Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Domen Požrl

Križci in krožci z robotom

diplomsko delo

visokošolski strokovni študijski program
prve stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Danijel Skočaj

Ljubljana, 2018

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V diplomski nalogi razvijte senzorsko-robotski sistem, ki bo s človekom odigral igro Križci in krožci. Sistem naj s kamero opazuje igralno površino, zaznava poteze nasprotnika in izračuna optimalne poteze, ki naj jih s pomočjo nizkocenovnega lahkega robotskega manipulatorja tudi odigra.

Zahvaljujem se mentorju izr. prof. dr. Danijelu Skočaju za strokovne nasvete ter usmerjanje pri izdelavi diplomske naloge. Zahvala gre tudi doc. dr. Luki Čehovin Zajcu za vso pomoč pri implementaciji naloge in as. Anžetu Rezlju, za pomoč pri 3D tiskanju. Na koncu bi se rad zahvalil še staršem in bratu za vso podporo tekom študija.

Kazalo

Povzetek

Naslov: Križci in krožci z robotom

Avtor: Domen Požrl

Intenzivna prisotnost robotov v našem vsakdanjem življenju mnogim ljudem povzroča skrbi. Izkazalo se je, da so zaskrbljeni predvsem tisti, ki so do sedaj imeli malo stika z robotiko. Cilj te diplomske naloge je razviti sistem, ki robotskemu manipulatorju omogoča igranje popolne igre Križcev in krožcev. S pomočjo algoritma za optimalno izbiranje potez pri igri Križci in krožci in z uporabo različnih metod računalniškega vida ter robotike, smo razvili sistem, ki uporabnikom preko igre, ki jo poznajo že od otroštva, predstavi interakcijo med roboti in ljudmi.

Ključne besede: Križci in krožci, računalniški vid, robotski manipulator.

Abstract

Title: TicTacToe with a robot

Author: Domen Požrl

Many people are worried about the intense presence of robots in our everyday lives. It turns out that the people, who are worried, are also the same people who never had much contact with robotics. The goal of this bachelor thesis is to develop a system that enables a robot manipulator to play a perfect game of TicTacToe. With the help of an algorithm for optimal decision making in the game of TicTacToe and using various methods of computer vision and robotics, we have developed a system, which introduces its users to robot-human interaction using a game they know since their childhood.
Keywords: TicTacToe, computer vision, robot manipulator.

Poglavje 1

Uvod

1.1 Motivacija

Roboti so del našega življenja. Nekateri sesajo naša stanovanja in kosijo travo, spet drugi pa nam skuhajo izvrstno kavo. Prvi morda malo bolj ustrezajo definiciji robota kot drugi, vendar pa se vsekakor lahko strinjamo, da je v zadnjih letih tehnologija, namenjena olajšanju človeškega življenja, močno napredovala. V 70-ih letih prejšnjega stoletja je zaradi razvoja robotskih manipulatorjev še bolj intenzivno rast izkusila avtomobilska industrija. Danes se večina proizvajalcev avtomobilov zanaša na avtomatizacijo, ki jo omogočajo robotski manipulatorji [13].

Intenzivnejša prisotnost robotov v ljudeh vzbuja skrb zaradi vpliva, ki bi ga roboti imeli na družbo. Raziskave kažejo, da pogosteje skrbi ljudi, ki imajo malo stika z robotiko [2]. Cilj te diplomske naloge je razviti sistem, ki s pomočjo robota odigra popolno igro Križcev in krožcev. Menimo, da bi s takim izdelkom lahko ljudem predstavili interakcijo med robotom in človekom, ter jim približali robotiko.

1.2 Pregled področja

Tema, s katero se ukvarjajo vodilna podjetja na področju računalništva, so potencialne težave z robotiko in umetno inteligenco [1]. Velik problem predstavlja robotovo razumevanje okolja, v katerem se nahaja in robotova reakcija na situacije, v katerih se znajde prvič. Soglasni so si v mnenju, da rešitev ni ena sama, vendar se bo razvila vzporedno z napredki na tem področju. Predvidevajo, da se bodo med razvojem manjših projektov razvile manjše rešitve, ki se bodo kasneje razširile in aplicirale na splošne probleme.

1.2.1 Računalnik, ki je premagal svetovnega prvaka v šahu

Deep Blue je bil računalnik namenjen igranju šaha. Razvilo ga je podjetje IBM in je znan po tem, da je prvi računalnik, ki je premagal svetovnega prvaka v šahu. K zmagi je pripomoglo število različnih dejavnikov, pomembnejši od katerih so visoka raven paraleliziranosti sistema, kompleksna funkcija ovrednotenja potez in učinkovita uporaba podatkovne baze Grandmaster game. Deep Blue je en izmed prvih sistemov, ki v sklopu interakcije med računalnikom in človekom učinkovito izkoristi priložnost paraleliziranja izračunov [3].

Slika 1.1: Računalnik Deep Blue

1.2.2 Robot, ki igra igro Naseljenci otoka Catan

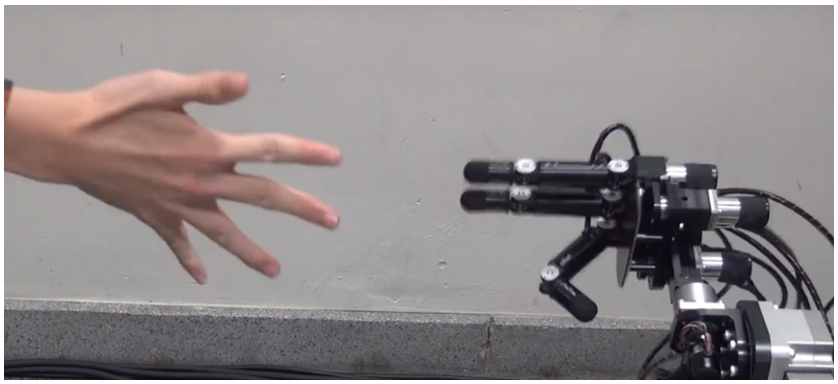
Študentje in profesorji šole OTH Regensburg so v sklopu predmeta, pri katerem spoznavajo naprednejše koncepte robotike, razvili sistem, ki igra igro Naseljenci otoka Catan [8, 7]. Za premikanje figuric so uporabili robotski manipulator podjetja KUKA, za igralno površino pa velik LCD zaslon. Najprej so razvili spletni vmesnik za igranje igre, na katerem so igrali človeški igralci. Podatke vseh iger so shranjevali in jih nato uporabili za učenje robota, ki sedaj igro igra na zelo visokem nivoju.

Slika 1.2: KUKA, ki igra igro Naseljenci otoka Catan

1.2.3 Robot, ki vedno zmaga v igri Kamen, škarje, list

V laboratoriju Ishikawa Watanabe v Tokiu na Japonskem so raziskovalci razvili robota, ki nikoli ne izgubi v igri Kamen, škarje, list. Najnovejša različica takega robota lahko v zelo kratkem času prepozna katerega koli od treh značilnih gibov človeške roke in nanj ustrezno reagira. Celoten proces prepoznavanja giba in odločanja o ustrezni reakciji robot izvede tako hitro, da svoj gib zaključi prej kot človek. Celoten sistem temeli na kameri, ki omogoča hitro zaznavanje katerega od treh značilnih gibov je izbral človek. Nato sistem sporoči trem robotskim "prstom", kateri gib naj izvedejo, da

premagajo človeka [6].



Slika 1.3: Robot, ki vedno zmaga v igri Kamen, škarje, list

1.3 Zgradba diplomske naloge

Za konec uvoda podajamo še opis vsebinske strukture diplomske naloge:

1. Predstavitev sistema
   V poglavju na kratko predstavimo glavne programske in strojne komponente sistema.

2. Algoritem za igranje popolne igre Križci in krožci
   Poglavje podrobno opiše algoritem za igranje popolne igre Križci in krožci.

3. Zaznavanje sprememb na igralni površini
   V tem poglavju predstavimo metode računalniškega vida, ki smo jih uporabili za zaznavanje figur ter igralne površine.

4. Upravljanje manipulatorja za izvedbo poteze
   V tem delu diplomske naloge opišemo postopek transformacije centroidov zaznanih objektov in upravljanje robotskega manipulatorja.

5. Vrednotenje sistema
   V poglavju so podani rezultati opravljenih meritev, ki vrednotijo izdelan sistem.

6. Zaključek
   V zaključku opišemo še možne izboljšave in alternativne pristope k implementaciji sistema. Podamo tudi sklepno misel.

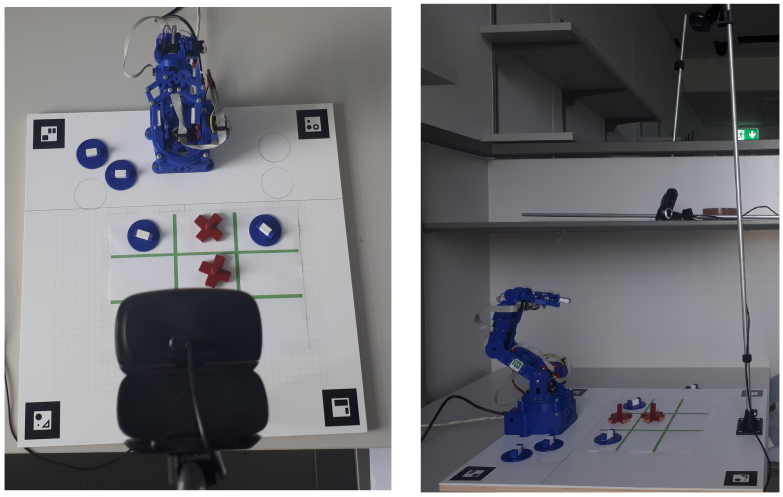
Poglavje 2

Predstavitev sistema

Diplomsko delo je razdeljeno na tri glavne dele:

- algoritem za igranje popolne igre Križcev in krožcev,
- zaznavanje spremembe igralne površine (računalniški vid),
  - Kamera Logitech HD Webcam C525
  - OpenCV
- upravljanje manipulatorja za izvedbo poteze.
  - Nizkocenovni lahek robotski manipulator
  - Manus

Implementirali smo jih s pomočjo različne programske (OpenCV, Manus) in strojne opreme (kamera in robotski manipulator), ki jo na kratko predstavimo v tem poglavju. Na koncu poglavja podamo še splošen pregled delovanja sistema.



Slika 2.1: Sliki celotnega sistema

2.1 Oprema za zaznavanje sprememb na igralni površini

2.1.1 Kamera

Kamera, ki smo jo uporabljali za izvajanje zaznavanja, je ključni del celotnega sistema. Robotski manipulator uporablja spletno kamero podjetja Logitech model HD Webcam C525. Povezava kamere z robotskim manipulatorjem ponuja sliko ločljivosti 720p. Vsebuje tudi mehanizem za avtomatsko ostrenje slike [9]. Kamera je dovolj zmogljiva za uporabo pri osnovnih in kompleksnejših aplikacijah računalniškega vida.

2.1.2 OpenCV

OpenCV je zelo razširjena programska knjižnica za delo s slikami. Uporabljamo jo lahko v kombinaciji z različnimi programerskimi jeziki. V tem diplomskem delu jo uporabljamo s Pythonom. Knjižnica je napisana v programskem jeziku C++ in vsebuje preko 2500 optimiziranih algoritmov za računalniški

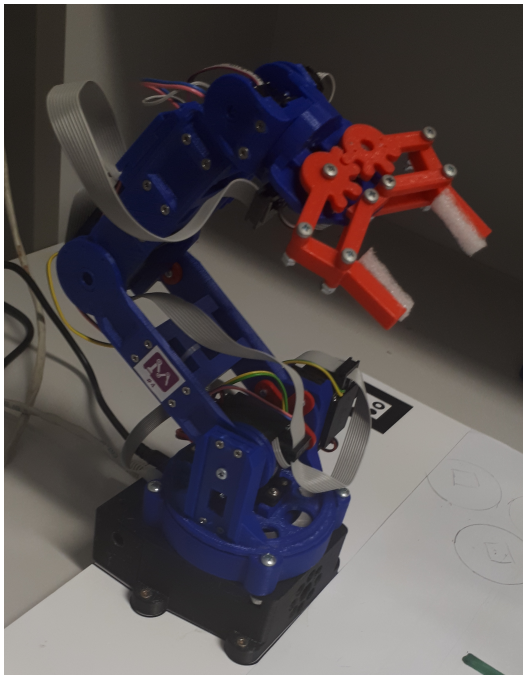
Slika 2.2: Logitech HD Webcam C525

vid in strojno učenje [11].

2.2 Oprema za premikanje figur

2.2.1 Nizkocenovni lahek robotski manipulator

Robotski manipulator, ki smo ga uporabili za implementacijo diplomske naloge, je razvil Anže Rezelj v svoji magistrski nalogi z naslovom Razvoj nizkocenovnega lahkega robotskega manipulatorja [12, 15]. Cilj magistrskega dela je bil razviti robotski manipulator, ki je cenovno ugoden, vendar vseeno dovolj natančen za poučevanje konceptov robotike. Ogrodje manipulatorja so natisnili s 3D tiskalnikom ter jih med seboj povezali tako, da ima vsak sklep svoj servomotor. Manipulator ima 5 prostostnih stopenj in teoretičen delovni prostor v obliki popačene krogle. Za podrobnejše informacije o robotskem manipulatorju priporočamo, da preberete prej omenjeno magistrsko delo.



Slika 2.3: Nizkocenovni lahek robotski manipulator

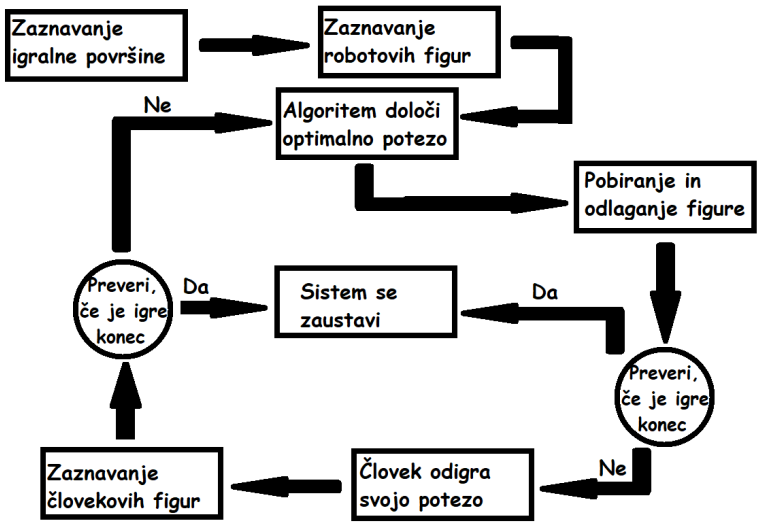
2.2.2 Manus

Projekt Manus je ogrodje za programsko krmiljenje robotskega manipulatorja. Razvit je bil v Laboratoriju za umetne vizualne spoznavne sisteme in se uporablja v izobraževalne namene v sklopu predmetov o robotiki in računalniškem vidu na Fakulteti za računalništvo in informatiko Univerza v Ljubljani. S pomočjo knjižnic, ki so bile razvite v istem laboratoriju (EchoLib, EchoCV, EchoMsg), in javno dostopnih knjižnic (OpenCV, Orcos KDL) omogoča programsko zajemanje slik s kamero, transformacijo točk iz koordinatnega sistema kamere v koordinatni sistem sveta in krmiljenje robotskega manipulatorja s pomočjo inverzne kinematike [10, 15]. Seveda ima ogrodje tudi druge funkcionalnosti, vendar smo v tem diplomskem delu uporabili le te.

2.3 Splošen pregled delovanja sistema

V kasnejših poglavjih diplomskega dela podrobneje predstavimo delovanje in implementacijo posameznih komponent sistema. Za lažjo predstavo in razumevanje pa najprej podamo splošen pregled delovanja sistema.

Najprej sistem za zaznavanje sprememb zazna igralno površino ter preračuna obsegajoče pravokotnike. Sledi zaznavanje robotovih figur in izračun centroidov. Naslednji korak je del zanke, ki se ponavlja, dokler se igra ne zaključi (slika 2.4). Algoritem za igranje popolne igre Križcev in krožcev določi optimalno polje, ki ga mora sistem igrati. Izvede se pobiranje robotove figure in odlaganje na izbrano polje. Nato algoritem zaustavi sistem, če se je igra z zadnjo potezo zaključila. V kolikor igre še ni konec, sistem počaka, da človek izvede svojo potezo. Nato se izvede zaznavanje človekovih figur in določanje, v katero polje je človek igral. Algoritem nato ponovno preveri, če se je igra zaključila. V primeru, da se ni, se ponovno začne zanka z izbiro optimalne poteze.



Slika 2.4: Shema celotnega delovanja sistema

Poglavje 3

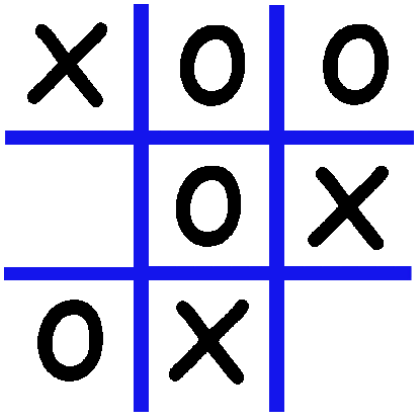
Algoritem za igranje popolne igre Križci in krožci

3.1 Igra Križci in krožci

V tem diplomskem delu smo implementirali sistem, ki odigra popolno igro Križcev in krožcev. Križci in krožci je igra za 2 igralca. Igra se na igralni površini z 9 prostimi polji v obliki tabele 3x3. Igralca izmenično postavljata znaka X in O na prosta polja igralne površine. Zmaga tisti, ki svoje znake postavi tako, da zasedajo vsa polja v eni od vrstic, stolpcev ali diagonal. Za lažje razumevanje bomo v nadaljevanju zmago v igri Križci in krožci označili s frazo "3 v vrsto". Primer odigrane igre Križci in krožci je prikazan na sliki 3.1.

3.2 Izbira algoritma

Prvi del sistema, ki igra popolno igro Križci in krožci, je seveda strategija, ki omogoča izbiro optimalnih potez oziroma nasprotniku onemogoča zmago. Algoritem Minimax je bil zasnovan prav za igre z dvema izmenjujočima igralcema in predpostavlja, da so za nas dobra končna stanja igre ovrednotena z nekim pozitivnim številom točk, slaba končna stanja igre pa z negativnim.



Slika 3.1: Primer zaključene igre Križcev in krožcev. Zmagal je igralec z znakom O.

Algoritem generira drevo potez, ki nas pripeljejo v končno stanje, ki ima največ točk glede na vsa ostala razpoložljiva stanja [14].

Cilj te naloge je ljudem približati robotiko, zato se nismo odločili za algoritem Minimax, ki je kompleksnejši algoritem iz teorije iger, ampak za enostavnejši algoritem, ki sta ga predlagala Kevin Crowley in Robert S. Siegler v svojem članku z naslovom Flexible Strategy Use in Young Children's Tic-Tac-Toe [4]. Algoritem ne temelji na predčasni generaciji vseh možnih stanj igralne površine, vendar na posamičnem odločanju, ki je veliko bolj podobno človeškemu razmišljanju. V nadaljevanju predstavimo postopek odločanja.

3.3 Predstavitev algoritma

Osrednja ideja uporabljenega algoritma je definicija vseh različnih potez, ki jih lahko igramo v igri Križci in krožci, ter razvrstitev le teh v hierarhijo igranja. Vsakič, ko algoritem prejme stanje igralne površine in mora izvesti optimalno potezo, se sprehodi po hierarhiji potez ter igra prvo, ki jo lahko [4]. V tem poglavju bomo predstavili vsako od potez, najprej pa podajamo splošen pregled hierarhije:

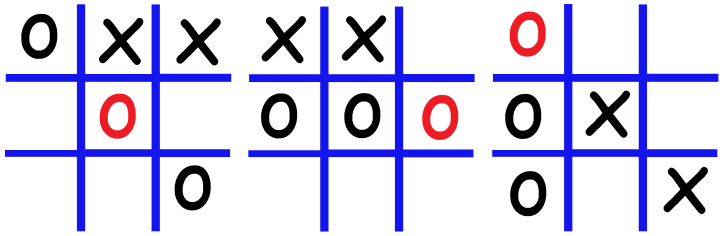
1. zmagaj (ang. win) - igraj potezo, ki igralcu prinese zmago.
2. blokiraj (ang. block) - igraj potezo, ki nasprotnemu igralcu prepreči zmago.
3. razcep (ang. fork) - igraj potezo, ki igralcu ustvari dve priložnosti za zmago.
4. blokiraj razcep (ang. block fork) - igraj potezo, ki nasprotnemu igralcu prepreči igrati potezo razcep.
5. sredina (ang. center) - igraj na sredino, če je prva poteza igre, igraj v prazni kot.
6. nasprotni kot (ang. opposite corner) - igraj kot, ki je nasproti tistemu, ki ga je igral nasprotnik.
7. prazen kot (ang. empty conrner) - igraj v prazni kot.
8. prazna stranica (ang. empty side) - igraj na prazno stranico.

V zgornjem seznamu je najpomembnejša poteza označena z 1, najmanj pomembna pa z 8. Algoritem postopek odločanja začne pri najpomembnejši potezi - potezi zmagaj. Če poteze zmagaj zaradi trenutnega stanja igralne površine ne moremo izvesti, se pomakne na manj pomembno potezo - potezo blokiraj, in spet preveri, ali jo lahko izvede. Algoritem ponavlja postopek preverjanja izvedljivosti potez, dokler ne pride do prve poteze, ki jo lahko izvede. Le ta predstavlja optimalno potezo glede na trenutno stanje igralne površine.

Algoritem smo implementirali v programskem jeziku Python. Zasnovali smo razred, ki v matriki 3x3 hrani vse spremembe igralne površine in poseduje metode, ki preverijo razpoložljivost vseh zgoraj omenjenih potez. Razred ima tudi metodo, ki glede na hierarhijo preveri, katero potezo je potrebno igrati.

3.3.1 Zmagaj

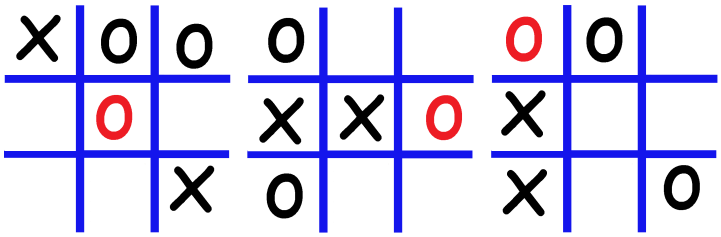
Poteza zmagaj je najpomembnejša poteza v algoritmu. Kadarkoli imamo priložnost za zmago jo moramo izkoristiti in z njo zaključiti igro. Metoda, ki preverja razpoložljivost poteze zmagaj pregleda, če ima igralec, ki je trenutno na vrsti za igranje, v katerem od stolpcev, vrstic ali pa diagonal že postavljena 2 v vrsto. Nato preveri ali je polje, s katerim bi dosegli zmago, prosto. V primeru, da v trenutnem stanju igralne površine ni priložnosti za zmago, algoritem zaključi iskanje zmage in nadaljuje na naslednjem koraku.



Slika 3.2: Primer različnih potez zmagaj. Igran znak je obarvan rdeče.

3.3.2 Blokiraj

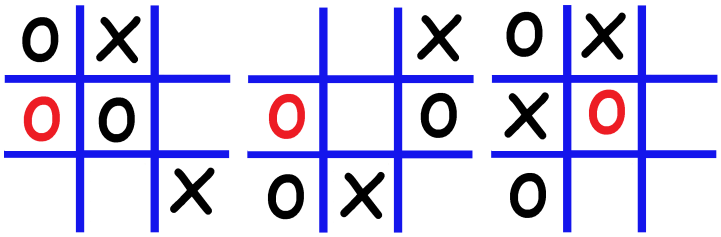
V primeru, da na igralni površini ni priložnosti za zmago, preverimo, ali imamo priložnost za blok. V tem primeru uporabimo enako metodo, kot pri iskanju razpoložljivosti poteze zmagaj. Tokrat, namesto da iščemo našo zmago, iščemo zmago nasprotnika. V primeru, da ima nasprotnik priložnost za zmago, mu jo blokiramo tako, da igramo naš znak na njegovo zmagovalno polje.



Slika 3.3: Primer različnih potez blokiraj. Igran znak je obarvan rdeče.

3.3.3 Razcep

Razcep je tretji po vrsti v hierarhiji potez. Namen te poteze ni takojšnja zmaga, ampak zmaga v naslednji naši potezi. Znake postavimo tako, da imamo naslednjič, ko smo na vrsti, dve priložnosti za zmago. Po taki potezi je nasprotnik tako rekoč že premagan, saj ne more blokirati dveh mest hkrati. Ne glede na to, katero polje blokira nasprotnik, lahko mi igramo na drugo in s tem zmagamo igro. Razpoložljivost poteze razcep preverimo tako, da navidezno igramo vsako možno potezo in nato preverimo, če ta poteza omogoča dve različni zmagi.

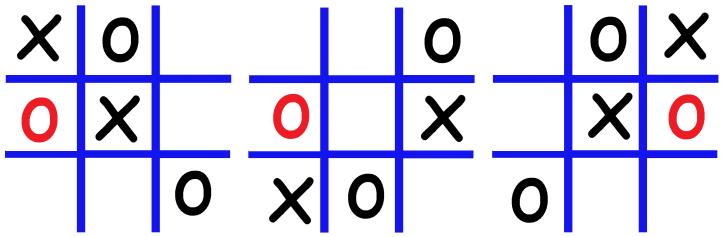


Slika 3.4: Primer različnih potez razcep. Igran znak je obarvan rdeče.

3.3.4 Blokiraj razcep

Ta poteza deluje po enakem principu kot poteza blokiraj. Ponovno poskusimo nasprotniku preprečiti, da igra potezo razcep. To storimo tako, da poiščemo

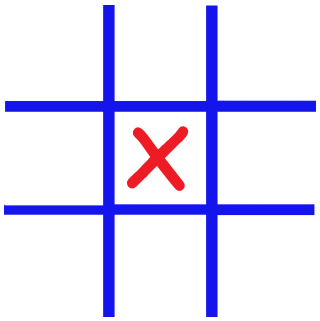
vsa polja, na katera lahko nasprotnik igra potezo razcep in jih zasedemo z našim znakom.



Slika 3.5: Primer različnih potez blokiraj razcep. Igran znak je obarvan rdeče.

3.3.5 Sredina

V primeru, da je trenutna poteza prva poteza igre, se poteza sredina ne izvede. Namesto nje se izvede poteza prazen kot. V vseh ostalih primerih pa poteza sredina preprosto pomeni igraj znak na sredino. Metoda, ki preverja razpoložljivost poteze, preveri, če je sredinsko polje prosto.

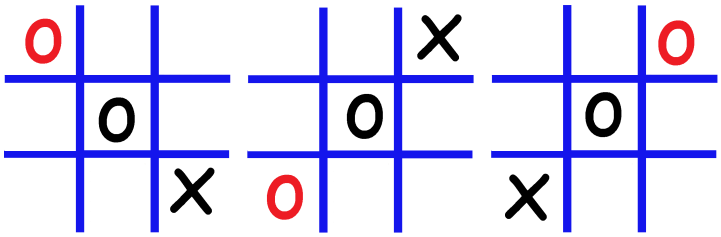


Slika 3.6: Primer poteze sredina. Igran znak je obarvan rdeče.

3.3.6 Nasprotni kot

Nasprotni kot se igra v primeru, ko je naš nasprotnik igral v kot in je sredina že zasedena (in seveda, če ni možnosti zmage, bloka ali pa razcepa). Poteza,

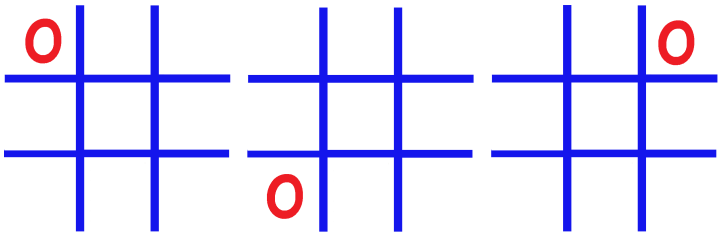
kot nam pove že ime, igra kot, ki je nasproti tistemu, katerega je igral naš nasprotnik.



Slika 3.7: Primer poteze nasprotni kot. Igran znak je obarvan rdeče.

3.3.7 Prazen kot

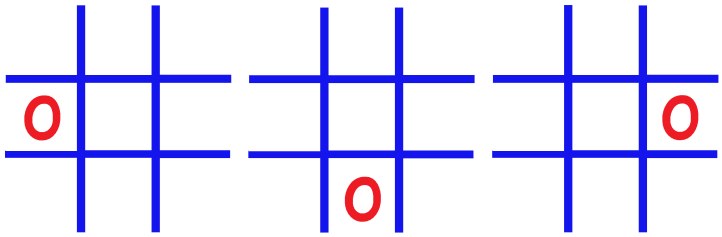
Poteza prazen kot se običajno igra na začetku ali pa koncu igre. Pomeni igrati znak v enega izmed kotov, ki so še prosti.



Slika 3.8: Primer poteze prazen kot. Igran znak je obarvan rdeče.

3.3.8 Prazna stranica

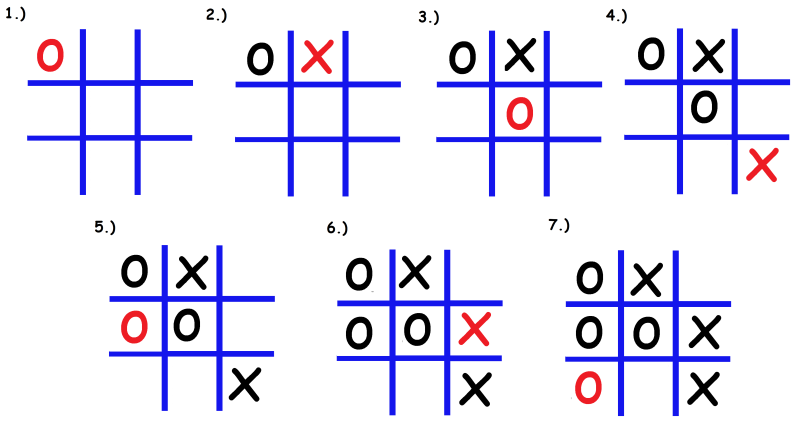
Poteza prazna stranica je zadnja v hierarhiji. Ko nobena od potez ni izvedljiva, igramo znak na prazno stranico.



Slika 3.9: Primer poteze prazna stranica. Igran znak je obarvan rdeče.

3.4 Primer odigrane popolne igre

V tem razdelku si bomo pogledali še primer igre odigrane z zgoraj opisanim algoritmom. Znak O pripada sistemu, ki uporablja algoritem, znak X pa človeku. Začel bo O. Potek igre je prikazan v sliki 3.9.



Slika 3.10: Primer odigrane igre s pomočjo algoritma

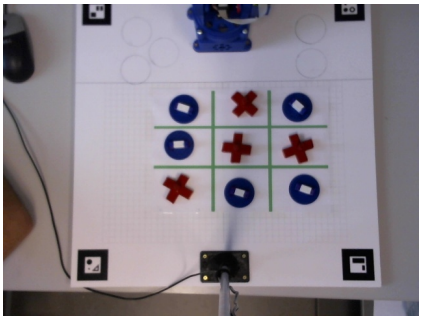
Prva razpoložljiva poteza je sredina, vendar je to prva poteza v igri, zato algoritem igra prazen kot. Človek igra zgornjo stranico. Ponovno je prva razpoložjiva poteza sredina. Tokrat jo algoritem igra. Človek nato blokira 3 v vrsto. Prva razpoložljiva poteza je sedaj razcep. Človek blokira eno od dveh potencialnih zmag. Algoritem igra potezo zmagaj in tako igro zaključi.

Poglavje 4

Zaznavanje sprememb na igralni površini

Po implementirani strategiji za popolno igro Križcev in krožcev smo razvili sistem, ki skrbi za komunikacijo med fizičnim svetom in algoritmom. Sistem zazna spremembe stanja igralne plošče v fizičnem svetu in jih prevede v informacije, ki so primerne za obdelavo. Na sliki 4.1 je primer stanja igralne površine.

Za pravilno delovanje celotnega sistema smo problem zaznavanja razdelili na zaznavanje robotovih figur, zaznavanje človekovih figur in zaznavanje igralne površine.



Slika 4.1: Stanje igralne površine.

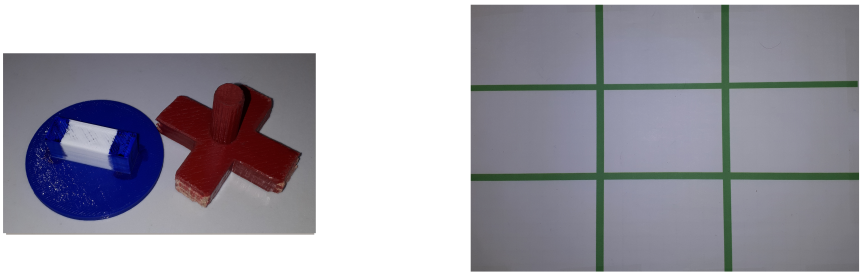
4.1 Zaznavanje igralnih figur

Igra Križci in krožci temelji na znakih X in O. V izvedbi igre za dve osebi se znake in igralno površino nariše na prazen list papirja. V naši izvedbi igre za osebo in robota pa se namesto risanja znakov uporabljajo igralne figure. Naloga robota je, da zazna, na katero polje igralne površine je svojo figuro postavil človek, ter nato na optimalno izbrano polje postavi svojo figuro. V tem poglavju se osredotočimo na zaznavanje igralnih figur, premikanje figur pa smo opisali kasneje.

4.1.1 Igralne figure in igralna površina

Igralne figure robota so izdelane s pomočjo 3D tiskanja, igralne figure človeka pa so izdelane iz lesa. Sestavljene so iz ploščic, ki so oblikovane v obliko X ali O, in ročk, ki so pravokotno prilepljene na ploščice. Figure O so namenjene robotu, zato so ročke v obliki kvadra, ki omogoča lažje pobiranje. Za lažje zaznavanje in ločevanje figur smo figure O pobarvali modro, figure X pa rdeče.

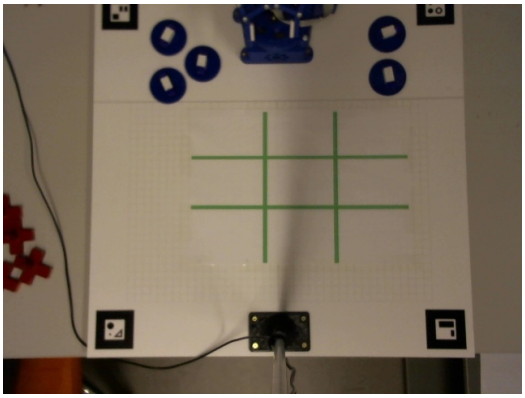
Igralna površina je natisnjena. Uporabili smo zeleno barvo za povečanje razlike med igralno površino in igralnimi figurami.



Slika 4.2: Igralni figuri in igralna površina

4.1.2 Zaznavanje robotovih figur

Zaznavanje robotovih figur se izvede na začetku igre ter po vsakem pobiranju. Preden robot odigra svojo prvo potezo, zazna vse figure ob igralni površini in preračuna središča detekcij.



Slika 4.3: Slika pred obdelavo

Zaznavanje robotovih figur se izvaja v 4 korakih:

1. *Preslikava slike iz RGB v normaliziran RGB.*
   Preslikava slike iz RGB v normaliziran RGB prostor nam omogoča lažje ločevanje objektov na sliki glede na osnovne barve RGB barvnega prostora. V primeru, ko poskušamo zaznati objekte modre barve, bi z uporabo klasičnega RGB barvnega prostora in upragovanja po komponenti B za objekt določili slikovne elemente, ki vsebujejo barve z visoko vsebnostjo modre. Preslikava v normaliziran RGB prostor vrednost vsake komponente v slikovnem elementu deli z vsoto vrednosti vseh komponent in tako normalizira njihovo intenzivnost. Sedaj z upragovanjem po komponenti B za objekt določimo samo tiste slikovne elemente, ki so modre barve. Celotno preslikavo lahko zapišemo s spodnjimi enačbami.

$$f(x, y) = (R, G, B) \tag{4.1}$$

$$total = (R + G + B) \tag{4.2}$$

$$R' = \frac{R}{total} \times 255 \tag{4.3}$$

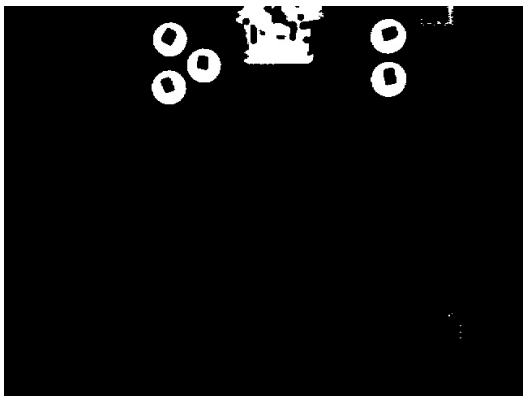
$$G' = \frac{G}{total} \times 255 \tag{4.4}$$

$$B' = \frac{B}{total} \times 255 \tag{4.5}$$

$$g(x, y) = (R', G', B') \tag{4.6}$$

2. *Upragovanje glede na komponento B.*
   Iz slike v normaliziranem RGB prostoru izberemo samo vrednosti slikovnih elementov za komponento B. Tako ustvarimo sivinsko reprezentacijo modre barve na sliki, ki jo sedaj lahko upragujemo. Upragovanje je proces, ki sivinsko sliko spremeni v binarno tako, da vrednosti slikovnih elementov, ki so pod izbranim pragom, spremeni v črne. Tiste vrednosti slikovnih elementov, ki so nad izbranim pragom, spremeni v bele. Spodnja enačba opisuje postopek upragovanja.

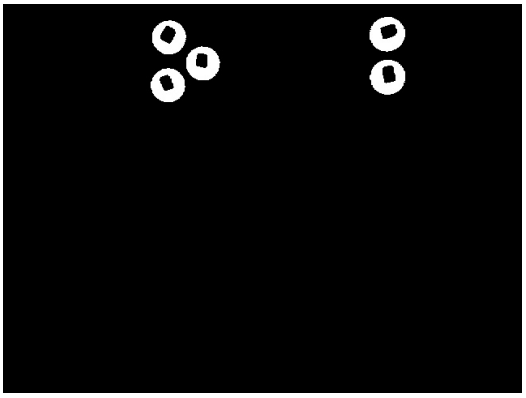
$$f_{threshold}(a) = \begin{cases} a_0 & \text{for } a < a_{th} \\ a_1 & \text{for } a \geqslant a_{th} \end{cases}$$



Slika 4.4: Upragovljena slika igralne površine po barvni komponenti B

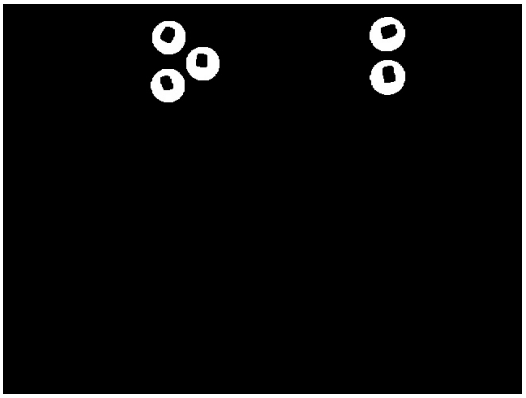
3. *Odpravljanje motenj.*
   Po upragovanju je iz slike potrebno odstraniti motnje. Najprej na sliko

apliciramo masko, ki pokrije vse objekte, ki se pojavijo na mestih, na katerih zagotovo ne morejo biti figure.



Slika 4.5: Aplikacija maske

Nato sliko obdelamo še z morfološkima operacijama zapiranja in odpiranja. Prva odpravi vse črne slikovne elemente na zaznanih objektih, druga pa vse bele slikovne elemente, ki niso del objektov [5].



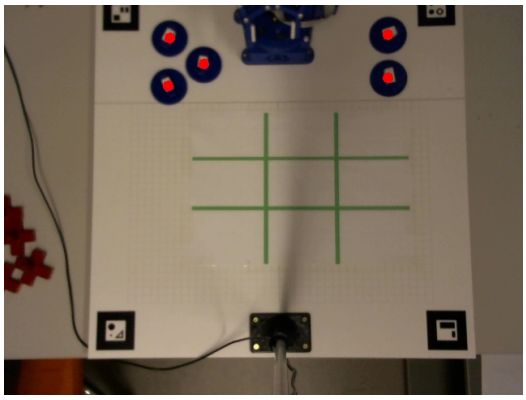
Slika 4.6: Operaciji zapiranja in odpiranja

4. *Izračun centroidov zaznanih objektov.*
   Tako obdelane slike so primerne za izračun centroidov zaznanih objektov. To lahko storimo s pomočjo momentov. Momenti so lastnosti zaznanih objektov, ki nam povejo kako intenzivno in v katero smer so usmerjeni

slikovni elementi zaznanega objekta. Poznamo več različnih momentov, vendar za izračun centroidov potrebujemo samo momente 1 0, 0 1 in 0 0. Izračun centroidov se izvaja po sledeči enačbi:

$$x = \frac{m_{10}(Objekt)}{m_{00}(Objekt)} \tag{4.7}$$

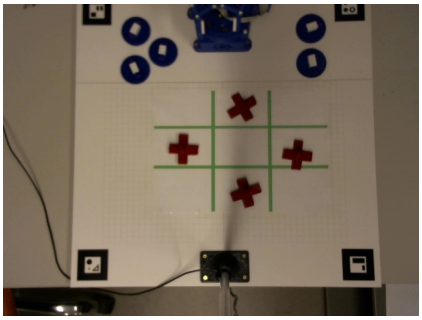
$$y = \frac{m_{01}(Objekt)}{m_{00}(Objekt)} \tag{4.8}$$



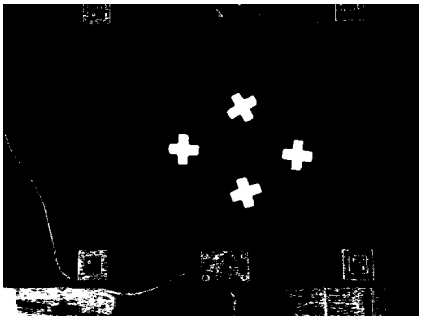
Slika 4.7: Izračunani centroidi detektiranih robotovih figur

4.1.3 Zaznavanje človekovih figur

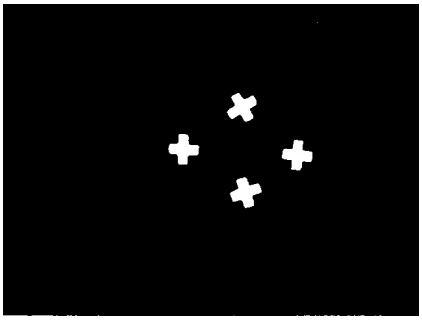
Zaznavanje človekovih figur deluje po enakih korakih kot zaznavanje robotovih figur. Edina razlika med zaznavanjem enih in drugih je ta, da pri zaznavanju človekovih figur sliko upragujemo glede na komponento R RGB barvnega prostora. Zaznavanje človekovih figur se izvede vsakič, ko človek odigra potezo.



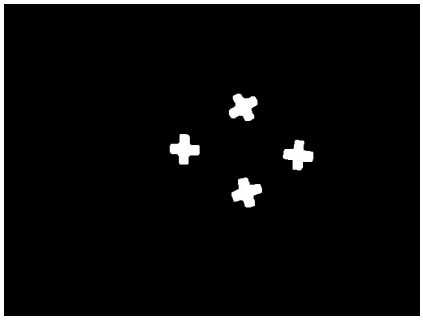
Slika 4.8: Upragovljena slika igralne površine po barvni komponenti R



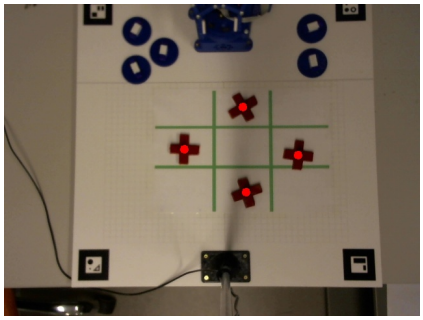
Slika 4.9: Upragovljena slika igralne površine po barvni komponenti R



Slika 4.10: Aplikacija maske



Slika 4.11: Operaciji zapiranja in odpiranja

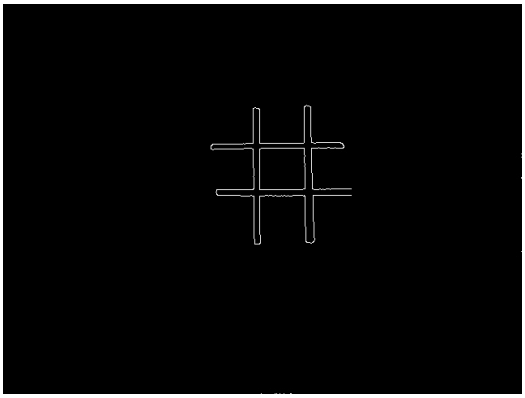


Slika 4.12: Izračunani centroidi detektiranih človekovih figur

4.2 Zaznavanje igralne površine

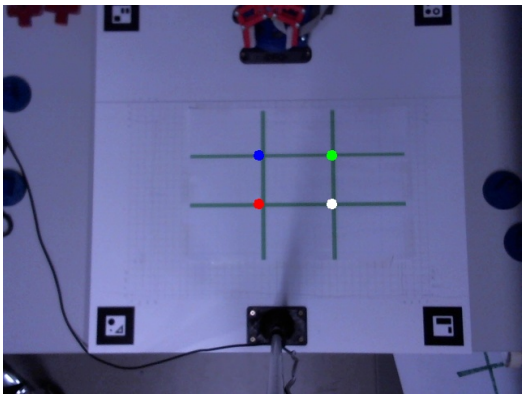
Zaznavanje igralne površine se izvede samo na začetku. Sistem najprej izvede detekcijo robov s pomočjo Cannyevega operatorja (glej sliko 4.13).

Nato vsebino pridobljene binarne slike ločeno projiciramo na x in y os tako, da dobimo dva vektorja, ki nam povesta, koliko slikovnih elementov obrobe igralne površine, je v vsaki vrstici ali stolpcu. Nad temi podatki izvedemo dve operaciji.

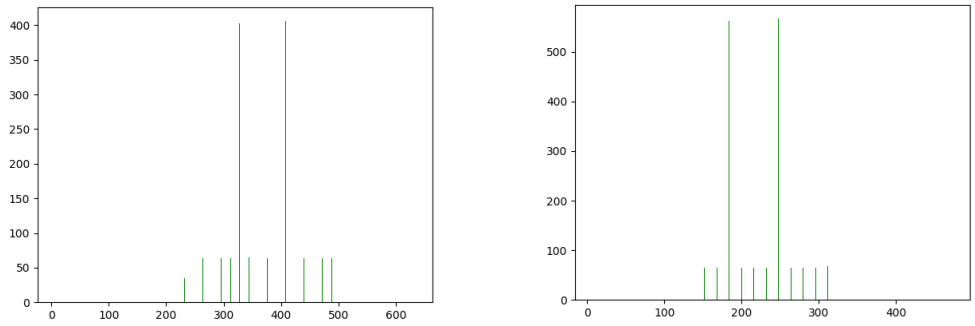


Slika 4.13: Rezultat Cannyevega operatorja

1. Vektorja razdelimo na več delov tako, da vsak del predstavlja nekaj zaporednih vrstic oziroma stolpcev. Sredinska vrstica ali stolpec posameznega dela predstavlja njegovo sklicno točko. Nato vrednosti vseh vrstic ali stolpcev v istem delu vektorja seštejemo. Tako sklicna točka dela vektorja, ki ima najvišjo vrednost, predstavlja sredinsko vrstico ali stolpec območja, v katerem se nahaja največ slikovnih elementov obrobe igralne površine. S kombinacijo štirih sklicnih točk, ki imajo najvišje vrednosti, približno izračunamo oglišča sredinskega pravokotnika igralne površine.

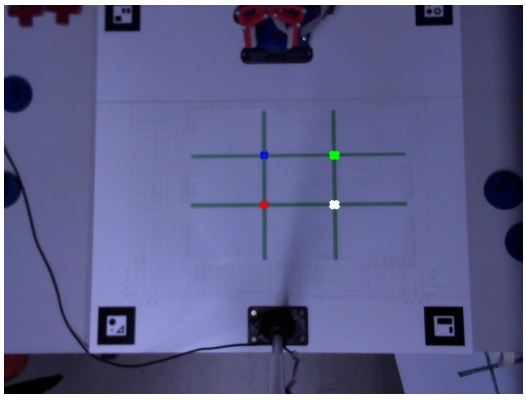


Slika 4.14: Približna oglišča sredinskega pravokotnika igralne površine

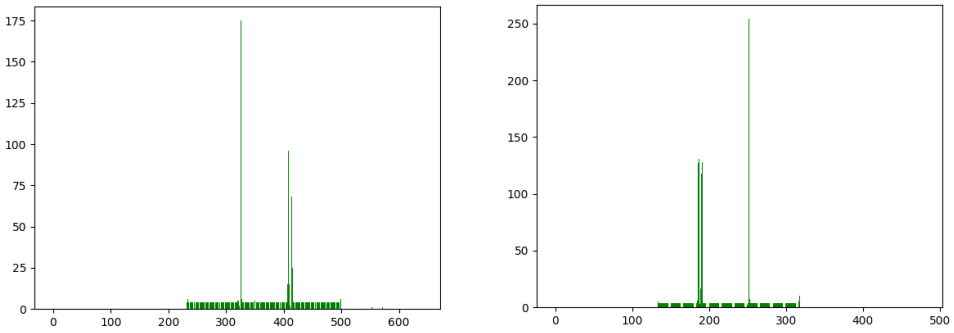


Slika 4.15: Vizualizacija sklicnih točk delov vektorja in njihovih vrednosti. Levo deli vektorja osi x, desno pa osi y.

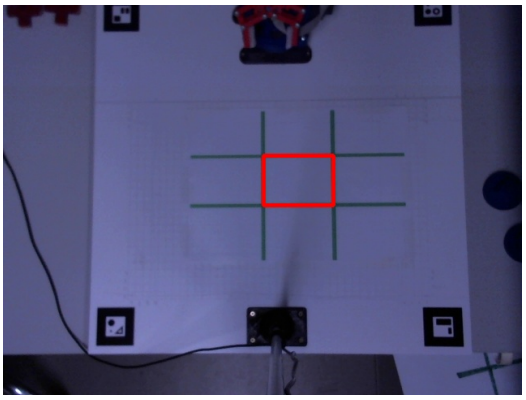
2. V drugem koraku postopek ponovimo brez deljenja vektorja. Vsaka vrstica ima svojo vrednost in svojo sklicno točko. Tako dobimo več glasov za oglišča sredinskega pravokotnika igralne površine, vendar jih lahko glede na oddaljenost od prej izračunanih približnih oglišč pravilno povprečimo in s tem robustno določimo oglišča sredinskega pravokotnika igralne površine.



Slika 4.16: Glasovi za oglišča sredinskega pravokotnika igralne površine

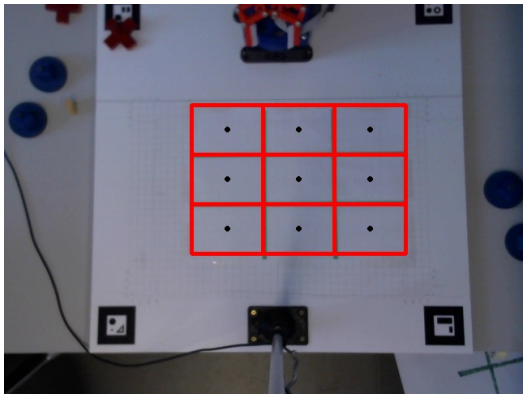


Slika 4.17: Vizualizacija sklicnih točk vektorja in njihovih vrednosti. Levo vektor osi x, desno pa osi y.



Slika 4.18: Pravilno določena oglišča sredinskega pravokotnika igralne površine

Po izračunanih ogliščih sredinskega pravokotnika igralne površine mu izmerimo višino in širino. S pomočjo teh dveh podatkov izračunamo obsegajoče pravokotnike in središča vseh ostalih igralnih polj.



Slika 4.19: Vsi obsegajoči pravokotniki in njihova središča

4.2.1 Delovanje sistema za zaznavanje sprememb na igralni površini

Vsi zgoraj opisani deli sistema so med seboj povezani. Tekom igre se najprej izvede zaznavanje igralne površine. Obsegajoče pravokotnike potrebujemo čez celo igro. Nato sistem izvede zaznavanje robotovih figur. S shranjenimi obsegajočimi pravokotniki in centroidi svojih figur je robot pripravljen na igro. Po odigrani robotovi potezi ponovno sledi zaznavanje robotovih figur saj tako preveri, ali mu je figuro uspelo pobrati. Temu sledi zaznavanje človekovih figur. V tem koraku preverimo, v katerem od na začetku izračunanih obsegajočih pravokotnikov se nahaja na novo zaznan centroid človekovih figur. To informacijo nato podamo algoritmu za igranje popolne igre Križci in krožci.

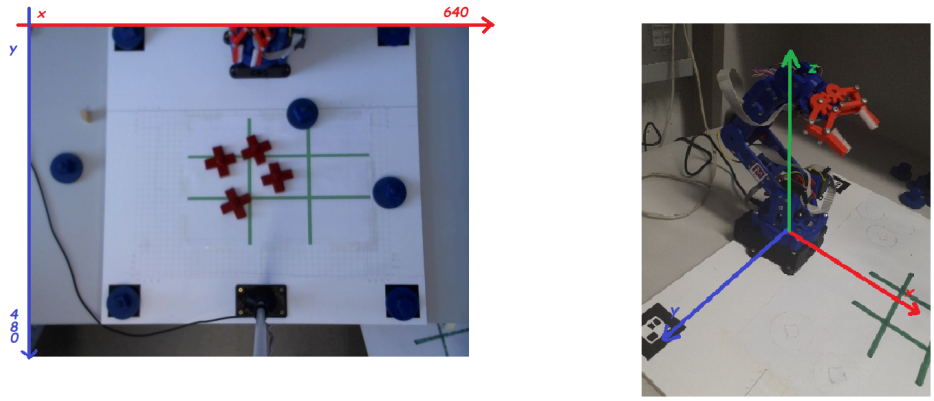
Poglavje 5

Upravljanje manipulatorja za izvedbo poteze

Zadnji del diplomske naloge je upravljanje robotskega manipulatorja za izvedbo poteze. Ta del sistema je odgovoren za pravilno transformacijo točk, ki jih prejme od sistema za zaznavanje sprememb na igralni površini ter za premik manipulatorja v transformirane točke. Sistem smo implementirali s pomočjo robotskega manipulatorja, ki je rezultat magistrske naloge Anžeta Rezlja [12, 15], ter ga upravljali s pomočjo programskega ogrodja Manus [10, 15]. Orodji smo že opisali v poglavju 2.

5.1 Transformacija centroidov

Pomembna funkcionalnost sistema za upravljanje manipulatorja je transformacija točk iz koordinatnega sistema kamere v koordinatni sistem sveta. Zaznavanje vseh objektov smo v celoti izvajali v koordinatnem sistemu kamere. To pomeni, da smo središča zaznanih objektov hranili kot indekse pikslov v sliki. Preden lahko taka središča uporabimo za pobiranje objektov, jih moramo najprej transformirati v koordinatni sistem sveta, ki ima izhodišče pri vznožju robota in ima 3 dimenzije (za razliko od koordinatnega sistema kamere, ki ima 2). Slika 5.1 prikazuje oba koordinatna sistema.

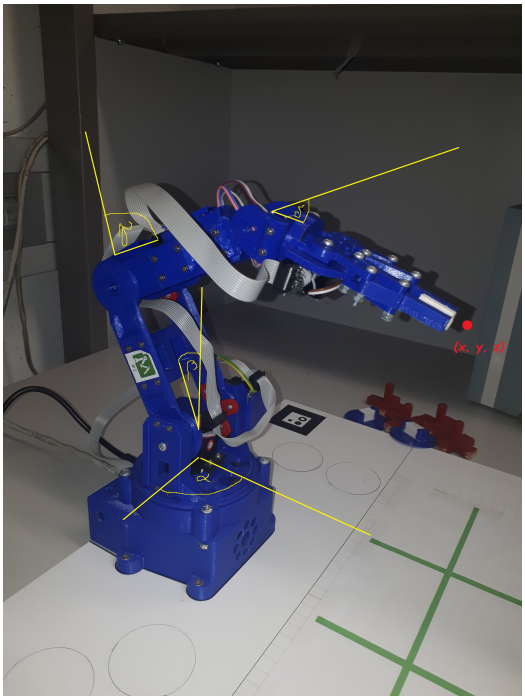


Slika 5.1: Koordinatna sistema kamere in sveta

Za transformacijo točk potrebujemo parameter kamere, ki se imenuje homografska matrika in nam ga Manus ob vsakem zajetju slike poda. Točko iz koordinatnega prostora kamere zapišemo v obliki stolpičnega vektorja in ji dodamo še homogeno komponento. Točko v taki obliki matrično množimo z inverzom homografske matrike. Rezultat je vektor, ki ga je treba deliti z vrednostjo spremenjene homogene komponente. Tako dobimo koordinati x in y točke v koordinatnem sistemu sveta. Manjka nam še koordinata z. Ker transformirane točke uporabljamo zgolj za odlaganje in pobiranje poznanih figur lahko koordinato z nastavimo na izmerjeno konstanto.

5.2 Premikanje robotskega manipulatorja

Programsko orodje Manus omogoča premikanje robotskega manipulatorja s pomočjo inverzne kinematike. Ta glede na podano točko v prostoru izračuna, za koliko mora robotski manipulator obrniti vsak sklep, da bo konec prijemala dosegel podano točko.



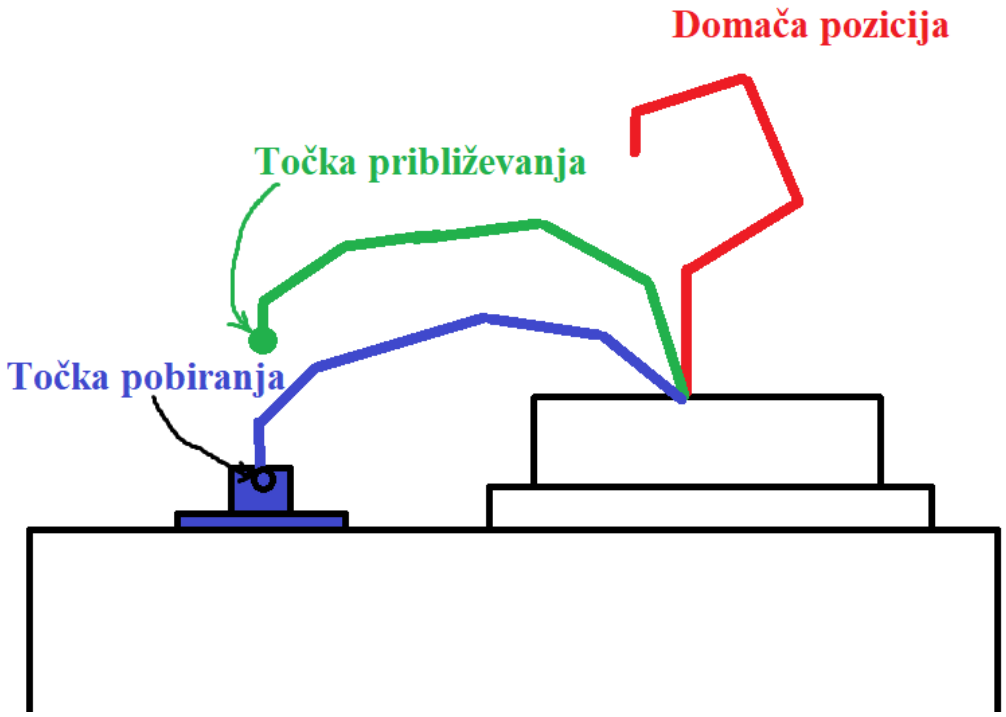
Slika 5.2: Konceptualni prikaz inverzne kinematike

Robotski manipulator mora v sklopu te diplomske naloge opraviti samo dva različna giba. Pobiranje in odlaganje figure. Oba lahko predstavimo kot zaporedje različnih točk, ki jih mora manipulator obiskati. Glede na lokacijo zaznanih figur in položaj robota smo v prostoru definirali 3 referenčne točke:

- Domača pozicija
  Točka, v katero se manipulator lahko varno premakne iz katere koli druge točke. Potrebovali smo jo za varno izvajanje gibov ter kot točko, v katero se manipulator odmakne, ko se izvaja zaznavanje človekovih figur.

- Točka približevanja
  Točka, v kateri se robot premakne neposredno nad objekt, ki ga želi pobrati. Uporabili smo jo v kombinaciji s točko pobiranja, da smo zagotovili večjo natančnost pobiranja.

- Točka pobiranja
  Točka, v katero se robot premakne, ko želi pobrati objekt. Koordinati x in y sta enaki kot pri točki približevanja, koordinata z pa je manjša.



Slika 5.3: Shema preračunanih točk v svetu

Podajamo še zaporedje točk, v katere se mora manipulator premakniti, da izvede giba pobiranje in odlaganje.

- Pobiranje
  1. Točka približevanja
  2. Točka pobiranja z odprtim prijemalom
  3. Zapri prijemalo
  4. Točka približevanja
  5. Domača pozicija

- Odlaganje
    1. Točka približevanja
    2. Točka pobiranja z zaprtim prijemalom
    3. Odpri prijemalo
    4. Točka približevanja
    5. Domača pozicija

Poglavje 6

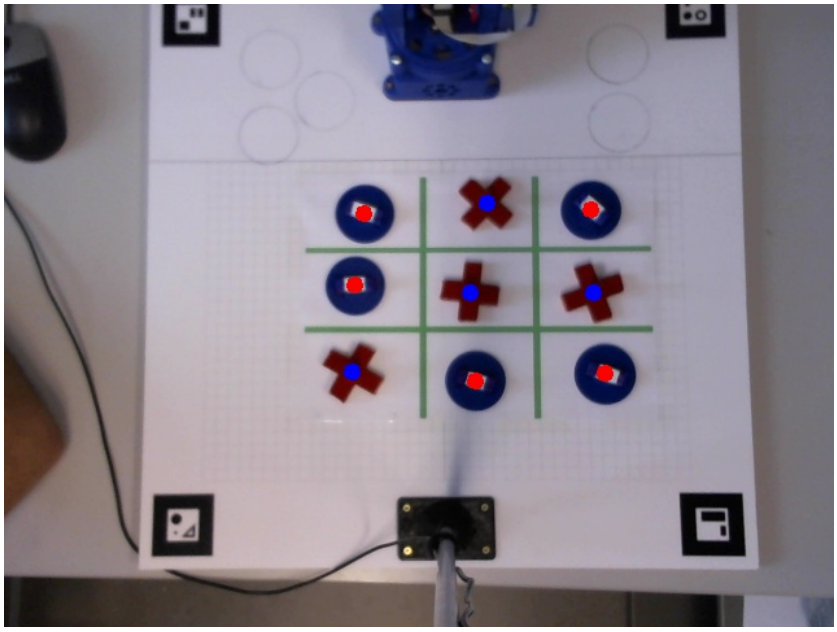
Vrednotenje sistema

6.1 Vrednotenje zaznavanja

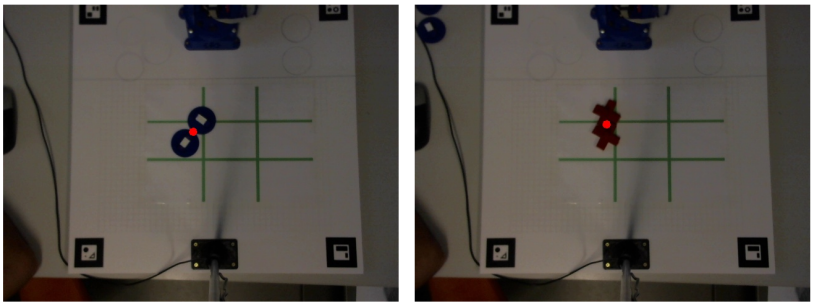
Vrednotenje zaznavanja smo izvedli tako, da smo naključno število figur postavili na naključno mesto pred robotom. Šteli smo, kolikokrat je robot pravilno zaznal figure in kolikokrat napačno. Kot napačno zaznavanje smo upoštevali zaznane objekte na mestih, kjer ni figure in odsotnost zaznanih objektov na mestih, kjer je figura bila. Primeri napačnih zaznavanj so se pojavili izključno takrat, ko so si bile figure tako blizu, da jih je sistem zaznal kot eno figuro.

	Vse testirane	Pravilno zaznane	Narobe zaznane	Natančnost
Krogci	295	285	10	96,61%
Križci	181	179	2	98,89%

Slika 6.1: Tabela meritev pri vrednotenju zaznavanja



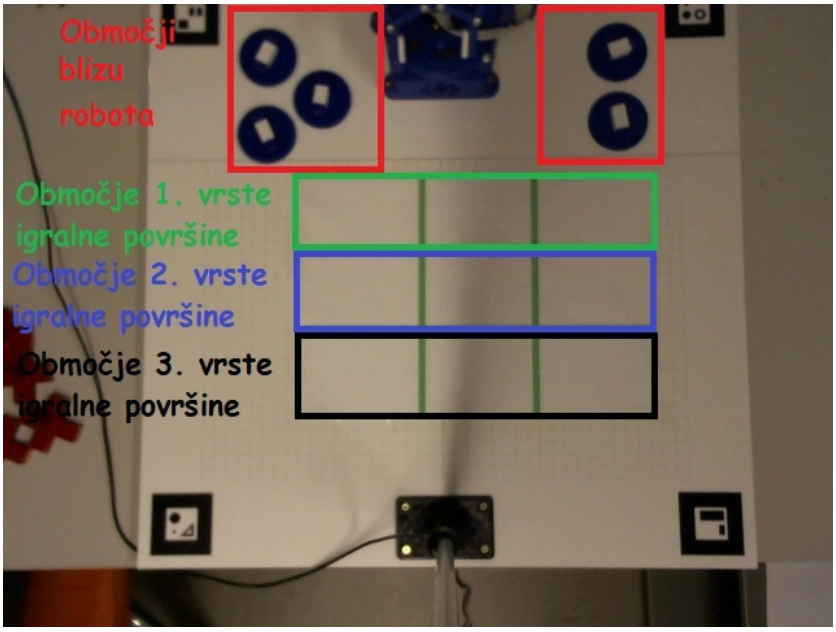
Slika 6.2: Primer pravilnih zaznavanj



Slika 6.3: Primera napačnih zaznavanj

6.2 Vrednotenje pobiranja in odlaganja

Vrednotenje pobiranja in odlaganja smo izvedli tako, da smo območje pred robotom razdelili na območje blizu robota, območje 1. vrste igralne površine, območje 2. vrste igralne površine in območje 3. vrste igralne površine.



Slika 6.4: Prikaz izbranih območij

Na vsako območje smo postavili dve figuri ter šteli, koliko figur je manipulator uspešno pobral ter odložil. Uspešno pobiranje je bilo vsako pobiranje, pri katerem se je manipulator vrnil v domačo pozicijo in v prijemalu obdržal figuro. Uspešno odlaganje je bilo vsako odlaganje, pri katerem je manipulator figuro odložil nazaj na mesto, kjer jo je pobral, brez, da bi mu padla iz prijemala. V primerih, ko je robot že v procesu pobiranja izpustil figuro se je proces odlaganja avtomatično štel kot neuspešen.

	Pobral	Odložil
Blizu robota	20/20 (100%)	20/20 (100%)
1. vrsta	20/20 (100%)	20/20 (100%)
2. vrsta	17/20 (85%)	17/20 (85%)
3. vrsta	16/20 (80%)	14/20 (70%)
Povprečna uspešnost	73/80 (91,25%)	71/80 (88,75%)

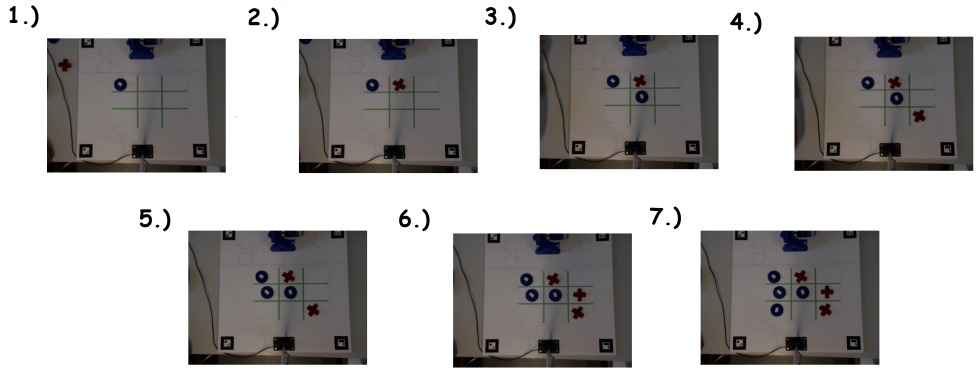
Slika 6.5: Tabela meritev pri vrednotenju pobiranja in odlaganja

Iz podanih meritev lahko opazimo, da se uspešnost pobiranja in odlaganja

manjša glede na oddaljenost figur od robota. Poglavitni razlog za to je oblika prijemala na figurah, ki je bilo namenoma zasnovano tako, da bi kar se da povečali uspešnost prijemanja figur blizu robota. Tako prijemalo povzroča slabše prijemanje na velikih razdaljah, saj se figura v robotsko prijemalo vpne pod takim kotom, da ima stik z zelo majhno površino prijemala. Posledično so bila zaradi takih slabših prijemanj, tudi odlaganja v območju 3. vrste igralne površine slabša. Pri igri se robotove figure nahajajo zgolj v območju blizu robota, zato slabše pobiranje in odlaganje figur v bolj oddaljenih območjih ne predstavlja težav. Vrednotenje bi lahko izvajali tudi tako, da bi figure pobirali v enem, odlagali pa v drugem območju. Tako vrednotenje bi bilo bolj v skladu s potekom igre, vendar pa bi robot še vedno slabše odlagal tiste figure, ki jih je pobral v najbolj oddaljenih območjih.

6.3 Vrednotenje igranja

Vrednotenje igranja smo izvedli tako, da smo proti robotu odigrali 10 iger Križcev in krožcev. Zaradi previdne izhodiščne postavitve robotovih figur v območje blizu robota je robot brez napake odigral vseh 10 iger zapored. Slika 6.5 prikazuje primer odigrane igre z robotom.



Slika 6.6: Primer odigrane igre z robotom

Poglavje 7

Zaključek

7.1 Doseženi cilji

V sklopu diplomskega dela smo razvili sistem, ki robotskemu manipulatorju omogoča igranje popolne igre Križcev in krožcev. Posamezni deli diplomske naloge se lahko uporabljajo tudi kot samostoječi sistemi za zaznavanje in lokalizacijo modrih ali rdečih objektov ter za pobiranje in odlaganje zaznanih objektov. Za potrebe te diplomske naloge vse razvite komponente sistema delujejo dovolj natančno, vendar pa bi jih morali, za uporabo pri reševanju drugih problemov, še izboljšati.

Delo s projektom Manus je potekalo brez večjih zapletov. Manjše probleme je povzročalo razvojno okolje, saj smo celotno diplomsko nalogo razvijali v spletnem vmesniku. Spletni vmesnik omogoča zgolj najpreprostejše funkcionalnosti urejevalnika programa, zato je bilo popravljanje in testiranje zelo zamudno. Za testiranje sistema zaznavanja smo porabili največ časa, saj smo kamero izklopili iz manipulatorja, jo priključiti na osebni računalnik, izvajali testiranje ter nato popravljeno kodo prenesli na spletni vmesnik.

7.2 Možne izboljšave

V trenutni izvedbi sistema zaznavanje igralnih figur deluje dovolj natančno. Robustnost bi lahko povečali tako, da bi z uporabo Cannyevega operatorja zaznali robove vseh objektov ter jim nato izračunali kompaktnost. Ker so ene figure okrogle, druge pa v obliki križa, bi s to mero lahko ločili ene od drugih tudi, če bi bile enakih barv. Prav tako bi lahko za zaznavanje uporabili bolj napredne metode, ki temeljijo na konvolucijskih nevronskih mrežah.

Robotski manipulator se glede na ceno proizvodnje odziva zelo natančno, vendar pa vsekakor prihaja do manjših odstopanj pri premikanju v želene točke. Odstopanja so bolj ali manj konstantna, zato smo jih odpravili z uporabo manjših odmikov. Prav tako je problematična sinhronizacija gibov. Nekateri sklepi se premikajo hitreje kot drugi in zato pride do nepredvidljivih trkov robotskega manipulatorja.

7.3 Sklepna misel

Uspelo nam je razviti sistem, ki s pomočjo igre človeku približa robotiko. Predvideni uporabniki sistema so predvsem otroci, vendar pa je namenjen vsem, ki bi se radi seznanili z interakcijo in delovanjem robotov. V nadaljnjem razvoju bi se bilo zanimivo spopasti s kompleksnejšimi igrami, kot so dama ali pa celo šah.

Literatura

[1] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul F. Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *CoRR*, abs/1606.06565, 2016.

[2] Christoph Bartneck, Tomohiro Suzuki, Takayuki Kanda, and Tatsuya Nomura. The influence of people's culture and prior experiences with aibo on their attitude towards robots. *AI & SOCIETY*, 21(1):217–230, Jan 2007.

[3] Murray Campbell, A.Joseph Hoane, and Feng hsiung Hsu. Deep blue. *Artificial Intelligence*, 134(1):57 – 83, 2002.

[4] Kevin Crowley and Robert S. Siegler. Flexible strategy use in young children's tic-tac-toe. *Cognitive Science*, 17(4):531–561.

[5] Nick Efford. *Digital Image Processing: A Practical Introduction Using Java (with CD-ROM)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 2000.

[6] Janken (rock-paper-scissors) robot with 100 percent winning rate (human-machine cooperation system). Dosegljivo: `http://www.k2.t.u-tokyo.ac.jp/fusion/Janken/index-e.html`. [Dostopano: 9. 8. 2018].

[7] Kuka kleinroboter spielt Siedler von Catan. Dosegljivo: `https://www.oth-regensburg.de/hochschule/aktuelles/einzelansicht/`

news/kuka-kleinroboter-spielt-siedler-von-catan-1.html. [Dostopano: 9. 8. 2018].

[8] Kuka small robot plays Settler of Catan. Dosegljivo: https://www.kuka.com/en-de/press/news/2016/03/kuka-small-robot-plays-settler-of-catan. [Dostopano: 9. 8. 2018].

[9] Logitech HD webcam C525 specifications. Dosegljivo: https://support.logitech.com/en_us/product/hd-webcam-c525/specs. [Dostopano: 9. 8. 2018].

[10] Manus. Dosegljivo: https://github.com/manus-project/manus. [Dostopano: 9. 8. 2018].

[11] OpenCV about. Dosegljivo: https://opencv.org/about.html. [Dostopano: 9. 8. 2018].

[12] Anže Rezelj. Razvoj nizkocenovnega lahkega robotskega manipulatorja. Master's thesis, Fakulteta za računalništvo in informatiko, Slovenija, 2017.

[13] The history of robotics in the automotive industry. Dosegljivo: https://www.robotics.org/blog-article.cfm/The-History-of-Robotics-in-the-Automotive-Industry/24. [Dostopano: 9. 8. 2018].

[14] Andreas Van Cranenburgh and Ricus Smid. Use of Mini max in a tic-tac-toe game. *Encyclopedia of AI project, University of Amsterdam.*

[15] Luka Čehovin Zajc, Anže Rezelj, and Danijel Skočaj. Open-source robotic manipulator and sensory platform. *Robotics in Education (RiE2017)*, Apr 2017.