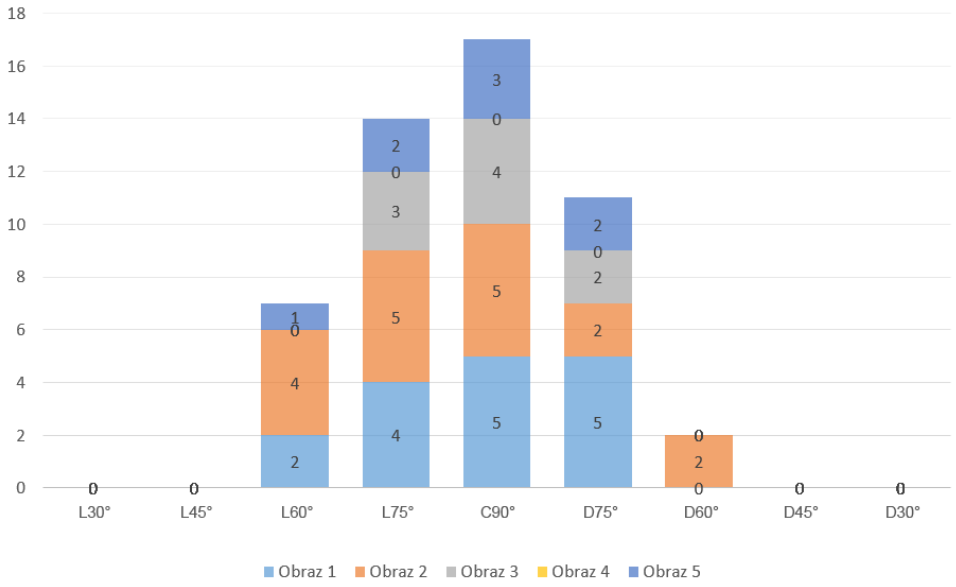
Iz tabele 4.3 razberemo, da metoda za kot C90° na **razdaljo 150 cm** zazna obraze najboljše. Za obraze 1 in 2 je metoda uspešno zaznala največ primerov glede na ostale obraze.
Metoda obraza 4 ni niti enkrat zaznala. Dodatno smo ugotovili, da metoda ne zaznava več obrazov na kotu L45°.
Za razdaljo 150 cm smo ugotovili iz grafa 4.5, da se je metodi število uspešnih zaznav zmanjšalo na približno polovico, za razliko od meritev na razdalji 100 cm in kar na dve tretjini pri razdalji 150 cm. To potrdi prejšnjo ugotovitev, da se ob povečani razdalji metodi poslabša zaznavanje obrazov.



Slika 4.4: Primeri zaznav na razdalji 100 cm.

	Leva				Center	Desna				
Obraz	30°	45°	60°	75°	90°	75°	60°	45°	30°	Skupaj
1	0	0	2	4	5	5	0	0	0	16
2	0	0	4	5	5	2	2	0	0	18
3	0	0	0	3	4	2	0	0	0	9
4	0	0	0	0	0	0	0	0	0	0
5	0	0	1	2	3	2	0	0	0	8
Skupaj	0	0	7	14	17	11	2	0	0	51

Tabela 4.3: Število primerov zaznav na razdalji 150cm.



Slika 4.5: Primeri zaznav na razdalji 150 cm.

4.1.2 Sklep

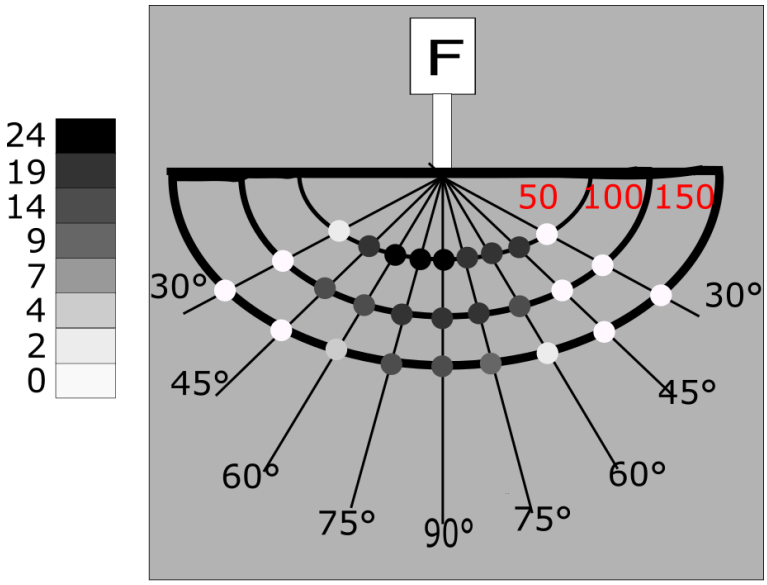
Z merjenjem na različnih razdaljah lahko sklepamo, da je metoda najboljše zaznala obraze na kotu C90°. Poleg tega metoda tudi dobro zazna obraz na kotu L75°.

Pri kotih L30° in D30° se je izkazalo, da metoda ne more zanesljivo zaznati obraza, kar se vidi z grafa 4.7, saj je uspešno zaznal le 3 primere za vse razdalje skupaj.

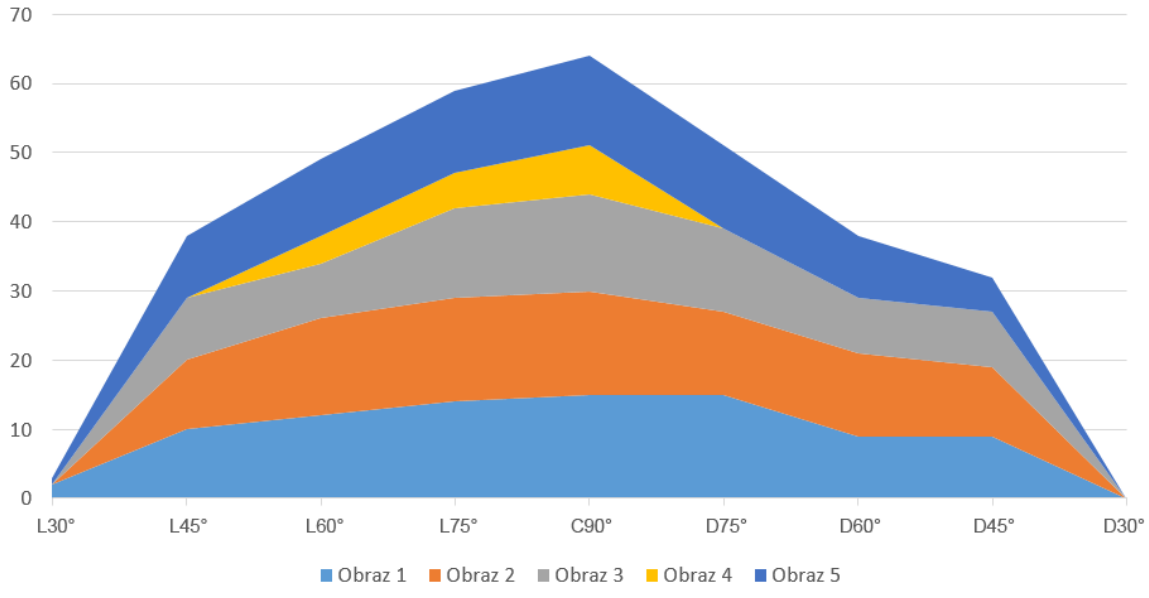
Metoda je, glede na ostale, obraza 1 in 2 najboljše zaznala. Predvidevamo da zato, ker je obrazna struktura na sliki bolj vidna in se boljše razloči meja med ozadjem in obrazom na sliki.

Metoda slabše zazna obraza 3 in 5 kot obraza 1 in 2, ker imata bolj osvetljene površine na obrazu in ozadju slike. To pripomore k prelivu struktur ob izenačenju histograma. Ta pojav ni tako viden kot pri obrazu 4.

Metoda glede na vse ostale obraze najslabše zazna obraz 4, saj ga je uspešno zaznala le na razdalji 50 cm.



Slika 4.6: Prikaz zaznav glede na pozicijo v testnem okolju za obraz.



Slika 4.7: Primeri zaznav na vseh razdaljah.

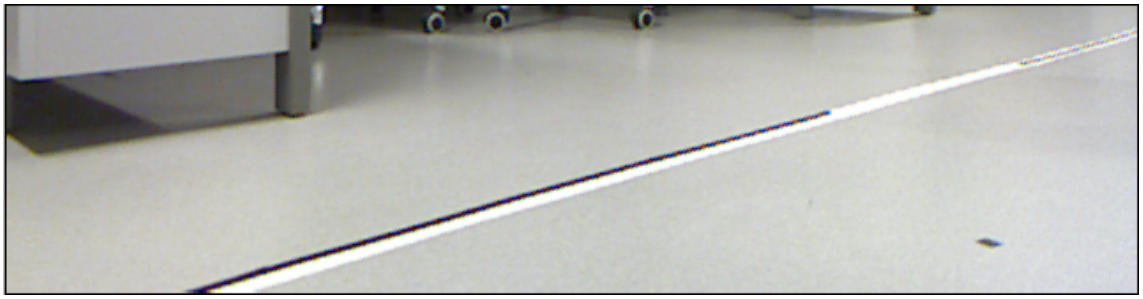
To se vidi iz tabele 4.4. Sklepamo, da je vzrok to, da metoda zaznavanja obraza izenači histogram, kar povzroči, da temni deli slike postanejo še temnejši in s tem se poslabša poudarek obraznih struktur. Ta posledica je za ostale obraze dobra, saj poudari strukturo le-tega.
Ugotovitev glede na poskuse pri različnih razdaljah je, da metoda zaznavanja obraza dobro zaznava do 100 cm, nakar se začne slabšati. Na razdaljah večjih od 150 cm ne moremo več zagotoviti zaznavanja obrazov. Prikaz vse lokacij z rezultati glede na učinkovitost zaznav v testnem okolju se razbere iz slike 4.6.

	Leva				Center	Desna				
Obraz	30°	45°	60°	75°	90°	75°	60°	45°	30°	Skupaj
1	2	10	12	14	15	15	9	9	0	86
2	0	10	14	15	15	12	12	10	0	88
3	0	9	8	13	14	12	8	8	0	72
4	0	0	4	5	7	0	0	0	0	16
5	1	9	11	12	13	12	9	5	0	72
Skupaj	3	38	49	59	65	51	38	31	0	334

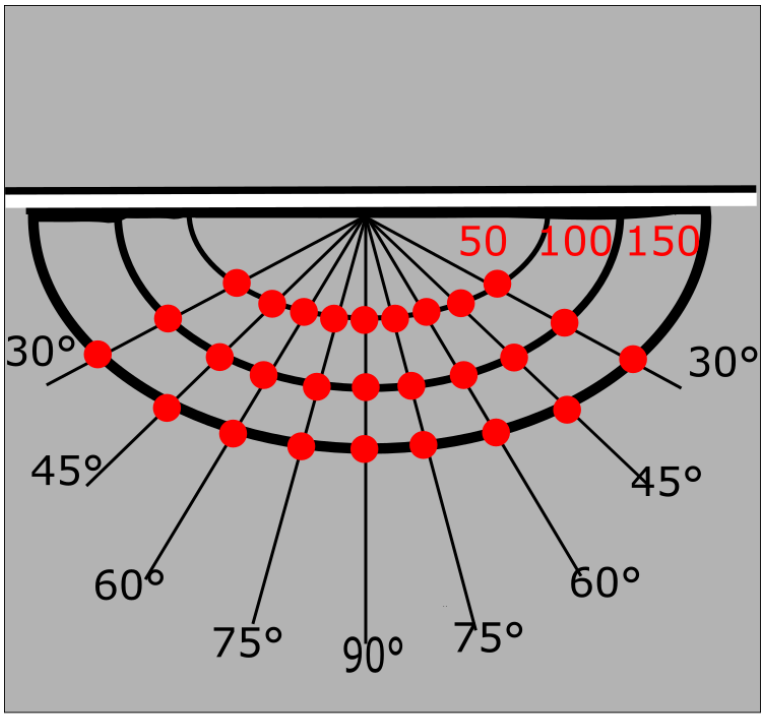
Tabela 4.4: Skupno število primerov zaznav na vseh razdaljah.

4.2 Evalvacija odziva na črte

Metoda zaznavanja in odziva na črto je pomembna, saj smo si kot cilj zadali, da je sposobnost sistema zaznati črto črno-bele barve in se črti izogniti. Zato moramo preveriti, kako dobro jih metoda zaznava in če se pravilno odziva nanjo. Preverili bomo tudi, kolikokrat metoda zazna napačno črto kot pravo, saj želimo, da te črte metoda ignorira.
Za testiranje smo uporabili enako postavitev kot v prejšnjem podpoglavju evalvacije zaznavanja obraza, kar se vidi na sliki 4.9.



Slika 4.8: Črta črno-bele barve, katero uporabljamo za omejevanje prostora.



Slika 4.9: Testno okolje za črto.

Razlikuje se le v tem, da imamo namesto obraza postavljeno črto in robotska enota na začetku miruje. Po določenem času robotska enota začne premik naprej. S to evalvacijo preverimo, kako se metoda odziva glede na različne razdalje začetne lokacije premikanja.
Za črto smo uporabili črni in beli lepilni trak, ki so bili postavljeni na talni površini prostora. Ta trak se vidi na sliki 4.8

Imamo dva poskusa zaznavanja in odziva črno-bele črte in neodzivnost na navadno črto. Za vsak kot preizkusimo delovanje 5-krat na treh razdaljah. Če je metoda izvedla pravilni odziv bomo to šteli k razdalji, s katere je začela robotska enota. Tako ugotovimo, koliko vpliva razdalja na odziv na črto.
Za vsako črto je bilo skupaj 15 primerov odzivov glede na posamičen kot za vsako razdaljo. Za eno razdaljo je bilo 45 primerov odzivov. Za vse skupaj je bilo 135 primerov odzivov. Število primerov je enako za obe črti.

4.2.1 Rezultati s črno-belo črto

Uspešen primer odziva na črno-belo črto smo šteli:

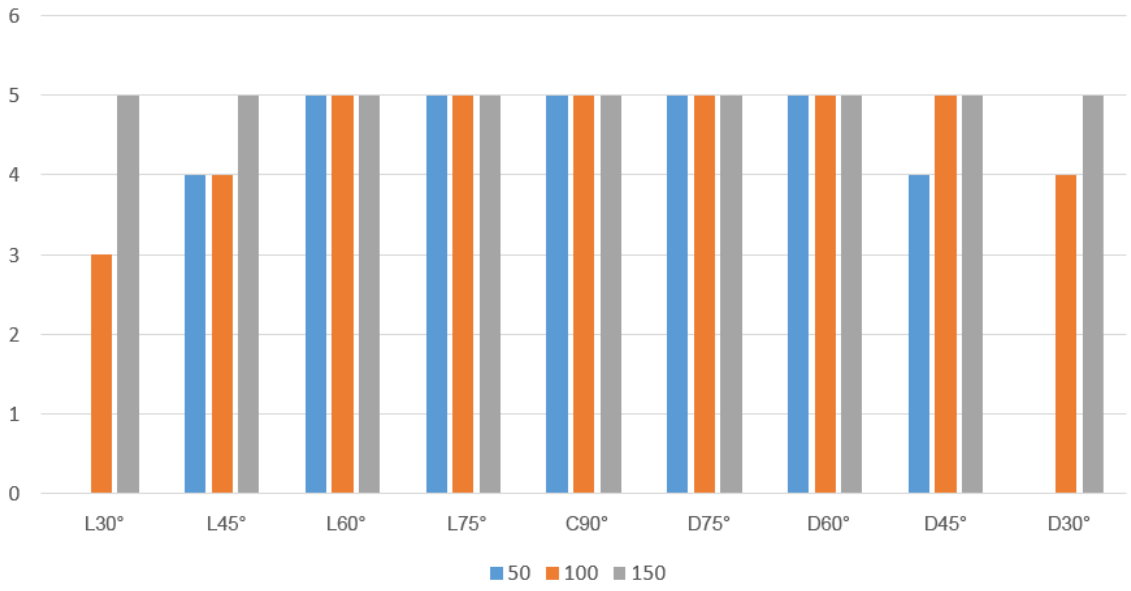
- če ob premikanju naprej metoda zazna črto kot črno-belo barve in se ji izogne,
- če se že na mestu sproži izogibanje.

Odziv nismo šteli kot uspešen:

- če ignorira črto in se premika naprej.

V poskusu s **črno-belo črto** smo iz grafa 4.10 videli, da za kote L60°, L75°, C90°, D75°in D60° metoda deluje zelo dobro, saj se je v vseh primerih metoda pravilno odzval na črto ne glede na različne razdalje. Najboljše je metoda delovala na razdalji 150 cm, kjer se je za vse primere, ne glede na kot, metoda pravilno odzvala. To je verjetno zato, ker ima ob povečani razdalji več časa za zaznavo črte.

Ugotovili smo, da se metoda na kotu L30° in D30° nepravilno odziva na razdalji 50 cm, kar je ratvidno iz tabele 4.5. Predvidevamo da zato, ker je robotska enota preblizu črte in je ne more dovolj hitro zaznati ter se posledično ne more odzvati nanjo. Ta zamisel se potrdi z rezultatom metode pri istem kotu na razdalji 100 cm, saj se je povečalo število pravilnih primerov odzivov. Povečanje razdalje izboljša delovanje metode, dokler ne pride na razdaljo 150 cm, kjer se je metoda odzvala na vseh primerih pravilno.
To lahko tudi opazimo pri kotih L45° in D45°, ko se poveča razdalja do črte se metodi izboljša število uspešnih primerov odziva.
Ugotovili smo, da metoda za črto črno-bele barve robustno deluje. Kar 119 od 135 je pravilnih primerov. Torej zagotavljamo robustno delovanje glede na zaznavanje črno-bele črte.



Slika 4.10: Primeri odzivov na črto črno-bele barve.

	Leva				Center	Desna				
Razdalje	30°	45°	60°	75°	90°	75°	60°	45°	30°	Skupaj
50 cm	0	4	5	5	5	5	5	4	0	33
100 cm	3	4	5	5	5	5	5	5	4	41
150 cm	5	5	5	5	5	5	5	5	5	45
Skupaj	8	13	15	15	15	15	15	14	9	119

Tabela 4.5: Primeri odzivov na črto črno-bele barve.

4.2.2 Rezultati z navadno črto

Za ta poskus preverjamo, kako metoda ločuje med navadno in črno-belo črto. Za omejevanje prostora uporabljamo samo črto črno-bele barve, zato želimo, da ostale črte ignorira, s tem se lahko premika po prostoru in se ne izogiba vsaki črti, ki jih zazna. Za testiranje delovanja metode na navadno črto bomo v tem poskusu uporabili črto bele barve, postavljeno na talni površini.
Ker smo iz poskusa črno-bele črte ugotovili, da za L 30° in D 30° na razdalji 50 cm ne zaznava prav, ga v tem poskusu ne upoštevamo pri ugotovitvah.

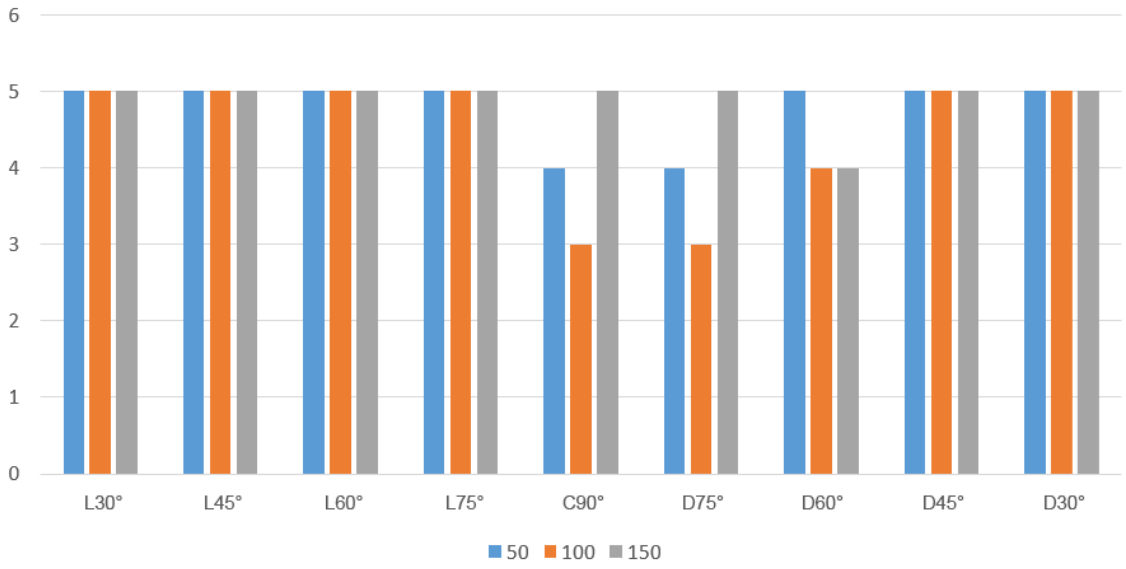
Uspešen primer odziva na navadno črto smo šteli:

- če metoda ignorira črto in se premika naprej.

Odziv na navadno črto nismo šteli:

- če ob premiku naprej metoda zazna črto kot črno-belo in se ji poskusi izognit,
- če že na mestu metoda zaznala črto in se ji poskusi izognit.

Pri poskusu **navadne črte** smo iz grafa 4.11 in tabele 4.6 ugotovili, da se na kotih L45°, L60°, L75° in D45° ni metoda v nobenem primeru odzvala na črto. Na razdalji 150 cm metoda zelo dobro deluje, saj ima glede na ostale razdalje največje število uspešnih primerov. Zanimivo je tudi, da se je metoda na razdalji 50 cm na kotih C90° in D75° boljše obnesla kot na razdalji



Slika 4.11: Primeri odzivov na navadno črto.

100 cm, ki ima manjše število uspešnih primerov. To se ne ujema s prejšnjo ugotovitvijo, da na večji razdalji metoda boljše deluje. Vzrok je morda zato, da na razdalji 50 cm leži črta na meji vidnega polja, zato posledično metoda nima toliko časa za zaznavo črte ter ima manjšo možnost, da pride do napake. Na kotu C90° in D 75°se je metoda najslabše obnesla glede na ostale kote, ker osenčena tla zazna kot črni del črte.

	Leva				Center	Desna				
Razdalje	30°	45°	60°	75°	90°	75°	60°	45°	30°	Skupaj
50 cm	5	5	5	5	4	4	5	5	5	43
100 cm	5	5	5	5	3	3	4	5	5	40
150 cm	5	5	5	5	5	5	4	5	5	44
Skupaj	15	13	15	15	12	12	13	15	15	125

Tabela 4.6: Primeri odzivov na navadno črto.

4.2.3 Sklep

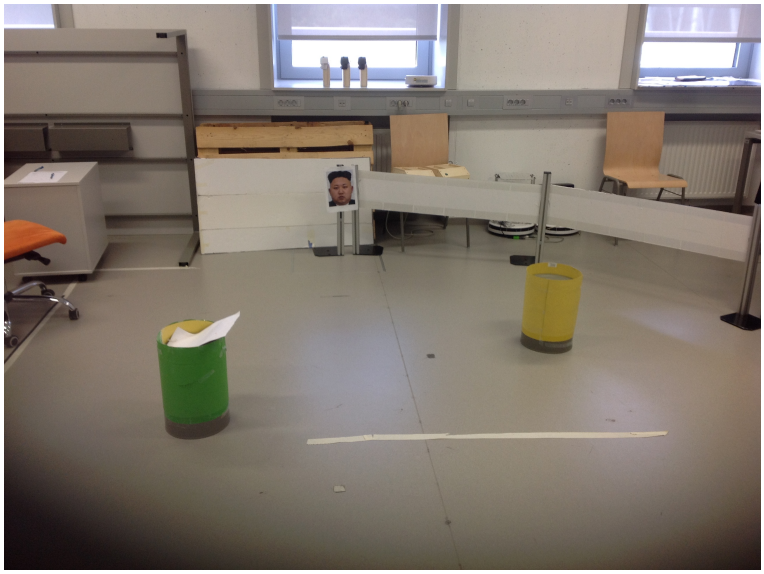
Iz tabele 4.5 in tabele 4.11 ugotavljamo, da je metoda dokaj robustna. To je opazno pri tem, da je od 270 skupnih možnih primerov, kar 244 uspešnih primerov. Glede na razdalje smo opazili, da metoda deluje boljše, ko ima večjo razdaljo med robotsko enoto in črto. Poveča se čas možnega zaznavanja in odziva. To velja za navadno in črno-belo črto.
Metodo se najboljše odziva na črno-belo črto pri kotih: L60°, L75°, C90°, D75°, D60°, navadno črto pa pri kotih: L30°, L45°, L75°, L60°, L45°, L30°. Pri kotu L30° in D30° na razdalji 50 cm ima metoda mrtvi kot, katerega zaznavanje je nemogoče.
Pri zaznavanju navadne črte ima metoda malo napak, kadar se premika naravnost proti črti. Ta se opazi pri kotih C90° in D75° na razdalji 50 cm in 100 cm. To bi lahko preprečili, tako da bi metodi izboljšali ločevanje med črno-belo črto in tlemi. Metodi bi implementirali večjo število zaznav slikovnih pik. Te izboljšave metode bi izboljšale tudi rezultate pri odzivu na črno-belo črto za kote L30°, L40°, D 5°, D35°.

4.3 Evalvacija celotnega sistema

Poleg zaznavanja obraza in odziva na črte je zelo pomembno tudi celotno delovanje sistema. To smo preverili s poligonu viden na sliki 4.12. Poligon ima isto talno površino in različno postavljene ovire. Robotsko enoto smo pustili, da se premika po poligonu 20 minut in šteli, kolikokrat je prišlo do pravilnega ali nepravilnega odziva sistema. Pri teh poskusih previsa nismo preverjali. Beležili smo sledeče odzive.

Odziv na oviro pred robotsko enoto:

- Pravilen: izogne se oviri
- Napačen: trči v oviro.



Slika 4.12: Poligon.

Odziv na črto črne-bele barve na tleh:

- Pravilen: izogne se črti in prepreči premikanje čeznjo.
- Napačen: črto ignorira in se premika čeznjo.

Odziv na navadno črto na tleh:

- Pravilen: črto ignorira in se premika čeznjo.
- Napačen: črto zazna kot črto črne-bele barve.

Odziv na obraz:

- Pravilen: Ko je dovolj blizu obraza, ga pozdravi.
- Napačen: Pozdravi obraz, kjer obraza sploh ni.

Odziv na trk:

- Pravilen: Premika se vzvratno od ovire in obrne glede na smer trka.
- Napačen: Ne zazna trka in se premika naprej proti oviri.

Vsaka postavitev poligona ima skico z označenimi lokacijami, kot so ovire, zidovi, črte, obrazi, kje so se odzivi zgodili in začetno lokacijo robotske enote. Testiranje smo začeli tako, da postavimo robotsko enoto na začetno pozicijo in jo pustimo delovati 20 minut. Robotsko enoto smo pustili delovati nemoteno in jo premaknili samo v primeru, da ne bi prišlo do morebitne poškodbe robotske enote ali prenosnega računalnika. Ko je prišlo do izvajanja, smo odziv zabeležili na skici poligona. Ker je poligon majhen, so se nekateri odzivi zgodili na podobnem mestu. Zato se število simbolov odzivov na skici ne ujema s številom zapisanih odzivov v tabeli.

Legenda poligona:

- B ali rdeči pravokotnik: testna ovira za preverjanje, ali se odbijač pravilno odziva na trk.
- L ali črno-bela črta.
- F ali beli pravokotnik: slika obraza na višini 35,5 cm.
- FL ali bela črta.
- W ali črna črta: zid, ki šteje kot ovira.
- OB ali rdeči obrobljeni krogi: ovire na poligonu.
- Simbol zeleni X: predstavlja, kje se je približno sprožil pravilen odziv.
- Simbol rdeči krog O: predstavlja, kje se je približno sprožil napačen odziv.
- S ali beli krog s puščico: začetna lokacija robotske enote s smerjo zaznavanja.

4.3.1 Rezultati

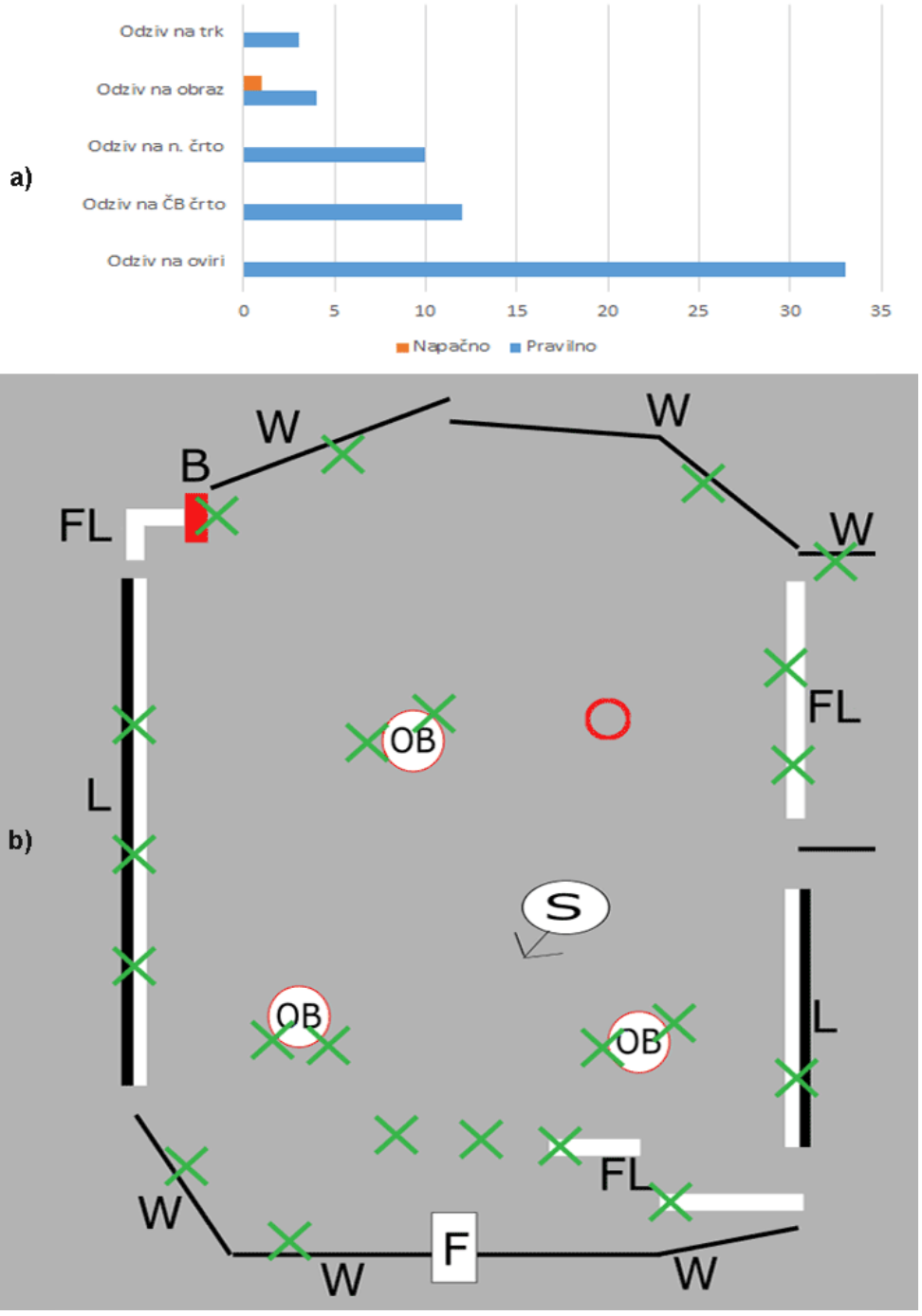
Za **prvo postavitev poligona**, viden na sliki 4.13, smo ugotovili iz grafa 4.13, da je sistem izvedel največ odzivov na oviro ter vse odzive pravilno opravil.

	Odziv na oviri	Odziv na ČB. črto	Odziv na n. črto	Odziv na obraz	Odziv na trk	Skupaj
Pravilno	33	12	10	4	3	62
Napačno	0	0	0	1	0	1
Skupaj	33	12	10	5	3	63

Tabela 4.7: Primeri odzivov za prvo postavitev poligona.

Sistem se je odzival na trk le na lokaciji namenjeni trku in zato se ta ni štel med napačne odzive na ovire.
Sistemov odziv na črno-belo ter navadno črto se je skupaj pojavil manjkrat kot ovire, kar se vidi iz tabele 4.7, ampak še vedno dovoljkrat, da lahko navedemo, da se je metoda zaznavanja in odziva na črto dobro izvajala. Moramo opomniti, da se je robotska enota črtam približevala pod koti med 45° in 90°ter razdaljo večjo od 50 cm, kar smo ugotovili iz prejšnje evalvacije metode zaznavanj in odziv na črto.
Na prvi postavitvi poligona smo uporabljali le en obraz, kar se odraža v manjšem številu zabeleženih primerov odzivov. Sistem sprožil tudi en napačen primer odziva na obraz, ki ga je zaznal v delu prostora, čeprav ga tam ni bilo. Ta odziv na obraz je bil zaznan kot dovolj blizu , da je sistem izvedel pozdrav. Pri tem moramo opomniti, da ob izvedbi poskusa še nismo implementirali ponovnega preverjanja obraza za preprečevanje zaznav napačnih obrazov.

V **drugi postavitvi poligona**, vidnem na sliki 4.14, se je sistem največkrat odzval na ovire ter deluje zelo dobro, saj ima največ pozitivnih primerov odzivov glede na ostale. To je vidno v tabeli 4.8 in grafu 4.14. Sistem je sprožil odziv trka le na označenem mestu, tako da ni odziv trka vplival na rezultat odziva na ovire. Ko se je sistem odzval na navadno črto, je prišlo do večjega števila napak, saj smo na drugi postavitvi poligona postavili novo črto. Ti primeri napačnih odzivov so se izvajali med koti 75° in 90°, ki smo jih ugotovili iz prejšnje evalvacije metode zaznavanja in odziva na črto.



Slika 4.13: a) Primeri odzivov glede na napačno ali pravilno izvedbo. b) Skica prve postavitve poligona.

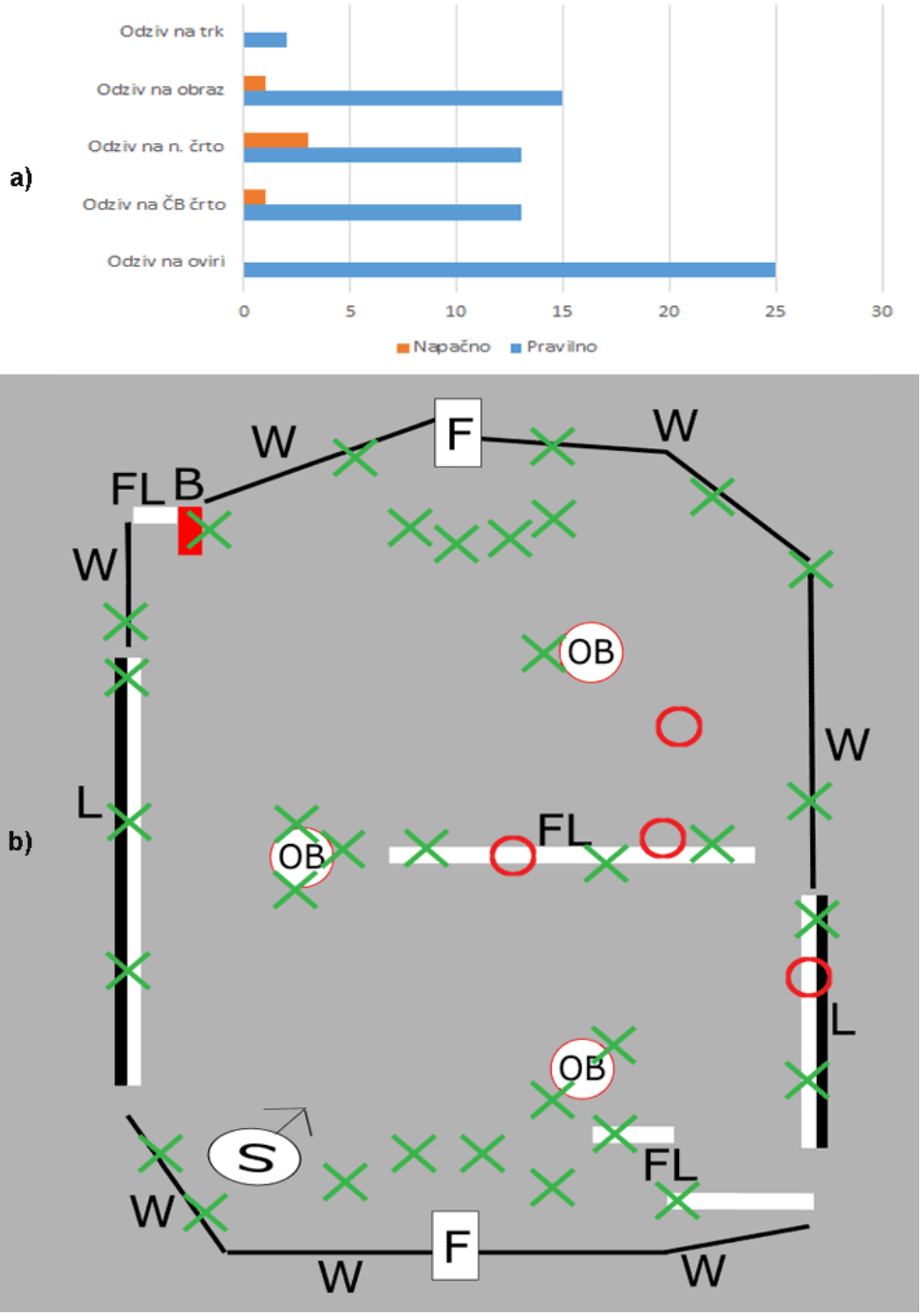
	Odziv na oviri	Odziv na ČB. črto	Odziv na n. črto	Odziv na obraz	Odziv na trk	Skupaj
Pravilno	25	13	13	15	2	68
Napačno	0	1	3	1	0	5
Skupaj	25	14	16	16	2	72

Tabela 4.8: Primeri odzivov za drugo postavitev poligona.

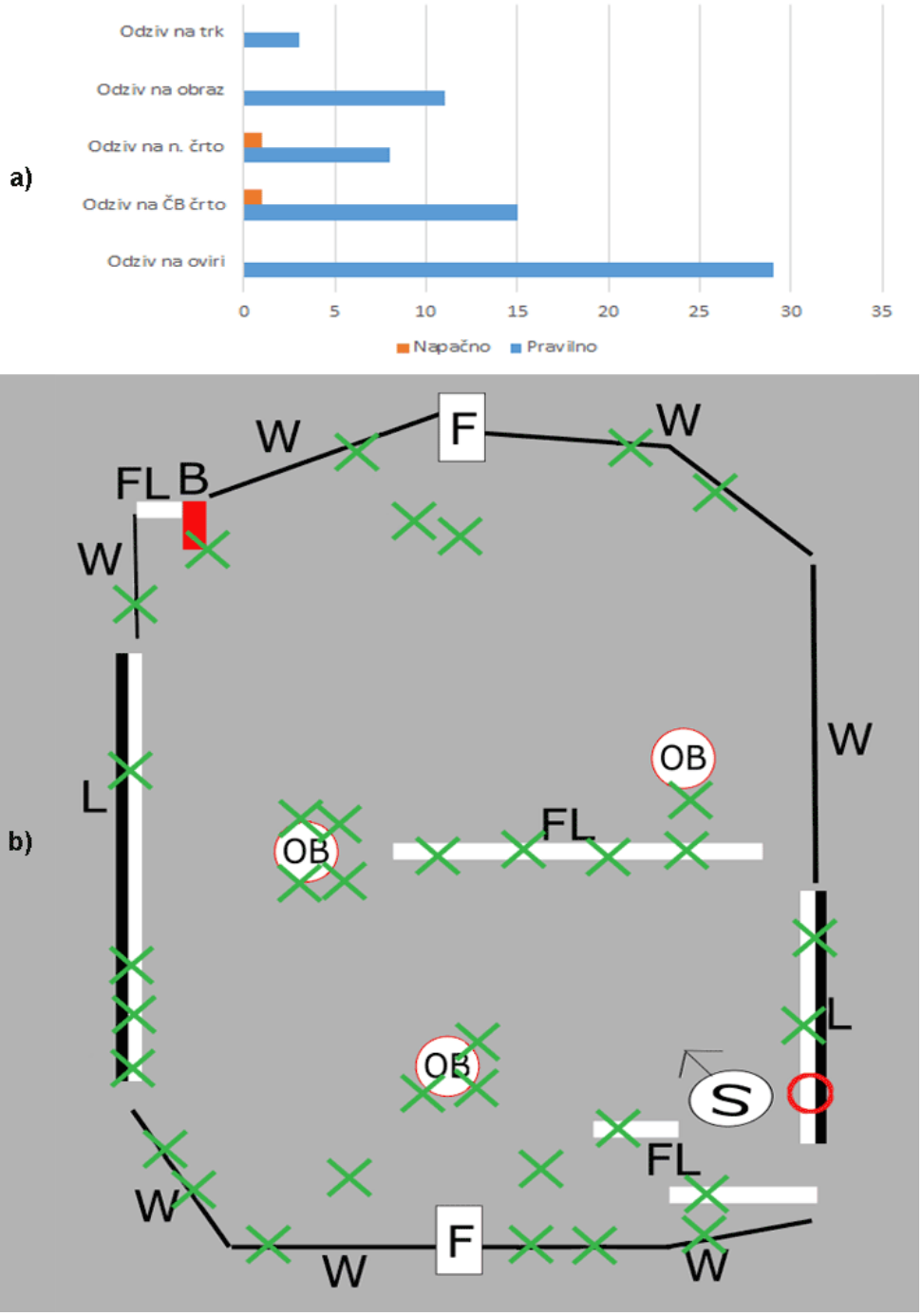
Napačen primer sistemskega odziva na črto črno-belo barve se je izvedel zaradi kota manjšega od 30°.
Za primere odziva na obraze vidimo, da sistem izvajal več odzivov na obraz, saj smo na poligon postavili dodaten obraz. V celoti je delovanje sistema dobro, razen v enem primeru napačnega odziva, ki se je zgodil na istem mestu kot v prvi postavitvi poligona.

Za **tretjo postavitev poligona**, viden na sliki 4.15, smo opazili, da je sistem spet največkrat izvedel odziv na ovire in se najboljše obnesel. Odziv na kolizijo se je ponovno izvedlo samo na označeni lokaciji.
Za odziv sistema na črno in belo ter navadno črto so napake nastale na kotih med 75° in 90°ter med 0° in 30°.
Pri sistemskem odzivu na obraz smo ugotovili, da več ne zaznava napačnega obraza, saj smo implementirali ponovno preverjanje obraza, ki preprečuje takojšen odziv na predmet v prostoru, ki ni obraz. To se vidi iz tabele 4.9 in grafa 4.15.

Iz **četrte postavitve poligona**, vidnega na sliki 4.16, smo ugotovili, da je rezultat delovanja sistem na ovire enak prejšnjemu. V poligonu smo spremenili lokacijo ovir. Ugotovili smo lahko, da se je robotska enota med dvema ovirama začela odbijati od ovire do ovire in zato je sistem potreboval nekaj časa, da se je lahko rešil in nadaljeval s premikanje. To je vplivalo na povečanje števila zabeleženih primerov odzivov.



Slika 4.14: a) Primeri odzivov glede na napačno ali pravilno izvedbo. b) Skica druge postavitve poligona.

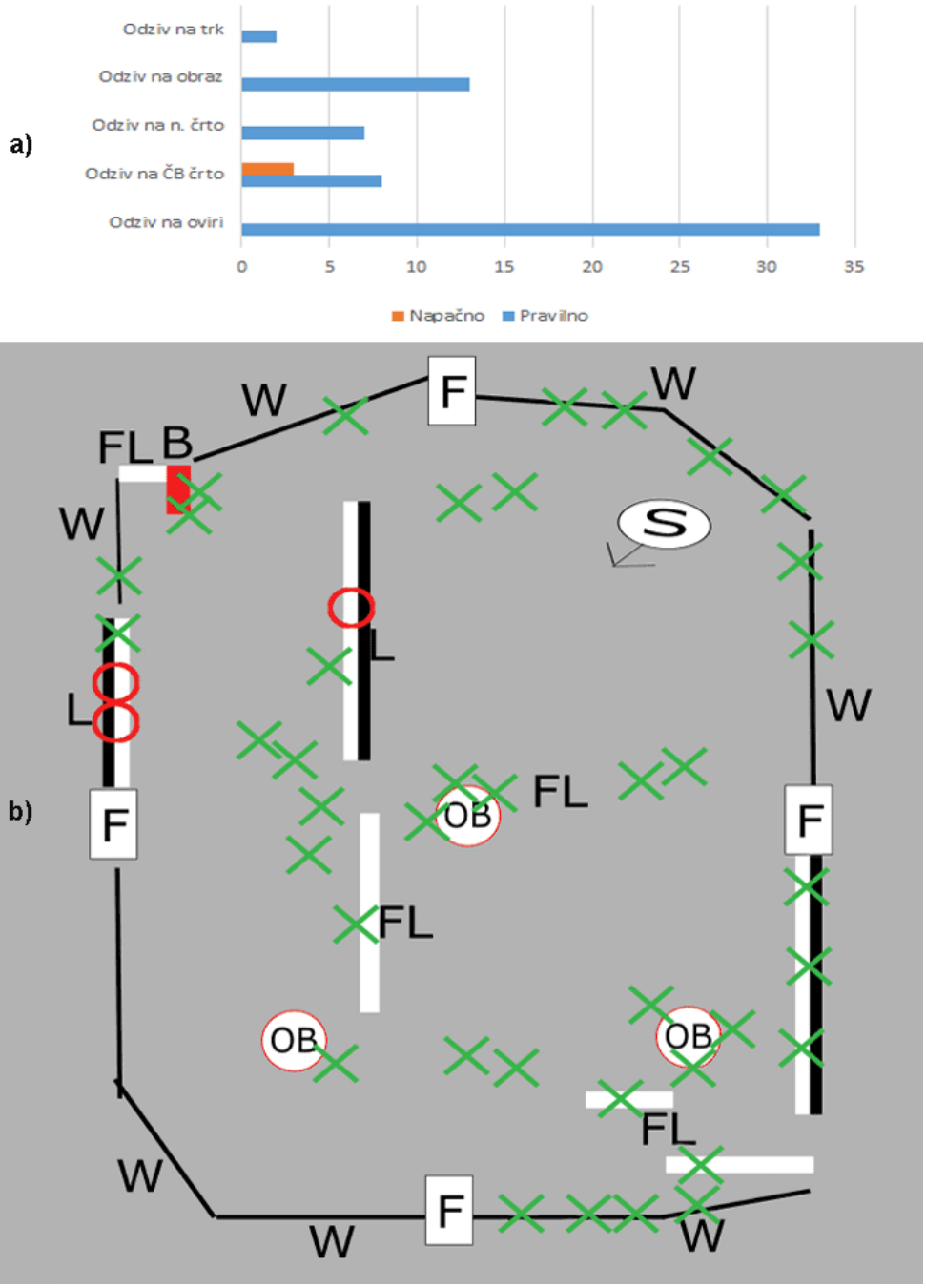


Slika 4.15: a) Primeri odzivov glede na napačno ali pravilno izvedbo. b) Skica tretje postavitve poligona.

	Odziv na oviri	Odziv na ČB. črto	Odziv na n. črto	Odziv na obraz	Odziv na trk	Skupaj
Pravilno	29	15	8	11	3	66
Napačno	0	1	1	0	0	2
Skupaj	29	16	9	11	3	68

Tabela 4.9: Primeri odzivov za tretjo postavitev poligona.

Sistem je trke sprožil spet na označeni lokaciji trka na poligonu.To je razvidno iz tabele 4.10 in grafa 4.16.
Sistemov odziv na navadno črto je deloval brezhibno. Verjetno je bil razlog, da smo premaknili črto glede na prejšnjo postavitev poligona, zato ni prišlo do napačnega odziva med koti 75° in 90°. Pri sistemskem odzivu na črno-belo črto se je zgodilo več napak. Robotska enota je v enem primeru po odzivu na oviro pristala na kotu, ki je bil manjši od 30° ter z manjšo razdaljo do črte. V drugem primeru po odzivu na oviro se je robotska enota premikala čez črto. ker ima sistem za odziv na oviro višjo prioriteto kot odziv na črta. To pa deluje, kot smo nameravali, saj preprečuje morebitne poškodbe robotske enote. Za četrto postavitev poligona imamo štiri različne obraze. Tega se ne vidi v številu vseh sproženih odzivov na obraz. Predvidevamo, da zato ker so pred obrazi ovire in črno-bela črta, ki imajo višjo prioriteto kot odziv na obraz. Prišlo pa je tudi do kotov med 0° in 30°, kar je zmanjšalo možnost zaznavanja obraza. To je razvidno iz evalvacije metode za zaznavanja in odziva obraza.



Slika 4.16: a) Vsi primeri odzivov glede na napačno ali pravilno izvedbo. b) Skica četrte postavitve poligona.

	Odziv na oviri	Odziv na ČB. črto	Odziv na n. črto	Odziv na obraz	Odziv na trk	Skupaj
Pravilno	33	8	7	13	2	63
Napačno	0	3	0	0	0	3
Skupaj	33	11	7	13	2	66

Tabela 4.10: Primeri odzivov za četrto postavitev poligona.

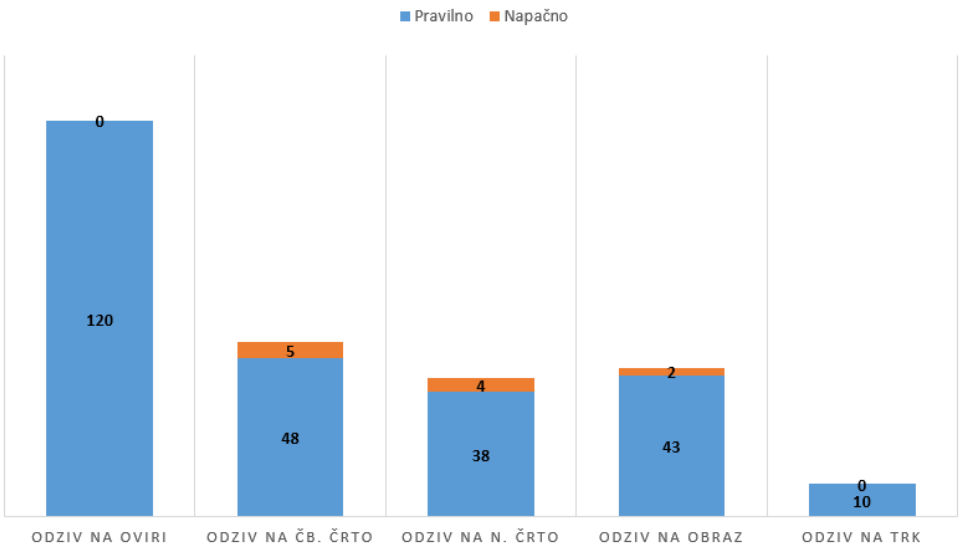
4.3.2 Sklep

Iz **rezultatov celotne evalvacije** smo ugotovili in zabeležili v tabeli 4.11, da v sistemu najbolje deluje metoda odziva na oviro. To metodo smo tudi največkrat zabeležili in ni imela niti enega napačnega primera odziva glede na vse skupne rezultate. Opazili smo, da ko se odziv na oviro že izvaja lahko pride po ponovnega izvajanja odziva na oviro, če sta oviri preblizu. To lahko sproži obračanje robotske enote na mestu, tako da sistem potrebuje več poskusov da se robotska enota premakne iz mest. Razlog je verjetno to, da ima v sistemu metoda za odziv na ovire samo en način obračanja. Ugotovili smo tudi, da se robotska enota po končanem sistemskem odzivu na oviro premakne na mesto pred črno-bele črto med koti 0° in 30°, zato je ne zazna.
Na trk se je sistem v vseh primerih pravilno odzival in se ni niti enkrat izvajal kot posledica ne zaznavanja ovir na pravi višini. Ta odziv na trk se je pojavil samo na označeni lokaciji poligona. Opazili smo, da se je lahko ob izvajanju odziva na trk robotska enota premaknila čez črto ali trčila v oviro. Razlog je, da se robotska enota ob sprožitvi sistemskega odziva na trk začne najprej premikati vzvratno, kar pomeni da ne vidi, ali se ovire nahajajo za robotsko enoto. To bi lahko izboljšali, tako da bi pomanjšali razdaljo premikanja nazaj. V naših poskusih se to ni pojavilo, saj ovire niso bile preblizu.
Skupno število primerov odziva na črno-belo in navadno črto se je izvedlo manjkrat kot odziv na oviro, kar je razvidno iz grafa 4.17. To pomeni, da je manj primerov, ampak smo videli, da ima skupaj samo 9 primerov napačnih odzivov glede na vseh 95 primerov.

	1-Odziv na oviri	2-Odziv na ČB. črto	3-Odziv na n. črto	4-Odziv na obraz	5-Odziv na trk	Skupaj
Pravilno	120	48	38	43	10	259
Napačno	0	5	4	2	0	11
Skupaj	120	53	42	45	10	271

Tabela 4.11: Skupno število vseh primerov odzivov za vse poligone.

Ugotovili smo tudi, da ko se robotska enota premika proti črti pod ozkim kotom ter malo razdaljo, med njima lahko pride do ne zaznavanja črte. Za navadno črto pride tudi do napačnih odzivov, saj kot smo v prejšnji evalvaciji omenili, lahko pod napačnim kotom napačno zazna črto kot črno-belo, kjer so senčena tla ob beli črti, pa jo bo zaznala kot črno-belo črto.
Metoda za odziv na obraze se je zelo dobro obnesla, saj ima samo dva napačna primera odziva. Po popravljeni implementaciji metode smo opazili, da se metoda na predmete ni več napačno odzvala ali jih zamenjala za obraze, kot v prvih dveh postavitvah poligona. Ugotovili smo tudi, da ko se robotska enota premika proti obrazu pod ostrim kotom, ga sistem ne more zaznati, kar smo tudi ugotovili v evalvaciji metode za zaznavanja in odziva na obraz. Sistem obraze pod primernimi koti uspešno zazna, vendar se na koncu, pri približevanju obrazu, lahko prehitro premakne naprej in sproži odziv na oviro, preden sploh lahko izvede pozdrav. Sistem bi lahko izboljšali tako, da povečamo razdaljo, ki jo robotska enota potrebuje za pozdrav obraza.



Slika 4.17: Vsi primiri odzivov za vse postavitve poligona.

Poglavje 5

Zaključek

Cilj diplomske naloge je bilo implementirati avtonomni robotski sistem, ki bi zaznal obraze in bi jih tudi pozdravil. Premikal naj bi se avtonomno, ne da bi pri tem beležil podatke prostora, v katerem se je nahajal. Pri tem je bil omejen s črto na tleh, ki je predstavljala zaprt prostor, ki ga ne sme zapustiti. Robotska enota bi se morala izogibati oviram in pravilno odreagirati na trke. Tem zahtevam smo zadostili s čimboljšimi implementiranimi metodami. Seveda so se pojavile tudi nekatere napake, kar smo videli v evalvaciji sistema.

Za diplomsko nalogo smo implementirali metode zaznavanja in odziva na črto in metodo zaznavanja in odziva na obraz. S tem smo želeli rešiti oba glavna problema diplomske naloge. Pri tem smo se srečali s problemi računalniškega vida ter spoznali algoritem Houghove transformacije za zaznavanje črt in algoritem AdaBoost za zaznavanja obrazov. Implementirali smo še metode, ki so omogočale premikanje in obračanje same robotske enote. Za varno delovanje robotske enote smo implementirali še metodo za odziv na ovire in metodo za odziv na trk ali previs.

Pri evalvaciji smo odkrili napake, ki smo jih nato skušali odpraviti. Nekaterih napak nismo popolnoma odpravili, saj bi odpravljanje le-teh zahtevalo veliko število poskusov za natančno določanje popravkov. Za preverjanje delovanja

sistema v različnih okoljih, bi ga morali preizkusiti na različnih tleh. V evalvaciji sistema smo opazili, da odziva na ovire in trke zelo dobro delujeta. Za metodo odziva na črte smo odkrili, da ima mrtvi kot, ki se začne med 0°in 30°, glede na oddaljenost pa se odziv izboljša, ker ima več časa za zaznavo črte. Pri tej metodi smo opazili, da je treba povečati preverjanje števila slikovnih točk za boljše zaznavanje črt ter povečati preverjanje primerjave med barvami tal in črt, da zmanjšamo možnost zaznavanja navadne črte kot črno-belo črto. Metoda za obraz je delovala dobro in glede na kot zaznave se je takoj pravilno odzvala.

Iz evalvacije lahko sklepamo, da metode in celoten sistem zadoščata zahtevam postavljenega cilja.
Kot vsi sistemi, je naš sistem, ki smo ga naredili za robotsko enoto, mogoče še izboljšati in razširiti z novimi metodami. Metodo odziva ovir bi se lahko nadgradilo z zaznavanjem ovir na različnih višinah. Implementirali bi lahko še naključno obračanje robotske enote za zmanjšanje verjetnosti premikanja po isti poti. Nadgradili bi lahko tudi metodo zaznavanja in odziva na obraz, tako da bi metoda zaznala celotno človeško telo, in implementirali sposobnost sledenja človeka.

Literatura

[1] Matas, J. and Galambos, C. and Kittler, J.V., "Robust Detection of Lines Using the Progressive Probabilistic Hough Transform. CVIU 78 1, pp 119-137 (2000)

[2] J. Canny. A Computational Approach to Edge Detection, IEEE Trans. on Pattern Analysis and Machine Intelligence, 8(6), pp. 679-698 (1986).

[3] Viola, Paul; Jones, Robert (2001). "Rapid Object Detection Using a Boosted Cascade of Simple Features".

[4] Rainer Lienhart and Jochen Maydt. An Extended Set of Haar-like Features for Rapid Object Detection. IEEE ICIP 2002,

[5] Tutorial py face detection. [Online]. Dosegljivo: http://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html. [Dostopano 31. 8. 2016].

[6] Nocks, Lisa (2007). The robot : the life story of a technology. Westport, CT: Greenwood Publishing Group.

[7] OpenCV doc - uradna dokumentacija. [Online]. Dosegljivo: http://docs.opencv.org/2.4/index.html. [Dostopano 31. 8. 2016].

[8] ROS wiki - uradna dokumentacija. [Online]. Dosegljivo: http://wiki.ros.org/. [Dostopano 31. 8. 2016].

[9] Kinect wiki. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Kinect. [Dostopano 31. 8. 2016].

[10] Roomba wiki. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Roomba. [Dostopano 31. 8. 2016].

[11] Roomba. [Online]. Dosegljivo: https://www.amazon.com/iRobot-530-Roomba-Vacuuming-Robot/dp/B000UU7TZE.[Dostopano 31. 8. 2016].

[12] Robotika. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Robotics. [Dostopano 31. 8. 2016].

[13] Sound play. [Online.] Dosegljivo: http://wiki.ros.org/sound_play. [Dostopano 31. 8. 2016].

[14] Python. [Online]. Dosegljivo: https://www.python.org/. [Dostopano 31. 8. 2016].

[15] Ubuntu. [Online]. Dosegljivo: http://www.ubuntu.com/. [Dostopano 31. 8. 2016].

[16] Numpy. [Online]. Dosegljivo: http://www.numpy.org./. [Dostopano 31. 8. 2016].

[17] Gazebosim. [Online]. Dosegljivo: http://gazebosim.org/. [Dostopano 31. 8. 2016].

[18] ASIMO. [Online]. Dosegljivo: http://world.honda.com/ASIMO/history/. [Dostopano 31. 8. 2016].

[19] Haar-algorithm-in-face-detection. [Online]. Dosegljivo: http://visionandarm.blogspot.si/2014/02/4-haar-algorithm-in-face-detection.html. [Dostopano 31. 8. 2016].

[20] linear hough transform using python. [Online]. Dosegljivo: https://nabinsharma.wordpress.com/2012/12/26/linear-hough-transform-using-python/. [Dostopano 31. 8. 2016].

[21] Roomba priročnik.[Online]. Dosegljivo: http://homesupport.irobot.com/euf/assets/images/faqs/roomba/500/manual/en-US.pdf. [Dostopano 31. 8. 2016].

[22] Turtelbot[Online]. Dosegljivo: http://www.turtlebot.com/ [Dostopano 31. 8. 2016].