Универзитет у Београду
Машински факултет

Мастер академске студије

Индустрија 4.0

МАШИНСКО УЧЕЊЕ ИНТЕЛИГЕНТНИХ РОБОТСКИХ СИСТЕМА

П Р О Ј Е К А Т

Оцена пројектног задатка:	Предметни наставници:	Проф. др Зоран Миљковић Проф. др Радиша Јовановић Доц. др Милица Петровић
	Предметни сарадници:	Александар Јокић, маст.инж.маш. Лазар Ђокић, маст.инж.маш. Митра Весовић, маст.инж.маш.

Група: 1

Потпис наставника:	РБ	Презиме и име:	Бр.инд.	Потпис:
	1.	Јелена Цвијан	4010/2020	
	2.	Катарина Пантовић	4005/2020	
	3.	Ива Барбуловић	4005/2019	
	4.	Владимир Станковић	4002/2019	
	5.	Стефан Ковач	4003/2020	
	6.	Огњен Збиљић	4002/2020	
	7.	Немања Јанев	4004/2020	

Школска година: 2020/2021.

Резиме

Пројекат описује унапређење постојећег транспортног система у оквиру „НН“ предузећа. То се постиже заменом постојећих машина (виљушкара и палетара) са интелигентним мобилним роботом. Симулација кретања нових транспотрних машина изведена је у лабораторијским условима, на Машинском факултету у Београду, израдом функционалне макете мобилног робота коришћењем Lego MINDSTORM EV3 сета. Управљачки систем је имплементиран на поменутог робота помоћу вештачких неуронских мрежа, различитих метода локализације и више хеуристичких норми. Извођењем експеримената који су обухватали кретање мобилног робота по различитим трајекторијама, у оквиру модела технолошког окружења предузећа, доказана је изводљивост креираног и примењеног концепта.

Кључне речи: *транспортни систем, симулација кретања, интелигенти мобилни робот, вештачке неуронске мреже, А звезда алгоритам, Калманов филтер, извршени експерименти.*

Садржај табела

Садржај слика

Садржај

1. Увод

У пројектном задатку ће се анализирати примена мобилног робота у предузећу „НН“, ради повећања ефикасности транспортног система у производном погону за SIVACON електро-ормане. Како манипулација материјалом заузима велики део времена у производном процесу, а и до 35% укупних трошкова производње, модернизација овог подсистема има вишеструку корист. Иако, на први поглед, увођење аутономних мобилних робота у овај систем не представља оправдану инвестицију због великог иницијалног улагања при куповини опреме, уложена средства ће се исплатити у виду смањења циклусног времена производње, као и смањења радне снаге потребне за опслуживање производње.

Верификација успешности управљања роботом заснована је на експерименту изведеном на лабораторијском моделу радног погона предузећа „НН“. Мобилни робот који је коришћен је конфигурисан деловима из комплета LEGO Mindstorm EV3, а управљање се остварује помоћу рачунара на ком је инсталиран софтвер MATLAB и у њему имплементирани потребни Toolbox-еви. Функције кретања остварене су помоћу модела кретања мобилног робота уз коришћење вештачких неуронских мрежа како за ротацију робота тако и за кориговање положаја помоћу Калмановог филтера, док се иницијално генерисање оптималне путање врши применом A* алгоритма.

У другом поглављу, дефинисан је проблем који се обрађује у оквиру пројекта, као и детаљно објашњење Layout-а технолошког окружења.

Треће поглавље се односи на концепцијско решење пројектног задатка.

У четвртом поглављу се објашњавају појмови везани за мобилне роботе, сензоре и интелигентне технолошке системе.

У петом поглављу се даје осврт на основне карактеристике LEGO Mindstorm EV3 робота, са детаљима коришћене конфигурације.

Шесто поглавље говори о моделима кретања уопштено, уз нагласак на моделу кретања на основу пређеног пута, који је и коришћен у овом пројектном задатку.

Седмо поглавље објашњава вештачку интелигенцију и примењене вештачке неуронске мреже.

Осмо поглавље се односи на Калманов филтер и његову имплементацију.

У деветом поглављу је детаљно објашњен A* алгоритам, његове теоријске основе, као и начин програмирања и имплементације.

Десето поглавље обухвата анализу и објашњење коришћених кодова у софтверском пакету MATLAB.

Једанаесто поглавље детаљно описује остварене резултате приликом симулирања кретања робота по моделу технолошког система предузећа.

Дванесто поглавље обухвата сажете закључке изведене израдом овог пројекта, док је у тринаестом поглављу наведена коришћена литература.

2. О пројектном задатку

Транспортни систем представља саставни део сваког производног система. Он представља есенцијалну интегративну компоненту целокупног погона, без кога савремено предузеће не може заиста бити савремено. Немогуће је смањити циклусно време чак и најсавременијим машинама ако се транспорт обрадака између тих машина обавља у неадекватном временском року. Полазна тачка у поступку имплементације интелигентног транспортног система за резултат је имала оптимално распоређивање машина у погону у складу са потребним технолошким поступцима, али она није обухваћена овим пројектом, већ је тај део синтетисан у радно окружење у којем је савремени мобилни робот вршио кретање.

2.1 Layout технолошког окружења

Пројектни задатак реализован је за реално предузеће „НН“ тако да су сви подаци који су коришћени резултат раније детаљне студије. Такође, то значи да се и решење самог пројекта може применити у реалним условима.

Предузеће се бави производњом SIVACON електро-ормана тако да се целокупни процес од производње делова до монтаже одвија према Siemens AG-овом технолошком поступку. У предузећу разликујемо основне машине које су приказане у следећој табели:

Табела 1. Машине у погону предузећа

M1	Маказе за сечење
M2	CNC машина за пробијање и просецање
M3	CNC хидраулична „апкант" преса
M4	Машина за исецање профила
M5	Стубна бушилица
M6	Стубна бушилица
M7	Кружна тестера
M8	Оштрилица
M9	Линија за обраду делова од бакра

Простори за складиштење су организовани и прилагођени одговарајућим технолошким процесима. Међутим, као отежавајућа околност приликом пројектовања Layout-а се појавила сама потпорна конструкција хале. Наиме, постојећи простор морао се прилагодити, јер је производна хала предузећа изграђена пре извршене анализе токова материјала, па се носећи стубови нису могли изместити. Решење ове анализе је имплементирано у поставку овог пројектног задатка и оно садржи (слика 2.1): улазно

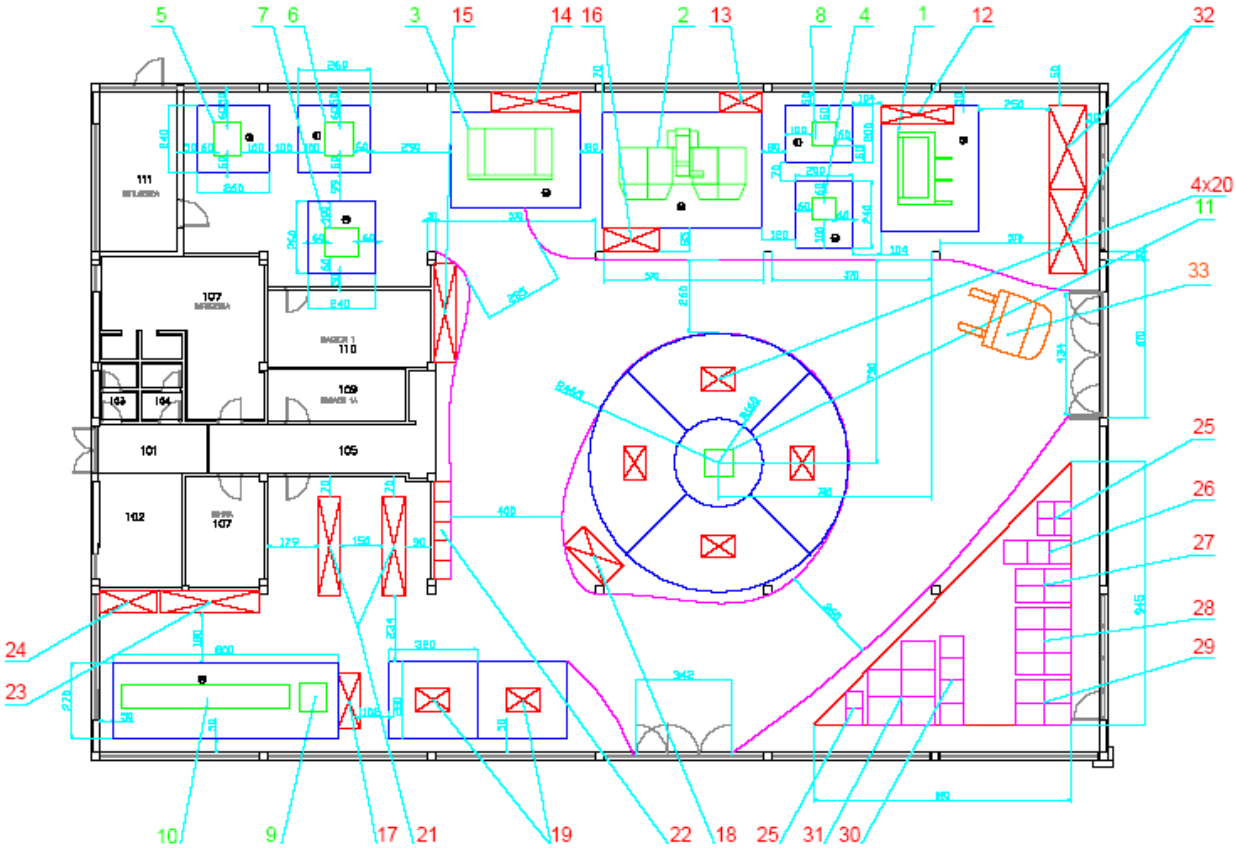
складиште полуфабриката (позиција 32), међускадиште (позиција 15), складиште готових делова од лима (позиција 21), складиште вијчане робе и компонената за монтажу (позиција 22), складиште готових делова од бакра (позиција 23) и складиште готових производа (позиције 25 до 31).

Складиште полуфабриката урађено је према постојећој комплетној конструкционој документацији и представљено је као регално складиште са полицама. Овом типу складишта припадају и међускладиште, складиште готових делова од лима и складиште готових делова од бакра. Складиште вијчане робе и компонената за монтажу је реализовано као складиште кутија којим је предузеће већ располагало. Складиште готових производа је палетног типа и смештено је у простор у углу хале. У оквиру тог простора врши се одлагање различитих врста SIVACON електро-ормана у зависности од претходно дефинисаног плана складиштења.

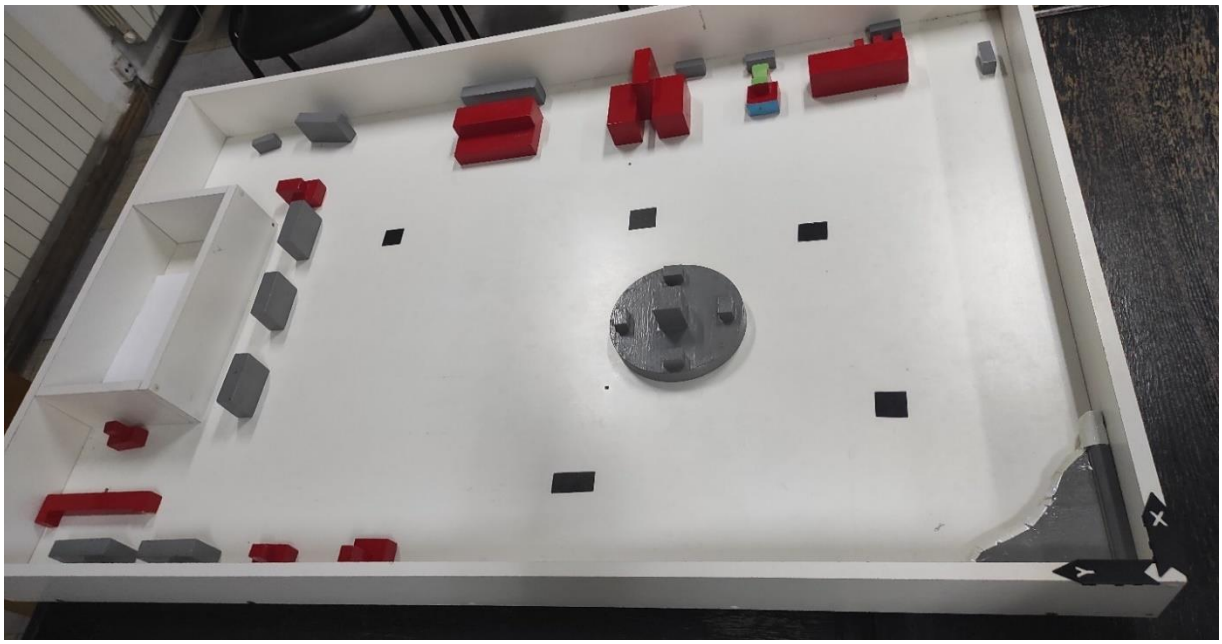
Следећа целина која се у погону издваја је свакако монтажа. У овом погону реч је о ручној монтажи која је с обзиром на неколико различитих типова готових производа који се формирају на палети изискивала два диспозициона простора за монтажу. Један је са четири радна места у кружном распореду и он је предвиђен за веће ормане (позиција 20), док је други са два радна места за мање ормане (позиција 19). Свака од целина допуњена је још и радним столом за формирање подсклопова (позиција 18), као и складиштем вијчане робе и компонената (позиција 22), складиштем полупроизвода (позиција 21) и складиштем готових делова од бакра (позиција 23).

Овај распоред у предузећу „НН“ верно је представљен на моделу који је коришћен за експериментално испитивање софтверског решења. Једина разлика је у томе што позиција 18, односно сто за монтажу подсклопова у оквиру монтаже великих ормана није био постављен у лабораторијско окружење. Та разлика се лако компензује јер се та препрека може унети као препрека у алгоритам за генерисање путање мобилног робота који смо користили за дефинисање дозвољених кретања.

Layout производно-монтажног погона и његов модел реализован у просторијама Машинског факултета у Београду приказани су на сликама 2.1 и 2.2 респективно.



Слика 2.1. Layout производно-монтажног предузећа „HH" [5]



Слика 2.2. Лабораторијски модел окружења

3. Концепцијско решење пројектног задатка

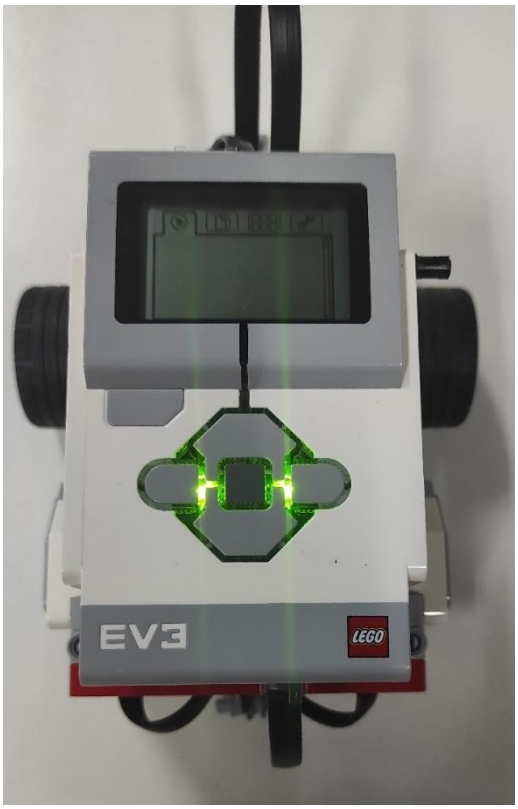
Пројектовање транспортног система од великог је значаја за правилно функционисање самог производног процеса. Процењено је да са имплементацијом оваквог решења долази до великих уштеда у времену, материјалним средствима и другачијој ангажованости радника, јер манипулација материјалом и транспорт чини до 35% укупних трошкова производње. Увођење роботског мобилног система у првом тренутку био би већи материјални издатак, али би се временом инвестиција исплатила и довела до значајних уштеда.

3.1. Ток развоја пројектог решења

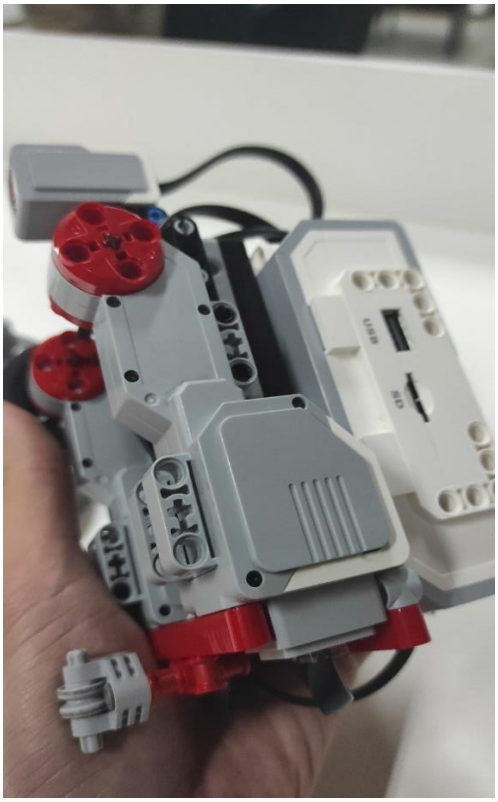
Почетак пројектовања везан је за упознавање са layout-ом предузећа „HH" који је описан у поглављу 2.1. Уочено је да мобилни робот мора обављати све задатке које су до сада обављали виљушкари и ручни хидраулични виљушкари (палетари). Почетна претпоставка је да робот у реалним условима мора имати све карактеристике које поседује и претходно коришћено транспортно средство. Ту је првенствено реч о одговарајућој носивости и о адекватним димензијама које би омогућиле да мобилни робот има у најмању руку једнако добар приступ свакој машини као и виљушкар. Ипак, у случају пројектног задатка није се до детаља разматрала ова проблематика већ је било неопходно пре свега извршити одговарајући експеримент у окружењу које је формирано у факултетским условима.

Конфигурација мобилног робота: Први потребан технички захтев при конфигурисању мобилног робота је стабилност. Коришћена је структура са три тачке ослонца која је у исто време омогућавала и добру мобилност. Помоћни точак има два степена слободе и он омогућава ротацију робота око своје осе, те су на тај начин смањене грешке које би се јављале при кретању. Управљачка јединица мобилног робота (енг. *Intelligent Brick*) имплементирана је у конструкцију тако да се повезивање са моторима и сензорима остварује на што интуитивнији начин. Сам модул за управљање је повезан спојевима који обезбеђују његово симетрично постављање и стабилно приањање на саму носећу структуру мобилног робота (слике 2.3 и 2.4). Такође, за дефинисани проблем предвиђен је оптички сензор и два погонска мотора, један за ротацију левог и један за ротацију десног точка.

Кратак преглед потребних и остварених функција мобилног робота: Почетак управљања самим роботом заснован је на моделу кретања који је имплементиран у MATLAB окружење. Следећа фаза обухвата обучавање вештачких неуронских мрежа преко обучавајућих парова за ротацију робота у односу на задате углове при кретању. Након тога потребно је обучити нову мрежу која врши детекцију боје и процес класификације, а која се касније имплементира у потпрограм за корекцију кретања. Задатак не би био потпун без завршне фазе која обухвата имплементацију A* алгоритма. Овај алгоритам формиран је у циљу остваривања жељених позиција на основу задавања почетне и крајње тачке, као и мапе окружења где су дефинисане препреке, уз кориговање положаја применом Калмановог филтера.



Слика 2.3 Управљачки модул робота



Слика 2.4 Мотори

4. Индустријски и мобилни роботи, сензори и интелигентни технолошки системи

Постизање циљева, који су постављени дефинисањем пројектног задатка, могу се остварити искораком у мобилну роботику са тенденцијом достизања што већег степена аутономности. Међутим, мора се имати у виду да је за постизање тога потребно имплементирати знања о индустријским роботима, мобилним роботима, сензорици и наравно интелигентним технолошким системима. У наставку ће ови појмови бити детаљније описани.

4.1. Индустријски роботи

Како би се прецизно дефинисао појам индустријског робота биће искоришћене две популарне дефиниције:

1. Дефиниција по RIA:

Индустријски робот је вишефункционални манипулатор пројектован да помера материјал, делове, алате и специјалне уређаје кроз различита програмирана кретања за извршавање различитих задатака [5].

2. Дефиниција по ISO:

Индустријски робот је аутоматски управљана вишенаменска манипулациона машина са неколико степени слободе, која може бити фиксна или покретна, а користи се за аутоматизацију у индустрији [5].

Иако не постоји савршена дефиниција индустријског робота може се рећи да су два појма кључна: програмабилност и флексибилност. Програмабилност нам говори да се програмирана кретања и помоћне функције могу мењати без физичких измена у структури робота, док флексибилност омогућава прилагођавање различитим применама са или без физичких измена.

Разлог развоја оваквих система огледа се у потреби за аутоматизацијом послова који подразумевају принудни и тешки рад, односно њиховим олакшавањем и смањењем ризика по човека, што је и инспирисало људе да теже ка развоју ове области. Постепено се стигло до савремене примене индустријских робота која има за циљ повећање продуктивности, подизање и одржавање константног нивоа квалитета производа, подизање флексибилности и хуманизацију рада. Наиме, у почетној фази примене робота у индустрији они су углавном обављали оне послове који су монотони или послове који се обављају у нездравим условима (просторије са високим температурама или просторије контаминиране радиоактивним материјалом). Уопштено говорећи, то су послови од којих је човека пожељно ослободити, па тако роботи имају одређену улогу у поменутој хуманизацији рада.

Илустративно се ови послови описују са три D и три H.

dull-досадан/монотон, **dirty**-прљав, **dangerous**-опасан. **hot**-вруће, **heavy**-тежак/рискантан и **hazardous**-рискантно.

Ипак роботи су веома продуктивни, раде у више смена, имају уједначен ритам рада, праве мало шкарта те је у капиталистичком уређењу, главни мотив за њихову примену првенствено економски.

Индустријски роботи се данас примењују у широком спектру области и функција од којих се издвајају [3]:

- Манипулација (опслуживање машина, палетизација/депалетизација),
- Обављање процеса (заваривање, обрада резањем, фарбање) ,
- Манипулација и обављање процеса (монтажа) и
- Специјални задаци (мерење и контрола)

Индустријски роботи се могу поделити на различите начине према [3]:

- Намени,
- Степену универзалности,
- Кинематичким, геометријским и енергетским параметрима,
- Методама управљања итд.

Предмет изучавања у оквиру овог курса су мобилни роботи који спадају у категорију интелигентних робота, тј. они који разумеју задатак и околину на основу информација са сензора, а помоћу техника вештачке интелигенције могу доносити одлуке у реалном времену.

4.2. Мобилни роботи

Мобилна роботика се разликује од индустријске роботике и обухвата шири спектар проблема којима се може приступити. Из овог разлога у покушајима да се дефинише мобилни робот наилази се на веће проблеме него код индустријског робота јер се овде мора имати у виду појам мобилности и сва питања која она отвара, а о којима ће више бити речи у одељку о интелигентним технолошким системима. Из тог разлога дефиниција индустријског робота не може бити строго примењива и на мобилне роботе.

Мобилни робот тако можемо дефинисати као мобилан и манипулативан физички систем који се креће кроз неуређен простор, остварујући притом интеракцију с људским бићима или обављајући неки посао уместо њих [4].

На овај начин они се посматрају као тзв. услужни роботи. Услужни роботи би временом требало да се развију у персонализоване роботе који ће умногоме личити на људе. Наиме са становишта величине тржишта робота кућни робот представља свети грал тржишта робота. Ипак, до тада будућност индустријске роботике у великој мери лежи у мобилним роботима [4].

Мобилни роботи се могу класификовати по неколико независних карактеристика, од којих свака у великој мери одређује кључне аспекте њиховог система управљања и навигације. Тако имамо класификацију према средини у којој се крећу [3]:

- ваздушни,
- водени,

- свемирски и
- копнени.

Друга подела мобилних робота је извршена са аспекта врсте локомоције [3] :

- Роботи са точковима (енг. *wheeled*),
- Роботи са ногама (енг. *legged*),
- Гусеничари (енг. *tracked*) и
- Змијолики роботи (енг. *serpentine*).

Постоје још и поделе према [6]:

- Врсти терена коју савладавају (унутрашњи или спољашњи простори),
- Флексибилности тела робота (једно и вишетелесни роботи, роботи с еластичним или крутим телом),
- Облику тела робота (роботи с једноставном или сложеном структуром, роботи у облику инсеката, итд.),
- Примени (индустријски роботи, роботи за едукацију, истраживачки роботи, роботи за пружање услуга, роботи за забаву, итд.),
- Начину настанка (модернизована стара возила, нова возила) и
- Степену аутономности

Мотиви развоја оваквих робота на самом крају могу се таксативно навести кроз следеће ставке:

- Могућност приступа местима која су за човека: опасна по живот (нпр. минска поља, нуклеарна постројења, експлозивне зоне итд.), превише удаљена (нпр. Марс) или недоступна (нпр. микроскопски простори),
- Смањење трошкова и повећање продуктивности рада: мањи додатни трошкови рада због смањења администрације, потребног простора и сл., већа свеукупна расположивост робота (нема пауза за ручак, годишњих одмора итд.) бржи рад у односу на човека, нпр. брже бојење, сечење, кретање итд., повећан квалитет производа или услуге и
- Старење становништва у развијеним земљама.

Примена им је стога шира од индустријских робота и може бити у [6]:

- Индустрији,
- Војној примени,
- Домаћинству,
- Осигурању објеката,
- Истраживању свемира, подморског света и планета,
- Пољопривреди,
- Сервисима,
- Помоћи старијима и немоћнима,
- Чувању и одгоју деце.

4.3. Сензори

Говорећи о дефиницијама робота истичу се две карактеристике робота које у великој мери одређују његове особине и функционалност. Самосталност и интелигенција робота доводе до високе флексибилности и успешне примене. Да би робот могао радити самостално, без обзира на ниво самосталности, он мора бити „свестан" себе и своје околине. Прецизније речено, робот мора имати могућност мерења сопственог положаја и брзине, као и мерења различитих величина у радној околини чиме стиче представу о спољном простору. Различити мерни уређаји и системи којима робот добија информације о себи и околини називају се сензори. Данас су то уређаји за мерење угаоног и транслаторног померања, различити сензори додира, уређаји за мерење растојања, силе, убрзања и сл. Посебну класу сензорских система чине визуелни системи, данас често у употреби. Генерално говорећи сензори у роботици омогућавају:

- Мерење физичких величина у управљачким повратним спрегама,
- Детекцију објеката (препознавање, одређивање позиција и оријентација објеката),
- Корекцију грешака у моделу робота и околине,
- Откривање и решавање проблема приликом дешавања нежељених ситуација,
- Мониторинг интеракције са околином,
- Осматрање промена у околини које могу утицати на задатак и
- Инспекцију резултата процеса.

У роботици сензори се деле на [5]:

- унутрашње и
- спољашње сензоре.

Унутрашњи сензори дају информације о позицији, брзини, убрзању, сили и моментима. Спољашњи сензори представљају сензоре околине у које спадају тактилни сензори, сензори близине и сензори удаљености.

Такође, не сме се заборавити ни следећа подела сензора која узима у обзир енергетски утицај окружења на систем и обрнуто. У том погледу се разликују [5]:

- пасивне и
- активне сензоре.

Пасивни сензори врше мерење енергетског утицаја окружења на систем и у њих спадају микрофони и камере, док активни сензори обухватају оне сензоре који емитују енергију у окружење мерећи при томе одговор окружења на енергетски стимуланс. У овој групи се налазе ласерски и ултразвучни сензори.

4.4. Интелигентни технолошки системи

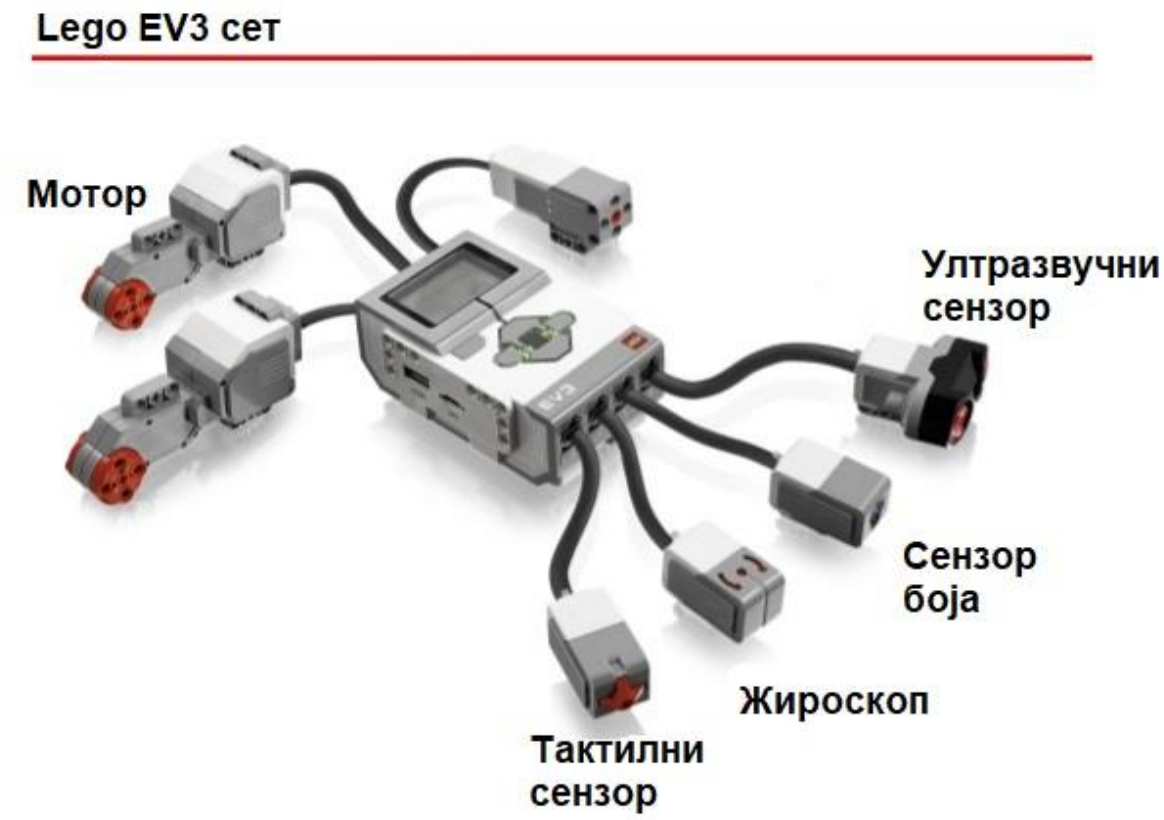
Говорећи о мобилном роботу уочено је да се са кретањем јавља много питања на која је потребно одговорити да би робот на жељени начин остварио локомоцију. Где се налази? Куда иде? Како до тамо доћи? Како избећи колизију са другим објектима? Закључено је да робот може сам формирати свест о простору у ком се креће коришћењем својих сензора. Уз то се мора имати у виду да он при томе обавља задатке различитог степена сложености. Свођењем свих чињеница закључујемо да робот има одређени степен интелигенције, па систем у коме се он креће можемо назвати интелигентни технолошки систем.

ИТС је највиша класа флексибилних технолошких система која је остварила синергију вештачке интелигенције и компјутерски интегрисаних технологија, са циљем да систем има могућност реализације активности у неодређеном технолошком окружењу, уз перманентан пораст вероватноће успешног понашања [11].

Мобилни робот коришћен у пројектном задатку представља синтезу два својства: мобилности и интелигенције и на тај начин он великој мери почиње да личи на природни организам, који може сам решавати проблеме који се поставе пред њега.

5. Опис расположивих компоненти

На самом почетку, по разумевању пројектног задатка било је неопходно саставити конфигурацију мобилног робота која би одговарала функцији постављеног проблема. На располагању је био мобилни робот типа MINDSTORMS® EV3, који је компанија LEGO® представила у јануару 2013. године. То је програмабилни роботски комплет, осмишљен као замена за другу генерацију Lego Mindstorm пакета који је назван Lego Mindstorm NXT. EV3 комплет је приказан на слици 5.1.

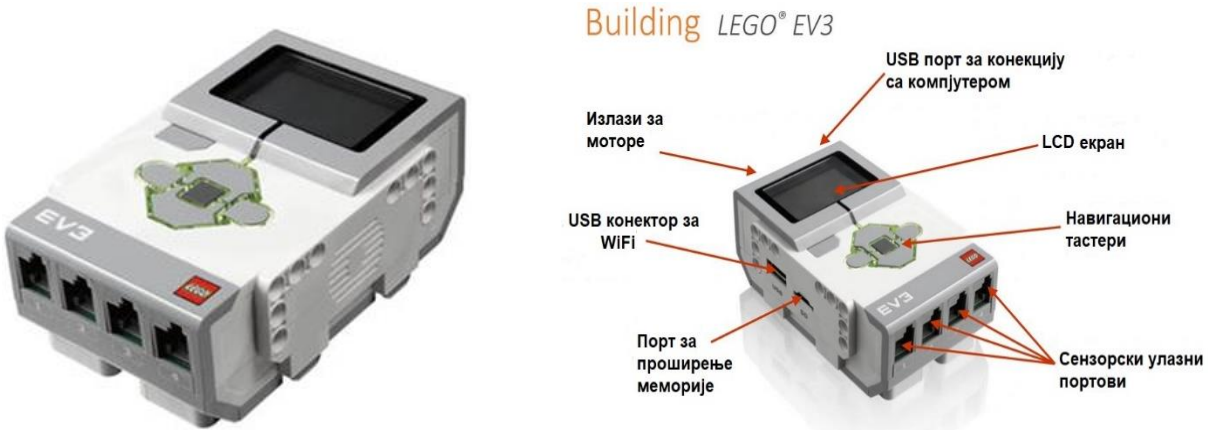


Слика 5.1 Lego Mindstorm EV3 компоненте у сету

Да би робот био мобилан и могао да изврши постављен задатак потребно је употребити следеће елементе EV3 пакета: Управљачка јединица, оптички сензор, погонски систем, носећа конструкција, и наравно напајање које је интегрисано у управљачку јединицу, а састоји се од 6 измењивих AA батерија.

Слика 5.2 Стара и нова генерација управљачке јединице Lego Mindstorm пакета

Главни део поменутог пакета је четвртасти компјутер звани EV3 управљачка јединица (*Intelligent Brick* , слика 5.3).



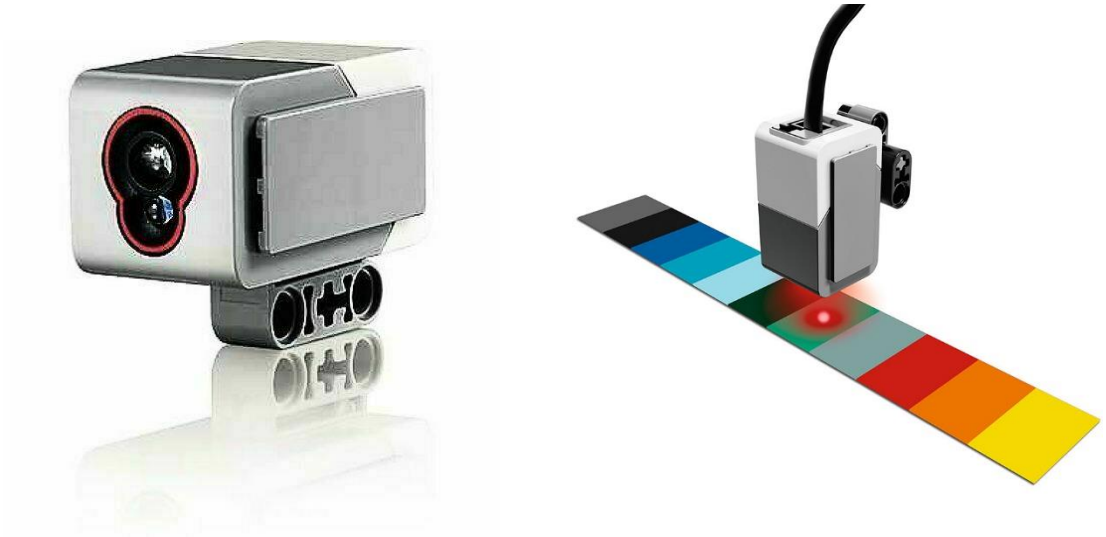
Слика 5.3 Управљачка јединица

Она се може повезати са четири сензора (преко монодирекционих портова), и управља три мотора преко RJ12 кабла који је веома сличан, али није и компатибилан са, телефонским кабловима RJ11. Управљачка јединица има дисплеј са резолуцијом 178×128 пиксела. Тип дисплеја је једнобојни LCD,а на УЈ се налази и пет тастера којима се управља интерфејс хијерархијског менија (Слика 5.4). Такође поседује и звучник са кога се може чути аудио запис. Управљачка јединица поседује могућност тестирања сваког од портова, били

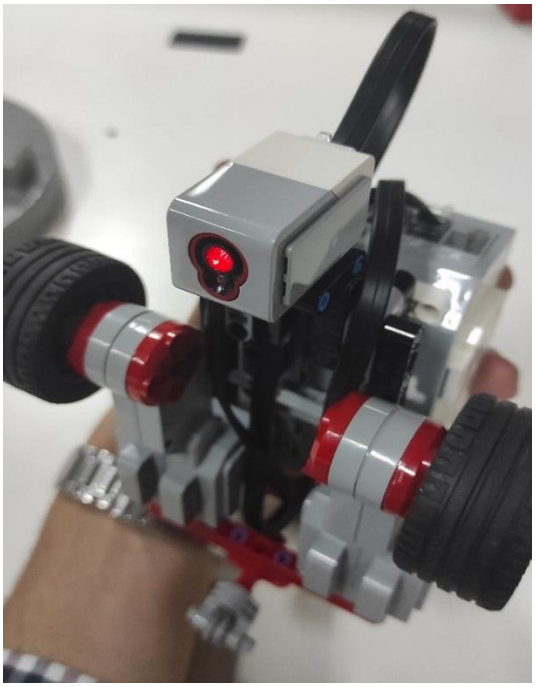
то сензорски портови или портови за покретање мотора, преко имплементираног интерфејса. Овај део је, практично, „мозак“ целог пакета.

Слика 5.4 Дугмад за навођење кроз интерфејс управљачке јединице

Оптички сензор детектује ниво осветљења, а такође поседује и једну LED која служи као извор светлости, која обасјава објекат са ког сензор детектује количину рефлектоване светлости. Сензор може да одреди и јачину амбијенталне светлости. Мерни опсег сензора је од 0 до 100 EV3 јединица, где се вредност 0 добија када је ниво осветљења најмањи, а 100 када је највећи. Проблем овог сензора је у томе што је сам квалитет сензора на јако ниском нивоу, чиме се онемогућује прецизно мерење. У различитим деловима дана очитавања са сензора су драстично промењена за исту боју, јер свака промена осветљења мења рефлексију боје што утиче на очитане резултате. Применом овог сензора, робот уз коришћење вештачких неуронских мрежа кроз примену софтвера MATLAB, може детектовати карактеристичне објекте, и тиме одредити/потврдити своју позицију у познатом окружењу, тј. постављањем референтних карактеристичних објеката црне боје, сензор их може детектовати и послати сигнал УЈ, а она ће дефинисати тренутни положај на основу података прикупљених са енкодера мотора и претходне позиције. Такође, помоћу сигнала са оптичког сензора, може се правити селекција делова који долазе на покретној траци, врсти препреке којој се робот приближава или може разликовати да ли је предмет испред робота препрека, машина или палета са које треба да изврши депалетизацију или палетизацију. Сензор који је коришћен у изради пројекта и његов изглед представљен је на слици 5.5, а његова позиција на роботу на слици 5.6.



Слика 5.5 Оптички сензор



Слика 5.6. Позиција светлосног сензора на роботу

Овај EV3 пакет може бити проширен са ултразвучним сензором који се купује засебно, а приказан је на слици 5.7. Он може да мери удаљеност до објекта у који је уперен као и да детектује померање објеката из окружења. Има могућност приказивања растојања у центиметрима или инчима. Максимална дистанца коју је у могућности да измери је 233

центиметра са толеранцијом 3 центиметра. Ултразвучни сензор ради тако што одашиње ултразвучне таласе који се одбијају од објекат испред сензора и мери време које је потребно да се талас врати до пријемника сензора.

Слика 5.7 Ултразвучни сензор

На слици 5.8 приказан је изглед тактилног сензора. Он детектује притисак сензора, ударац или отпуштање сензора. У случају да је сензор активиран он региструје вредност 1, док у случају када сензор није активиран као свој излаз даје вредност 0.

Слика 5.8 Тактилни сензор

У комплету се налази и инфрацрвени сензор, који ради по сличном принципу као и ултразвучни сензор. Наиме, на средини сензора се налази одашиљач инфрацрвених зрака, а на странама два пријемника. Сензор одашиље инфрацрвени талас и региструје га назад након његовог одбијања од објекта, и шаље податак о времену одашиљања и пријема

сигнала у управљачку јединицу, која прерачунава дистанцу по принципу времена прелета (ToF, *Time of Flight*). Овај сензор је приказан на слици 5.9.

Слика 5.9 Инфрацрвени сензор

EV3 комплет поседује три серво мотора, од којих су два са осом ротације управном на подужну осу мотора (Слика 5.10а), а један има осу ротације концентричну са осом мотора (Слика 5.10б). Они имају интегрисан редуктор са интерним оптичким обртним енкодером који има инкремент мерења од једног степена.

Слика 5.10 а,б: Изгледи мотора

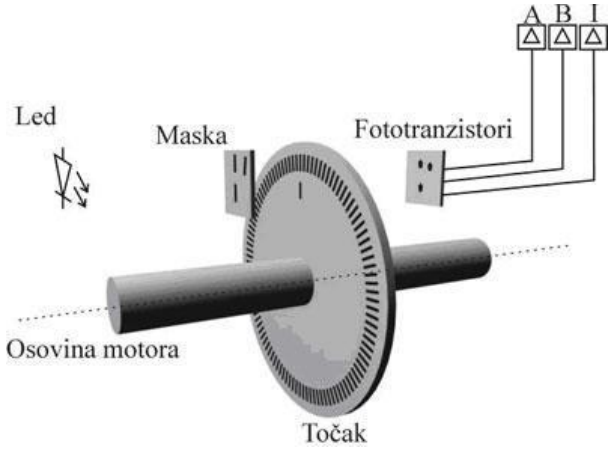
5.1. Детаљан опис коришћених сензора

Сензори који су примењени за израду овог пројекта су оптички сензор и интегрисани мерни елемент мотора.

На слици 5.11 је представљен редуктор и енкодер мотора. Енкодери представљају класу дигиталних сензорских система који се користе за мерења линијских и угаоних помераја. Највише су у примени оптички сензори. Могу се класификовати у две групе, и то у апсолутне и релативне енкодере. Апсолутни енкодери мере апсолутну позицију, док енкодери мере релативан положај у односу на своју претходну позицију.

Слика 5.11 Приказ редукционог система и енкодера мотора

Инкрементални енкодер ради по принципу генерисања импулса приликом пропуштања светлости на сваком прозрачном пољу подеоног диска енкодера. На тај начин, уз познавање тачног броја поља, уз услов да су поља истих димензија, може се одредити угаона позиција вратила на коме се налази енкодер. Прекидом зрака подеоним диском фотодетектор генерише на излазу сигнал који бива дискретизован и тиме је омогућена даља обрада сигнала. На слици 5.12 види се приказ једног типичног енкодера.



Слика 5.12: Шематски приказ инкременталног енкодера

Оптички сензор се најчешће састоји од предајника и пријемника. То је обично LED као извор светлости и пријемник. Светлосни зрак се емитује са диоде и рефлектује од површину. Ту рефлексију прикупља пријемник, обично фототранзистор. Шематски приказ је на слици 5.13

Како је оптички сензор у радном моду за мерење рефлектоване светлости у могућности да детектује само нијансе сиве, на слици 5.13 је дата слика сензора током рада.

Слика 5.13 Радни режим оптичког сензора за мерење рефлектоване светлости

6. Модел кретања

Један од основних проблема који је потребно описати при раду са мобилним роботом је прелазак из једног стања, одређеном врстом кретања (ротацијом или транслацијом), у друго, одабрано стање. При таквом кретању, није у интересу узрок (сила) него веза физичких величина које карактеришу ова два стања. Дакле, овај проблем је кинематичке природе и потребно је основним математичким апаратом доћи до њих. Модел кретања представља скуп једначина које у себи садрже информацију о вези између кинематичких величина које описују два стања у произвољном тренутку.

Вектор стања система у општем случају чине следеће компоненте

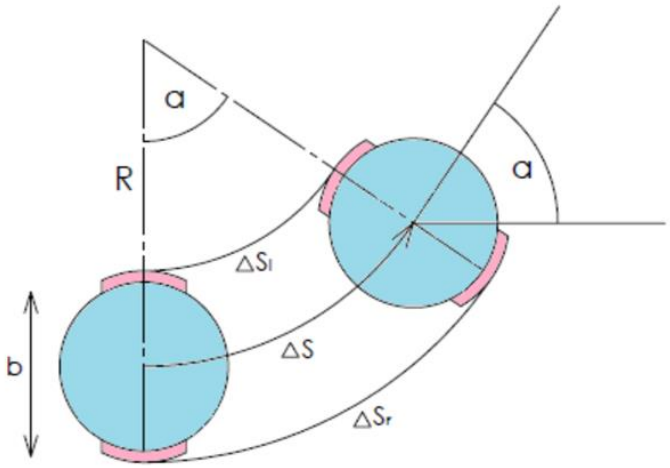
$$x_t = \{x, y, z, \theta, \psi, \varphi\} \tag{6.1}$$

где су *x, y* и *z* компоненте које дефинишу позицију мобилног робота, а *θ, ψ* и *φ* углови који одређују оријентацију мобилног робота у односу на неки спољашњи координатни систем.

Међутим, кретање мобилног робота у оквиру овог пројекта биће раванско, што значи да га је могуће описати са три степена слободе тј. са вектором стања:

$$x_t = \{x, y, \theta\} \tag{6.2}$$

где су *x* и *y* координате тежишта мобилног робота у односу на неки спољашњи систем, а угао *θ* оријентација робота у односу на *x* осу.



Слика 6.1 Шематски приказ модела кретања

Проблем описивања промене стања при кретању мобилног робота представљен је једноставним моделом промене стања тј. моделом кретања који је идејно дат релацијом:

$$p(x_t \mid x_{t-1}, u_t) \tag{6.3}$$

Овај израз говори да је стање робота у тренутку t одређено условном вероватноћом, тј. вероватноћом да он буде у стању x_t уколико је био у стању x_{t-1} и након њега остварио управљање u_t.

6.1 Модел кретања на основу пређеног пута

У овом моделу управљање u_t дефинисано је пређеним путем точкова мобилног робота током тачно дефинисаног временског интервала. На основу очитавања углова са енкодера точкова, може се једноставним математичким трансформацијама доћи до одговарајућег пређеног пута левог и десног точка. Одавде може да се закључи да је пређени пут заправо (индиректно) сензорска информација и да је резултат управљања и рада мотора.

Модел кретања изведен на основу пређеног пута мобилног робота може се применити искључиво на оне роботе који имају два погонска точка која се могу окретати потпуно независно један од другог. Самим тим, могуће је управљањем смером обртања точкова контролисати скретање мобилног робота, а помоћу енкодера на осовинама точкова могу се измерити углови ротације (обртања) точкова. С обзиром да је у питању раванско кретање робота, као што је и речено, може бити описано вектором положаја:

$$x_t = \{x, y, \theta\} \tag{6.4}$$

С обзиром да је помоћу модела кретања потребно одредити вектор положаја у тренутку t+1, видимо да је потребно наћи везу између позиције и оријентације у тренуцима t и t+1.

Са слике 6.1 је могуће уочити релације

$$\Delta s_l = (R - \frac{b}{2})\Delta\theta \tag{6.5}$$

$$\Delta s_d = (R + \frac{b}{2})\Delta\theta \tag{6.6}$$

$$\Delta s = R\Delta\theta \tag{6.7}$$

из којих се решавањем може добити следећа веза 6.8 представља пређени пут центра масе робота:

$$\Delta s = \frac{\Delta s_l + \Delta s_d}{2} \tag{6.8}$$

Такође, уколико се посматрају угао *α* и једнакостранични троуглови над њим, може се уочити веза између пређених путева левог и десног точка. Потребно је посматрати једнакостранични троугао са страницама R, углом *α* и основицом Δs_l и једнакостранични угао са истим страницама и углом над основицом али са основицом Δs_d. С обзиром да при ротацији за мале углове важи да је пређени део пута по кружници над углом *α* : *s = r * α* и да су углови над оба троугла исти види се да важи следећа релација:

$$\frac{\Delta s_l}{R-\frac{b}{2}} = \frac{\Delta s_d}{R+\frac{b}{2}} \tag{6.9}$$

Из које се на основу релација 6.5 и 6.6 добија:

$$\Delta\theta = \frac{\Delta s_l - \Delta s_d}{b} \tag{6.10}$$

Уз апроксимацију $\Delta s \approx \Delta d$ лако се види:

$$\Delta x = \Delta s \cos(\theta + \frac{\Delta\theta}{2}) \tag{6.11}$$

$$\Delta y = \Delta s \sin(\theta + \frac{\Delta\theta}{2}) \tag{6.12}$$

Где Δx представља растојање дуж *x* осе центра масе робота између два положаја робота, а Δy дуж осе *y*.

Комбинујући формуле 6.10, 6.11 и 6.12 могу се добити позиција и оријентација у следећем положају, x', y' и θ'. У формули се налази вектор стања мобилног робота након извршеног управљања:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{\Delta s_d + \Delta s_l}{2} \cos(\theta + \frac{\Delta\theta}{2}) \\ \frac{\Delta s_d + \Delta s_l}{2} \sin(\theta + \frac{\Delta\theta}{2}) \\ \frac{\Delta s_d - \Delta s_l}{b} \end{bmatrix} \tag{6.13}$$

Она представља основни кинематички модел кретања мобилног робота изведен на основу пређених путева левог и десног погонског точка. Из ове једначине видимо да на положај у наредном тренутку утиче промена оријентације робота, почетни угао оријентације робота, пређени путеви левог и десног точка као и претходна позиција.

7. Вештачке неуронске мреже

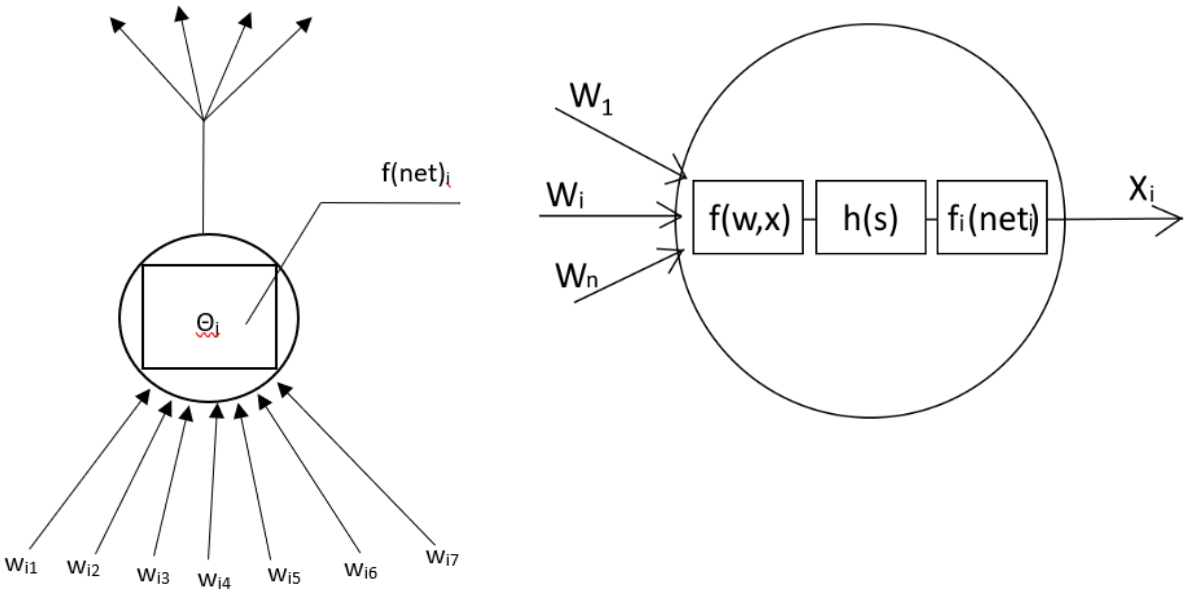
Неуронска мрежа је општа математичка рачунарска парадигма која моделира деловање биолошких неуронских система [1]. Примењује се при моделирању система који се због своје комплексности не могу описати општим моделом. У општем смислу, неуронске мреже представљају скуп једноставних процесирајућих елемената неурона, међусобно повезаних везама са одговарајућим тежинским односима [2]. Оно што разликује неуронске мреже од општих модела је то да оне могу да уче, тј. да памте пресликавање између улазног и излазног податка на одређеном обучавајућем узорку, као и да ту адаптивност прошире на податке које се не налазе у узорку. Обучавајући скупови се пажљиво бирају, тако да буду довољно велики и репрезентативни, а опет да не садрже превише података и успоре процес обучавања.

Основни елементи мреже су:

1. Неурон
2. Топологија
3. Алгоритам учења

Додатни елементи мреже су:

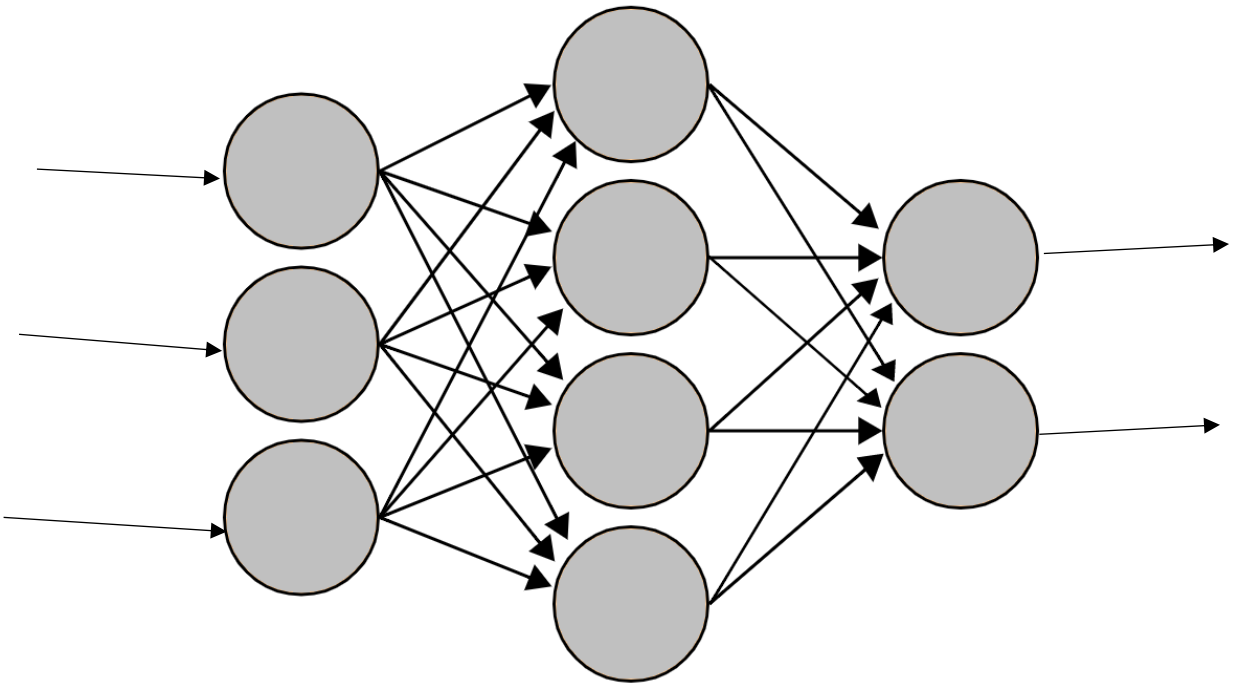
- Величина мреже
- Функционалност неурона
- Обучавање/валидност
- Имплементација



Слика 7.1 Приказ неурона

Табела 2. Делови неуронске мреже

w_{i1}, w_{11}, w_{in}	тежински односи
h(s)	функција преноса
$f_i=(net_i)$	активациона функција
x_i	излаз из неурона



Слика 7.2 Вештачка неуронска мрежа са једним скривеним слојем

У овом раду коришћено је два типа неуронских мрежа:

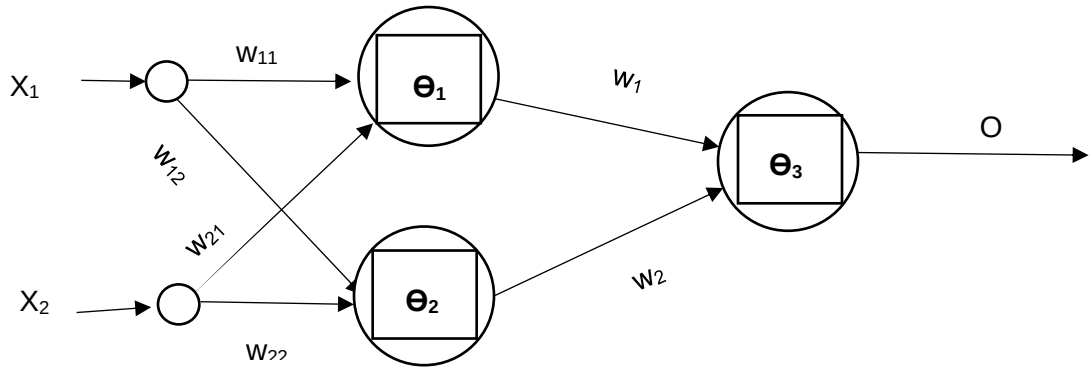
- Backpropagation (BP) неуронска мрежа, за обучавање ротације ИМР и
- Перцептрон, за обучавање препознавања боја

Backpropagation (BP)

Број неурона и скривених слојева варира у зависности од проблема на којем се примењује. Основни облик архитектуре BP неуронске мреже има један улазни, скривени и излазни слој. Осим тога, врло често оваква мрежа може да садржи још један додатни неурон, „Bias“, који не припада ни једном слоју, увек је активан и има излазно стање 1. Он се понаша у мрежи као и сваки други неурон, при чему има задатак да партиципира у процесу учења тако што тежински однос између „Bias“ неурона и сваког неурона у следећем слоју формира активацију која мора бити савладана остатком улаза у сваки неурон. На тај начин се контролише активација свих неурона. BP су способне да генерализују, па ако им се презентује некомплетан скуп улазних података, мрежа ће ипак бити у стању да генерише излаз.

Перцептрон

У савременом смислу, перцептрон је алгоритам за учење бинарног класификатора: функција која пресликава улазни вектор у излазну вредност [3]. Као линеарни класификатор, перцептрон је најједноставнији облик неуронске мреже са простирањем сигнала унапред (енг. *Feedforward neural network*).



Слика 7.3 Перцептрон

Перцептрон, као линеарни сепаратор, има велика ограничења у погледу нелинеарних пресликавања. То значи да је проблем његове примене везан за нелинеарно раздвајање класа, јер он користи бинарну активациону функцију, која условљава да неуронска мрежа има дисконтинуитете [4]. Иако је једноставнија вештачка неуронска мрежа, за потребе једноставне класификације података он се и даље може успешно користити и налази се као део *Matlab Neural Network toolbox*-а.

7.1 Обучавање мрежа за ротацију и препознавање боја

Прво су прикупљени подаци о реалним угловима ротације робота. Наиме, због грешака услед различитих фактора, робот се при задатом углу ротације никад неће заротирати за тај угао. Део података о реалном углу ротације робота налазе се у табели 3.

Табела 3. Обучавајући парови за ротацију

прикупљени подаци	1	2	3	4	5	6	7	8	9
θ остварено	-153	-144	-128	-100	-77	-52	-22.5	-13.5	-5
θ задато	-180	-170	-150	-120	-90	-60	-30	-20	-10

С обзиром на то да се ради о непознатој функционалности између задатог и оствареног угла, погодно је употребити вештачке неуронске мреже. Адекватним одабиром улазних и излазних вредности, могуће је обучити мрежу да се понаша као стваран систем (у овом случају робот), тј. могуће је пронаћи функционалност тако да за било коју нову улазну вредност добро предвиђа потребну излазну, на основу те функционалности.

Табела 4. Подаци о обученим мрежама за ротацију

Редни бр.	Бр. неурона у слоју	Активационе функције	Алгоритам учења	Параметар учења	грешка обучавања	грешка верификације	Најбоља епоха	Укупно епоха	Време тренирања
1	1-[8,8]$_2$-1	logsig,poslin	trainlm	0,3	1,885E-04	1,400E-03	5	11	0,5737
2	1-[8,8]$_2$-1	logsig,logsig	trainlm	0,3	9,158E-06	2,300E-03	9	9	0,5296
3	1-[8,8]$_2$-1	purelin,poslin	trainlm	0,3	1,110E-04	6,619E-05	5	9	0,5419
4	1-[8,8]$_2$-1	poslin,poslin	trainlm	0,3	3,350E-05	2,110E-04	5	11	0,5112
5	1-[8,8]$_2$-1	purelin,purelin	trainlm	0,3	4,440E-05	1,610E-04	4	5	0,5224
6	1-[5,5]$_2$-1	purelin,poslin	trainlm	0,3	2,870E-05	1,110E-04	3	6	0,4732
7	1-[12,12]$_2$-1	purelin,poslin	trainlm	0,3	2,880E-05	1,120E-04	7	9	0,5205
8	1-[12]$_1$-1	purelin	trainlm	0,3	4,710E-05	1,420E-04	2	5	0,5312
9	1-[24]$_1$-1	purelin	trainlm	0,3	4,450E-05	1,610E-04	5	5	0,5015
10	1-[16,8,8]$_3$-1	purelin,poslin,poslin	trainlm	0,3	9,830E-06	5,870E-05	10	10	0,9168
11	1-[16,16,16]$_3$-1	purelin,poslin,poslin	trainlm	0,3	1,960E-06	1,980E-04	4	4	0,4997
12	1-[8,8]$_2$-1	purelin,poslin	traingdx	0,3	1,780E-04	3,960E-04	57	64	0,5826
13	1-[8,8,8]$_3$-1	tansig,purelin,poslin	trainlm	0,3	1,980E-05	8,100E-03	5	11	0,6663
14	1-[16,16]$_2$-1	purelin,poslin	trainlm	0,3	3,040E-05	1,190E-04	9	15	0,7739
15	1-[16,4,16]$_3$-1	purelin,logsig,poslin	trainlm	0,3	6,320E-05	1,720E-04	5	11	0,646
16	1-[8,8]$_2$-1	purelin,poslin	trainlm	0,3	2,680E-05	2,150E-04	11	12	0,9228
17	1-[20,16]$_2$-1	purelin,poslin	trainlm	0,3	2,990E-05	8,790E-05	4	10	0,7094
18	1-[12,6]$_2$-1	purelin,poslin	trainlm	0,3	3,970E-05	1,580E-04	4	7	0,8092
19	1-[16,8,8]$_3$-1	purelin,purelin,poslin	trainlm	0,3	2,98E-05	1,780E-04	4	10	0,728
20	1-[16,4,16]$_3$-1	purelin,poslin,poslin	trainlm	0,3	2,13E-04	6,030E-04	2	8	0,806
21	1-[16,8,8]$_3$-1	poslin,poslin,poslin	trainlm	0,3	1,72E-05	1,63E-04	6	10	0,6168
22	1-[8,4,4]$_3$-1	purelin,poslin,poslin	trainlm	0,3	1,99E-05	7,11E-05	7	10	0,7005
23	1-[8,8,4]$_3$-1	purelin,poslin,poslin	trainlm	0,3	5,14E-05	9,37E-05	3	9	0,68
24	1-[12,10,6]$_3$-1	purelin,poslin,poslin	trainlm	0,3	9,12E-06	1,82E-04	6	10	0,6457
25	1-[30,15,15]$_3$-1	purelin,poslin,poslin	trainlm	0,3	3,90E-05	2,86E-04	4	8	0,941
26	1-[16,8,8]$_3$-1	purelin,poslin,poslin	trainlm	0,3	8,94E-05	5,17E-05	5	8	0,5841
27	1-[16,8,8]$_3$-1	purelin,poslin,poslin	traingdx	0,3	1,97E-04	1,55E-04	64	70	0,767
28	1-[16,8,8]$_3$-1	purelin,poslin,poslin	trainrp	0,3	2,91E-04	1,13E-04	18	24	0,7337
29	1-[16,8,8]$_3$-1	purelin,poslin,poslin	trainbr	0,3	6,05E-05	9.75E-5	29	30	1,3403
30	1-[16,8,8]$_3$-1	purelin,poslin,poslin	trainscg	0,5	9,34E-05	2,52E-03	14	20	0,656
31	1-[16,8,8]$_3$-1	purelin,poslin,poslin	trainlm	0,10	5,930E-06	7,04E-05	9	12	0,6655
32	1-[16,8,8]$_3$-1	purelin,poslin,poslin	trainlm	0,01	3,168E-05	3,98E-05	3	6	0,6224
33	1-[16,8,8]$_3$-1	purelin,poslin,poslin	trainlm	0,01	1,930E-04	1,56E-04	4	6	0,6013
34	1-[16,8,8]$_3$-1	purelin,poslin,poslin	trainlm	0,03	7,590E-05	2,17E-05	7	13	0,7157
35	1-[16,8,8]$_3$-1	purelin,poslin,poslin	trainlm	0,03	1,150E-05	3,05E-04	15	20	0,8652

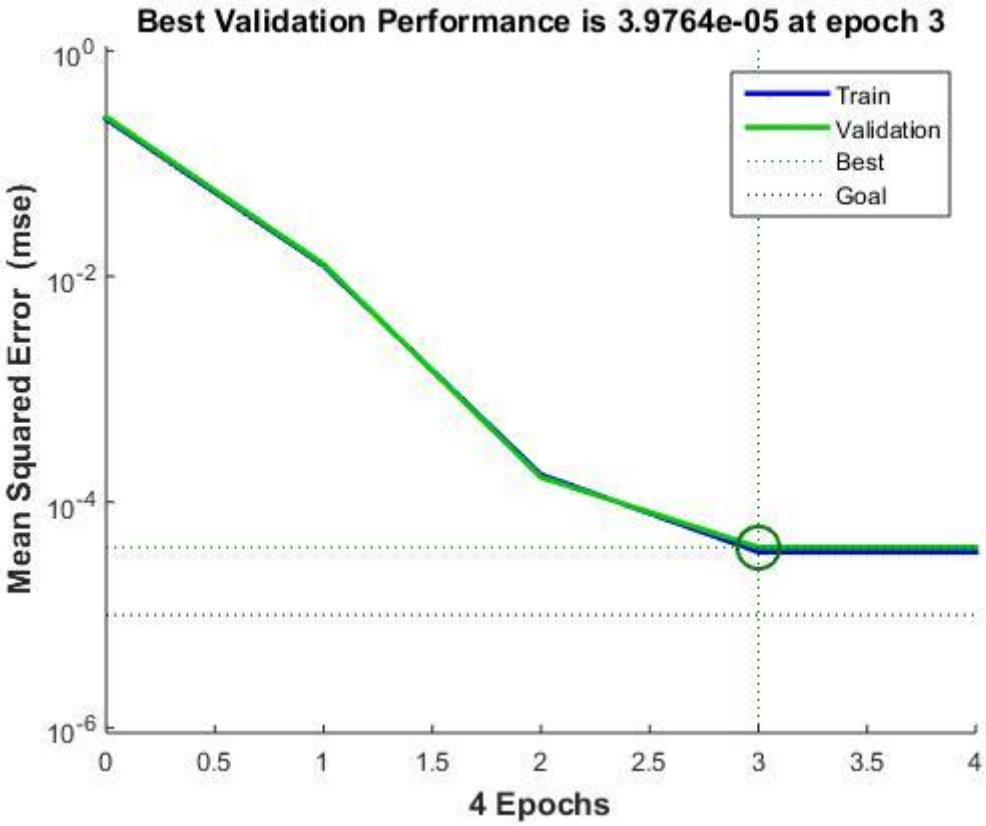
Улазни вектор чине вредности угла које је робот остварио *θостварено*, а излазни вектор вредности угла које су биле задате, *θзадато*.

Вештачка неуронска мрежа која је коришћена у даљем експерименту одабрана је на основу најмањих вредности грешака обучавања скупа за тренирања, *грешка обучавања* и валидацију *грешка валидације.*

Скуп обучавајућих парова подељен је на скуп за тренирање и валидацију, да не би дошло до преобучавања над тренинг скупом. Валидација служи за тестирање за време обучавања те побољшава само тренирање мреже.

```
net.divideFcn = 'divideind';
net.divideParam.trainInd = [1,3,4,6,7,9,10,12,13,15,16,18];
net.divideParam.valInd = [2,5,8,11,14,17];
```

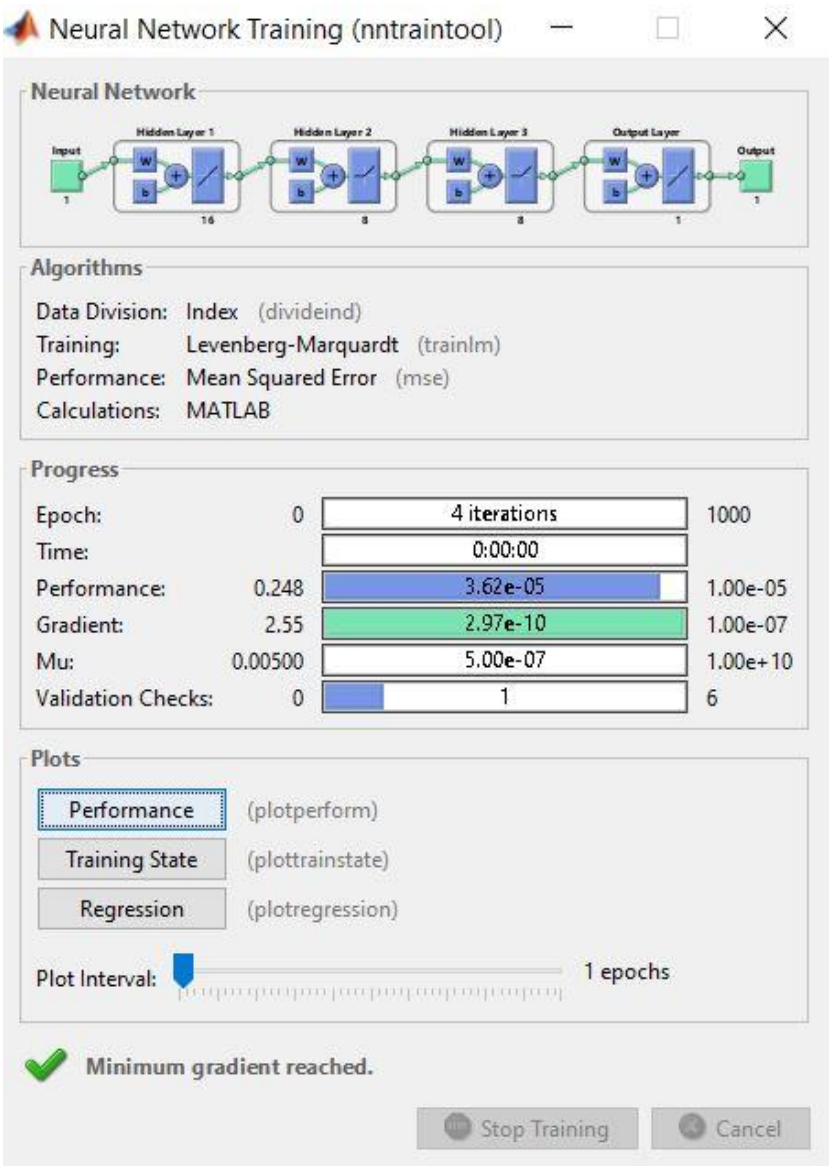
Из табеле 3. види се да је најбоља архитектура вештачке неуронске мреже са три скривена слоја. У првом скривеном слоју налази се 16 неурона, у другом 8 а у трећем такође 8. Коришћене су *purelin* и *poslin* активационе функције. Алгоритам обучавања био је *trainlm,* а параметар учења 0.01 (мрежа 32 у табели 4.)



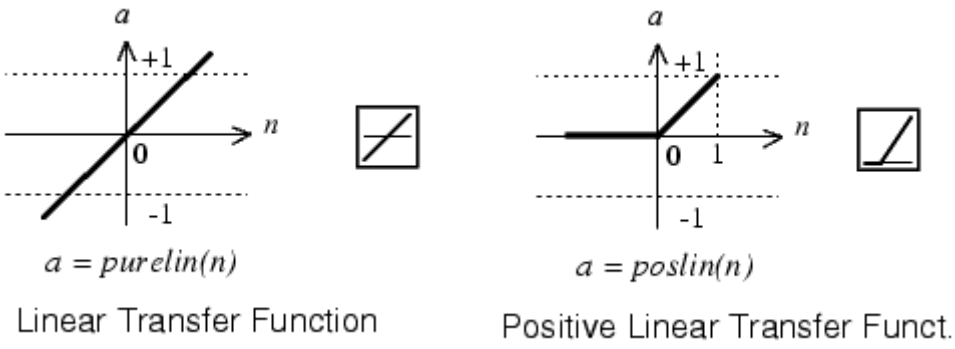
Слика 7.4 График валидационе криве током тренирања мреже за ротацију

Field	Value
trainFcn	'trainlm'
trainParam	*1x1 struct*
performFcn	'mse'
performParam	*1x1 struct*
derivFcn	'defaultderiv'
divideFcn	'divideind'
divideMode	'sample'
divideParam	*1x1 struct*
trainInd	*1x12 double*
valInd	[2,5,8,11,14,17]
testInd	*[ ]*
stop	'Minimum gradient r...
num_epochs	4
trainMask	*1x1 cell*
valMask	*1x1 cell*
testMask	*1x1 cell*
best_epoch	3
goal	1.0000e-05
states	*1x8 cell*
epoch	[0,1,2,3,4]
time	[0.1960,0.3190,0.3380,...
perf	[0.2476,0.0124,1.7423...
vperf	[0.2580,0.0128,1.6544...
tperf	[NaN,NaN,NaN,NaN,...
mu	[0.0050,5.0000e-04,5....
gradient	[2.5548,0.7381,0.0765,...
val_fail	[0,0,0,0,1]
best_perf	3.6168e-05
best_vperf	3.9764e-05
best_tperf	NaN

Слика 7.5 Параметри најбоље обучене мреже ротације



Слика 7.6 Резултат тренирања најбоље мреже за ротацију (са скалираним величинама)



Слика 7.7 Активационе функције *purelin* и *poslin*

Обучавање мреже за класификацију црне или беле боје

Као улазни вектор узети су подаци са оптичког сензора о интензитету рефлектоване светлости. С обзиром да црна површина апсорбује светлост, количина рефлектоване светлости са њене површине је мала, а пошто бела површина рефлектује светлост, количина рефлектоване светлости је велика. Као излазни вектор дефинисане су комбинације [1 0] за црну и [0 1] за белу боју.

Табела 5. Обучавајући парови за препознавање боја

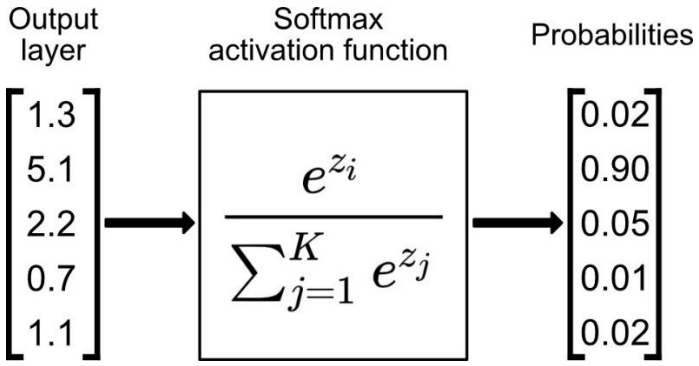
Интензитет рефлектоване светлости	7	5	6	5	6	47	82	67	66
Црна/ бела боја	1	1	1	1	1	0	0	0	0
	0	0	0	0	0	1	1	1	1

Табела 6. Подаци о обученим мрежама за препознавање боја

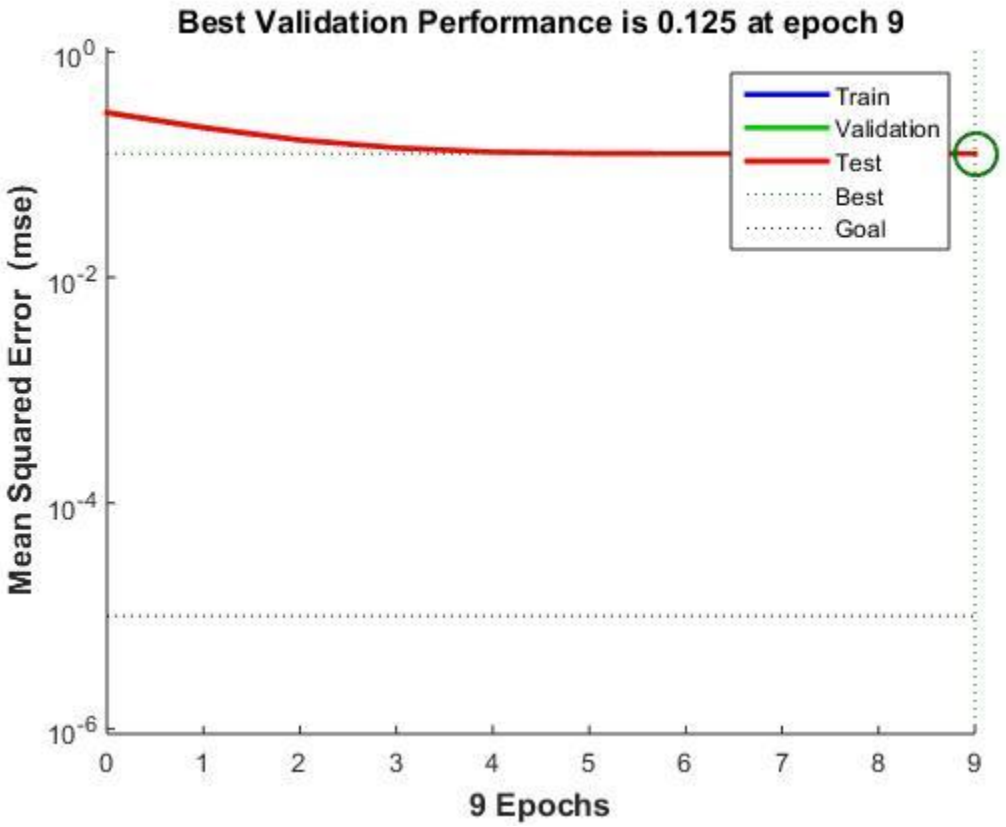
Редни бр.	Бр. неурона у слоју	Активационе функције	Алгоритам учења	Параметар учења	грешка обучавања	грешка верификације	Најбоља епоха	Укупно епоха	Време тренирања
1	1-[20,10]$_2$-1	{'logsig','logsig','softmax'}	traingdx	0.40	0.12500552	0.125004592	182	182	0.8072
2	1-[20,10]$_2$-1	{'logsig','logsig','softmax'}	trainlm	0.40	0.125001102	0.125001102	6	9	0.4349
3	1-[20,10]$_2$-1	{'logsig','logsig','softmax'}	trainscg	0.40	0.125000448	0.125000426	25	25	0.4094
4	1-[20,10]$_2$-1	{'logsig','logsig','softmax'}	trainbr	0.40	0.125	NaN	3	3	0.5506
5	1-[20,10]$_2$-1	{'logsig','logsig','softmax'}	trainrp	0.40	0.125000525	0.12500037	10	10	0.3661
6	1-[20,10]$_2$-1	{'tansig,'tansig','softmax'}	trainrp	0.40	0.125000242	0.12500024	5	5	0.3489
7	1-[20,10]$_2$-1	{'tansig','purelin','softmax'}	trainrp	0.40	0.125	0.125	3	3	0.2892
8	1-[20,10]$_2$-1	{'tansig','satlin','softmax'}	trainrp	0.40	0.125000583	0.125000261	10	10	0.3472
9	1-[20,10]$_2$-1	{'purelin','purelin','softmax'}	trainrp	0.40	0.125000187	0.125000196	2	2	0.275
10	1-[20,10]$_2$-1	{'purelin','poslin','softmax'}	trainrp	0.40	0.125000042	0.125000001	6	6	0.3434
11	1-[10,8]$_2$-1	{'purelin','purelin','softmax'}	trainrp	0.40	0.125000009	0.125	7	7	0.3712
12	1-[8,4]$_2$-1	{'purelin','purelin','softmax'}	trainrp	0.40	0.125000187	0.125000187	5	5	0.3715
13	1-[4,2]$_2$-1	{'purelin','purelin','softmax'}	trainrp	0.40	0.125000091	0.125000058	7	7	0.3449
14	1-[4]$_1$-1	{'purelin','softmax'}	trainrp	0.40	0.12500133	0.125000001	9	9	0.3574
15	1-[6]$_1$-1	{'purelin','softmax'}	trainrp	0.40	0.12500266	0.125001468	10	10	0.3479
16	1-[6]$_1$-1	{'purelin','softmax'}	trainrp	0.05	0.125000163	0.125	10	10	0.3338
17	1-[6]$_1$-1	{'purelin','softmax'}	trainrp	0.10	0.125001117	0.125	8	8	0.3325
18	1-[6]$_1$-1	{'purelin','softmax'}	trainrp	0.40	0.125000026	0.125000036	8	8	0.3335
19	1-[6]$_1$-1	{'purelin','softmax'}	trainrp	0.40	0.12500048	0.125	13	13	0.3624
20	1-[6]$_1$-1	{'purelin','softmax'}	trainrp	0.40	0.125000482	0.125	11	11	0.3369

Слично као и код обучавања мреже за ротацију, разматрана је вредност грешке тренирања и валидације. С обзиром да су за сваку мрежу оне исте, посматрали смо критеријум времена тренирања. С обзиром да је за мрежу 17 у табели 6. време тренирања минимално, одабрали смо њу за даљи рад у експерименту.

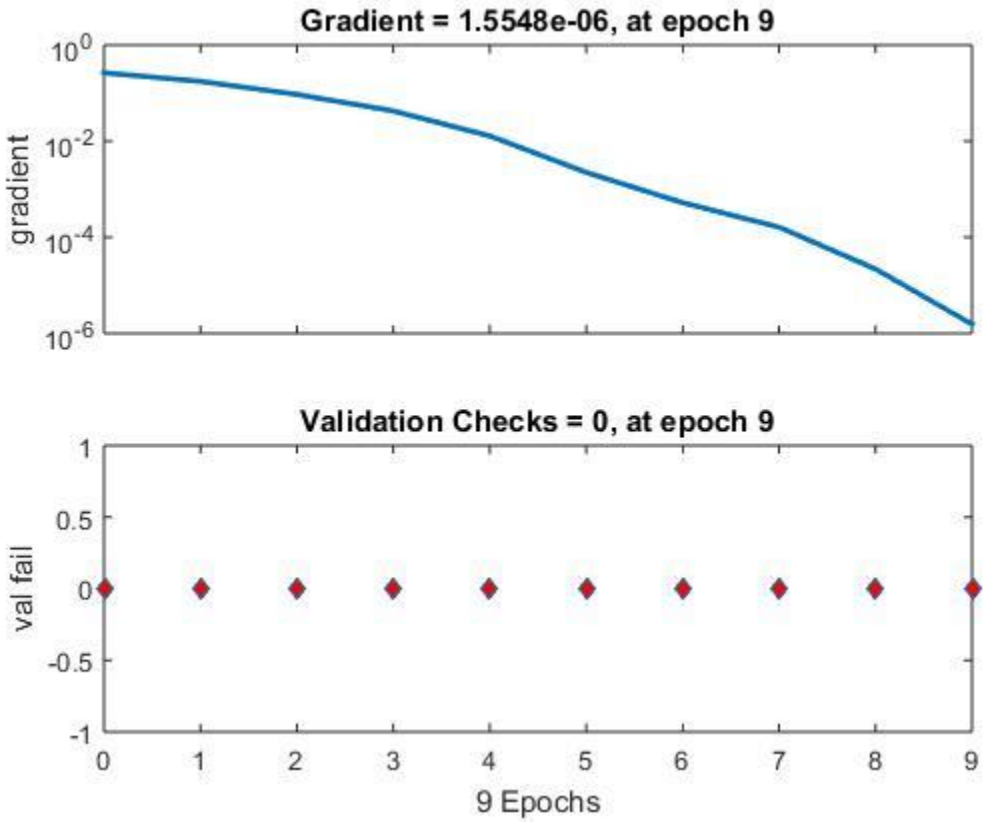
Архитектура одабране вештачке неуронске мреже има 1 скривени слој при чему је активациона функција неурона *purelin* а излазног слоја *softmax* с обзиром да је потребно извршити класификацију улазних података. Параметар учења је 0.1 а време обучавања 0.3325.



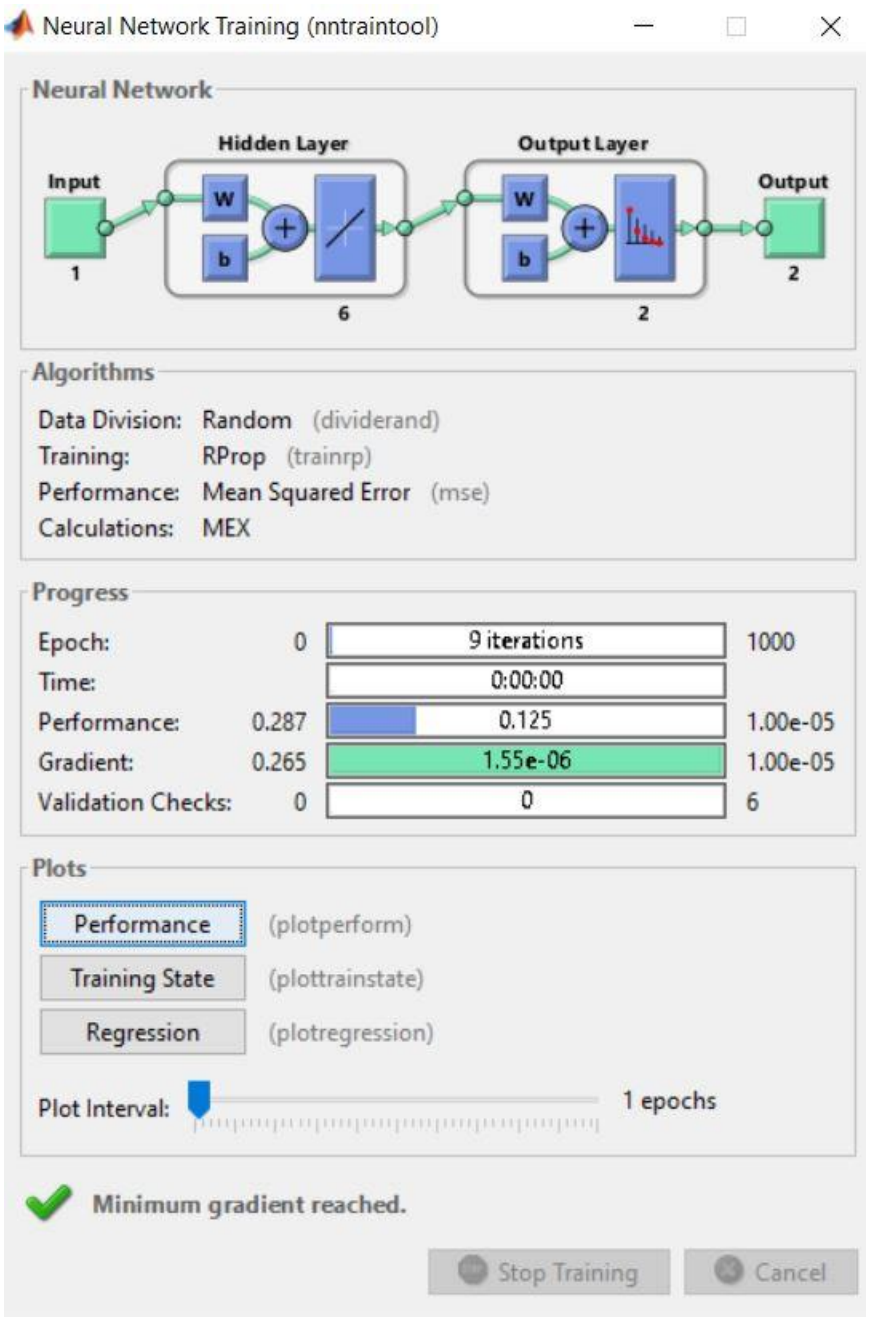
Слика 7.8 Активациона функција *Softmax*



Слика 7.9 График валидационе криве током тренирања мреже за препознавања боја



Слика 7.10 Дијаграми различитих параметара током епоха



Слика 7.11 Резултат тренирања најбоље мреже за препознавање боја

8. Калманов филтер

Калманов филтер добио име по Рудолфу Е. Калману, који је представио овај математички модел у свом раду 1960. године. Калманов филтер уз помоћ вероватноће и статистике, решава проблем процене стања система када се оно не може директно измерити или проблем одређивања стања система на основу комбинације мерења са различитих извора, а на која утичу разне сметње и шумови [10].

Оцењивање стања, тј. естимација стања система представља процес оцењивања параметара стања или самог стања система на основу прикупљених информација и можемо је представити преко следеће три функције:

- **Интерполација** - прикупљање што више валидних информација о неком процесу или систему.
- **Филтрација** - процес који за циљ има одређивање удела сигнала и шума у сензорској информацији (у сигналу који је сензор прикупио).
- **Предикција** - на основу свих прикупљених информација до тренутка t генерише се предвиђање стања система, или само једног елемента система, за тренутак t_1. Како је од интереса да предикција буде што прецизнија, вредно је напоменути да уз предикцију иде и корекција.

8.1 Формулација Калмановог филтера

Калманов филтер подразумева да се линерани динамички систем представи у форми једначина које описују стање система и да се над тим једначинама изведе рекурзивна естимација. Естимација је рекурзивна зато што се свака процена стања система израчунава из најскорије претходне процене и нових улазних података. Чињеница да нам је за израчунавање процене потребна само најскорија претходна нам указује на непотребност чувања свих израчунатих процена и на тај начин се повећава ефикасност рада овог модела.

Сходно овом исказу имамо следећу изведену једначину стања система:

$$x_t = A_t \cdot x_{t-1} + B_t \cdot u_t + \varepsilon_t \tag{8.1}$$

$$p(x_t|u_t,\, x_{t-1}) = N(x_t; A_t \cdot x_{t-1} + B_t \cdot u_t,\, R_t) \tag{8.2}$$

и једначину мерења (опсервациони модел):

$$z_t = C_t \cdot x_t + \delta_t \tag{8.3}$$

$$p(z_t|x_t) = N(z_t;\, C_t \cdot x_t, Q_t) \tag{8.4}$$

где су:

p- вероватноћа да случајна величина z припада расподели z_t под условом x_t

x_t- вектор стања система у тренутку t;

A_t- матрица која одређује како систем од тренутка $t-1$ долази до t без утицаја управљања или шумова;

x_{t-1}- вектор стања система у тренутку $t-1$;

B_t- матрица која одређује како управљање мењаја стање система од $t-1$ до t;

u_t- матрица управљања помоћу које дефинише како систем прелази из стања $t-1$ у стање t;

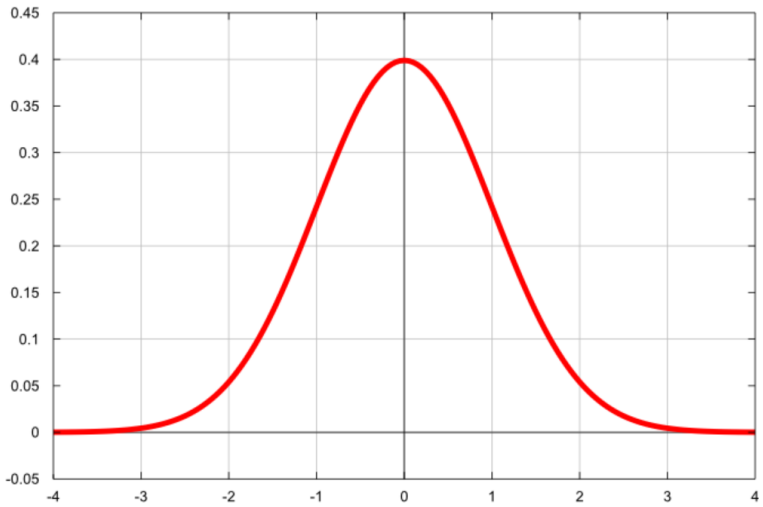
ε_t- шум система, бели шум, подлеже Гаусовој расподели са нултом средњом вредношћу са матрицом коваријанси R_t;

N- нормална расподела

z_t - вектор мерења у тренутку t;

C_t- матрица која дефинише пресликавање x_t у мерење z_t;

δ_t- шум мерења, бели шум, подлеже Гаусовој расподели са нултом средњом вредношћу са матрицом коваријанси Q_t, независно од ε_t;



Слика 8.1 Гаусова (нормална) расподела са нултом очекиваном вредношћу

8.2 Алгоритам Калмановог филтера

Калманов филтер је формулисан према следећем алгоритму ($\mu_{t-1}, \Sigma t-1, u_t, z_t$):

$$1.\ \underline{\mu_t} = A_t \cdot \mu_{t-1} + B_t \cdot u_t \tag{8.5}$$

$$2.\ \underline{\Sigma_t} = A_t \cdot \Sigma_{t-1} \cdot A_t^T + R_t \tag{8.6}$$

$$3.\ K_t = \underline{\Sigma_t} \cdot C_t^T \cdot (C_t \cdot \underline{\Sigma_t} \cdot C_t^T + Q_t)^{-1} \tag{8.7}$$

$$4.\ \mu_t = \underline{\mu_t} + K_t \cdot (z_t - C_t \cdot \underline{\mu_t}) \tag{8.8}$$

$$5.\ \Sigma_t = (I - K_t \cdot C_t) \cdot \underline{\Sigma_t} \tag{8.9}$$

return μ_t , Σ_t;

где су:

$\underline{\mu_t}$- предвиђено стање система,

μ_t- очекивана вредност расподеле,

μ_{t-1} - положај робота у претходној итерацији,

$\underline{\Sigma_t}$- предвиђена матрица коваријанси стања система,

Σ_t- матрица коваријанси стања система,

Σ_{t-1}- матрица коваријанси у претходном тренутку,

R_t- матрица коваријанси шума система,

K_t- матрица Калмановог појачања,

Q_t- матрица коваријанси шума мерења,

I- јединична матрица.

8.3 Линеаризовани Калманов филтер (проблем који решава)

Уколико имамо случај да је систем нелинеаран, то јест садржи бели шум у неком од модела који описују његово стање, онда је неопходно користити линеаризован Калманов филтер. Једначину стања нелинеарне функције управљања и претходног стања приказујемо следећом једначином :

$$x_t = g(u_t, x_{t-1}) \tag{8.10}$$

Опсервациони модел нелинеарне функције стања система се описује следећом једначином:

$$Z_t = h(x_t) \tag{8.11}$$

Да би се применио Линеаризовани Калманов филтер, неопходно је најпре извршити развој функције у Тејлоров ред на следећи начин:

$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + \frac{\partial g(u_t, \mu_{t-1})}{\partial x_{t-1}} \cdot (x_{t-1} - \mu_{t-1}) \tag{8.12}$$

$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + G_t \cdot (x_{t-1} - \mu_{t-1}) \tag{8.13}$$

$$h(x_t) \approx h(\underline{\mu_t}) + \frac{\partial h(\underline{\mu_t})}{\partial x} \cdot (x_t - \underline{\mu_t}) \tag{8.14}$$

$$h(x_t) \approx h(\underline{\mu_t}) + H_t \cdot (x_t - \underline{\mu_t}) \tag{8.15}$$

где су:

G_t- промена функције модела кретања у односу на положај робота

H_t- промена мерења са променом координата мобилног робота.

У наставку следи алгоритам линеаризованог Калмановог филтера (μ_{t-1}, $\Sigma t - 1$, u_t, z_t):

Алгоритам ЛКФ

1. $\underline{\mu_t} = g(u_t, x_{t-1})$ (8.16)
2. $\underline{\Sigma_t} = G_t \cdot \Sigma_{t-1} \cdot G_t^T + R_t$ (8.17)
3. $K_t = \underline{\Sigma_t} \cdot H_t^T \cdot (H_t \cdot \underline{\Sigma_t} \cdot H_t^T + Q_t)^{-1}$ (8.18)
4. $\mu_t = \underline{\mu_t} + K_t \cdot [z_t - h(\underline{\mu_t})]$ (8.19)
5. $\Sigma_t = (I - K_t \cdot H_t) \cdot \underline{\Sigma_t}$ (8.20)

return μ_t , Σ_t;

Табела 7. Поређење ЛКФ и КФ

		Калманов филтер	Линеаризовани Калманов филтер
Корак предикције	1. Предвиђање стања	$\underline{\mu_t} = A_t \cdot \mu_{t-1} + B_t \cdot u_t$	$\underline{\mu_t} = g(u_t, x_{t-1})$
	2. Предвиђање мерења	$\underline{\Sigma_t} = A_t \cdot \Sigma_{t-1} \cdot A_t^T + R_t$	$\underline{\Sigma_t} = G_t \cdot \Sigma_{t-1} \cdot G_t^T + R_t$
Корак корекције	3. Калманово појачање	$K_t = \underline{\Sigma_t} \cdot C_t^T \cdot (C_t \cdot \underline{\Sigma_t} \cdot C_t^T + Q_t)^{-1}$	$K_t = \underline{\Sigma_t} \cdot H_t^T \cdot (H_t \cdot \underline{\Sigma_t} \cdot H_t^T + Q_t)^{-1}$
	4. Очекивана вредност расподеле	$\mu_t = \underline{\mu_t} + K_t \cdot (z_t - C_t \cdot \underline{\mu_t})$	$\mu_t = \underline{\mu_t} + K_t \cdot [z_t - h(\underline{\mu_t})]$
	5. Матрица коваријанси	$\Sigma_t = (I - K_t \cdot C_t) \cdot \underline{\Sigma_t}$	$\Sigma_t = (I - K_t \cdot H_t) \cdot \underline{\Sigma_t}$

8.4 Примена Калмановог филтера у пројекту

У пројектном задатку, Калманов филтер се користи за решавање проблема локализације. Решавање обог проблема обухвата одређивање позиције и оријентације мобилног робота у односу на неки апсолутни координатни систем који је адекватно постављен на модел његовог окружења, за коју се претпоставља да је апсолутно тачна и да се на њој налазе положаји свих непомичних објеката који се могу наћи у окружењу.

9. A* алгоритам

А* алгоритам је алгоритам претраживања који служи за одређивање најкраће путање између почетне и крајње тачке. Има широку примену, због способности да брзо претражује велике просторе. Уз коришћење хеуристике, постиже значајно боље резултате од својих претходника, коришћених за исту намену. Помоћу хеуристике одређује растојање од почетне до сваке тачке у радном простору и бира пиксел са најмањом вредношћу.

Елемент матрице хеуристике окружења, *h(i,j),* представља оцену растојања између поља окружења чије су координате *i* и *j* и циљног пиксела. Вредност растојања од тренутног до пиксела у окружењу означавамо са *g(i,j)*, где су *i,j*, координате поља у коме се тренутно налази претраживач, узимајући у обзир и препреке. Укупна цена помераја у наредни пиксел рачуна се сабирањем вредности *h(i,j)* , *g(i,j)* и досадашњег пређеног пута (PP). Пиксел у који је потребно прећи је онај са најмањом ценом помераја.

Реално окружење је потребно поделити на једнаке делове, односно пикселе. најчешће су квадратног облика, а могу бити и шестоугаоног или троугаоног. Кретање од једног до другог пиксела може бити четворосмерно или осмосмерно. Четворосмерно кретање значи да је могуће кретање само по хоризонтали и вертикали, док осмосмерно кретање дозвољава и кретања по дијагоналама. У зависности од врсте кретања и хеуристике разликујемо 3 норме:

- Менхетн норма,
- Еуклидска норма,
- Норма „са часа“

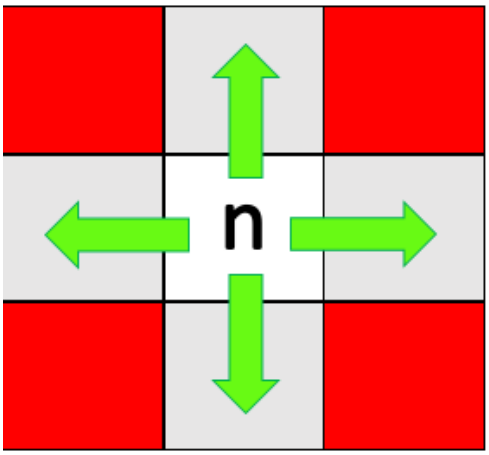
9.1 Менхетн норма

Употребом Менхетн норме користе се четвороспојиви пиксели, тј. мобилни робот може да оствари кретања у хоризонталном и вертикалном правцу, слика 9.10

Израз за Менхетн норму има облик:

$$dM(P, Pcilj) = |Px - Pxcilj| + |Py - Pycilj| \quad (9.6)$$

помоћу којег може да се израчуна хеуристика и пређени пут од тренутног до следећег пиксела. У случају за хеуристику, Px и Py представљају координате наредног пиксела, а $Pxcilj$ и $Pycilj$ координате циљног пиксела. За израчунавање пређеног пута Px и Py представљају координате тренутног пиксела, а $Pxcilj$ и $Pycilj$ координате пиксела у који робот треба да пређе.



Слика 9.10 Могућност кретања робота по Менхетн норми

Рад A* алгоритма по Менхетн норми се показује на произвољном примеру, чије су димензије пиксела 1x1 cm, где је пикселом (2,2) означена почетна позиција (пиксел старта), а пикселом (5,5) циљна позиција робота (пиксел циља). Најпре је потребно одредити хеуристику сваког поља где хеуристика за циљни пиксел има вредност 0, слика 9.11

5	h=4	h=3	h=2	h=1	h=0 циљ
4	h=5	h=4	h=3	h=2	h=1
3	h=6	h=5	h=4	h=3	h=2
2	h=7	h=6 старт	h=5	h=4	h=3
1	h=8	h=7	h=6	h=5	h=4
в/к	1	2	3	4	5

Циљна позиција
Стартна позиција
Препреке

Слика 9.11. Одређивање хеуристике за окружење по Менхетној норми

Након тога потребно је одредити вредност f(I,j) према следећем обрасцу:

$$f(i, j) = h(i, j) + g(i, j) + PP \tag{9.7}$$

где PP представља пређени пут од почетног пиксела до пиксела у коме се робот тренутно налази.

Вредности g(I,j) за овај пример износе:

- g вертикално= 1;
- g хоризонтално= 1;
- g дијагонално= 100;
- g вертикално-препрека= 100;
- g хоризонтално-препрека=100;
- g дијагонално-препрека=100;

Полазећи од пиксела (2.2) и употребом једначине (9.7), израчунавамо вредност f за околна поља, слика 11, с тим да вредност пређеног пута PP у овом тренутку износи 0 пошто је то почетна позиција робота.

f(3,1) = PP + g_dijag + h(3,1) = 0 + 100 + 6 = 106

f(2,1) = PP + g_horiz + h(2,1) = 0 + 1 + 7 = 8

f(1,1) = PP + g_dijag + h(1,1) = 0 + 100 + 8 = 108

f(3,2) = PP + g_vert-prepreka + h(3,2) = 0 + 100 + 5 = 105

f(1,2) = PP + g_vert + h(1,2) = 0 + 1 + 7 = 8

f(3,3) = PP + g_dijag-prepreka + h(3,3) = 0 + 100 + 4 = 104

f(2,3) = PP + g_horiz + h(2,3) = 0 + 1 + 5 = 6

f(1,3) = PP + g_dijag + h(1,3) = 0 + 100 + 6 = 106

Видимо да најмању оцену има пиксел (2,3), и он представља следећи пиксел у који се померамо и за који понављамо претходни поступак.

5	h=4	h=3	h=2	h=1	h=0
4	h=5	h=4	h=3	h=2	h=1
3	h=6 f=106	h=5 f=105	h=4 f=104	h=3	h=2
2	h=7 f=8	h=6 f=6	h=5 f=6	h=4	h=3
1	h=8 f=108	h=7 f=8	h=6 f=106	h=5	h=4
в/к	1	2	3	4	5

Слика 9.12 Одређивање параметра f за околне пикселе

Објашњени поступак се понавља све док не дођемо до циљног пиксела, слике 9.13, 9.14, 9.15, 9.16, 9.17, 9.18 и 9.19.

5	h=4	h=3	h=2	h=1	h=0
4	h=5	h=4	h=3	h=2	h=1
3	h=6 f=106	h=5 f=106	h=4 f=105	h=3 f=104	h=2
2	h=7 f=8	h=6 f=8	h=5 f=6	h=4 f=106	h=3
1	h=8 f=108	h=7 f=108	h=6 f=8	h=5 f=105	h=4
в/к	1	2	3	4	5

Слика 9.13 Корак 2 за одређивање оптималне путање

5	h=4	h=3	h=2	h=1	h=0
4	h=5	h=4	h=3	h=2	h=1
3	h=6 f=106	h=5 f=106	h=4 f=105	h=3 f=104	h=2
2	h=7 f=8	h=6 f=108	h=5 f=8	h=4 f=106	h=3
1	h=8 f=108	h=7 f=10	h=6 f=8	h=5 f=8	h=4
в/к	1	2	3	4	5

Слика 9.14 Корак 3 за одређивање оптималне путање

5	h=4	h=3	h=2	h=1	h=0
4	h=5	h=4	h=3	h=2	h=1
3	h=6 f=106	h=5 f=106	h=4 f=105	h=3 f=104	h=2
2	h=7 f=8	h=6 f=108	h=5 f=108	h=4 f=107	h=3 f=106
1	h=8 f=108	h=7 f=10	h=6 f=10	h=5 f=8	h=4 f=8
в/к	1	2	3	4	5

Слика 9.15 Корак 4 за одређивање оптималне путање

5	h=4	h=3	h=2	h=1	h=0
4	h=5	h=4	h=3	h=2	h=1
3	h=6 f=106	h=5 f=106	h=4 f=105	h=3 f=104	h=2
2	h=7 f=8	h=6 f=108	h=5 f=108	h=4 f=108	h=3 f=8
1	h=8 f=108	h=7 f=10	h=6 f=10	h=5 f=10	h=4 f=8
в/к	1	2	3	4	5

Слика 9.16 Корак 5 за одређивање оптималне путање

5	h=4	h=3	h=2	h=1	h=0
4	h=5	h=4	h=3	h=2	h=1
3	h=6 f=106	h=5 f=106	h=4 f=105	h=3 f=108	h=2 f=8
2	h=7 f=8	h=6 f=108	h=5 f=108	h=4 f=109	h=3 f=8
1	h=8 f=108	h=7 f=10	h=6 f=10	h=5 f=110	h=4 f=10
в/к	1	2	3	4	5

Слика 9.17 Корак 6 за одређивање оптималне путање

5	h=4	h=3	h=2	h=1	h=0
4	h=5	h=4	h=3	h=2 f=108	h=1 f=8
3	h=6 f=106	h=5 f=106	h=4 f=105	h=3 f=10	h=2 f=8
2	h=7 f=8	h=6 f=108	h=5 f=108	h=4 f=110	h=3 f=10
1	h=8 f=108	h=7 f=10	h=6 f=10	h=5 f=110	h=4 f=10
в/к	1	2	3	4	5

Слика 9.18 Корак 7 за одређивање оптималне путање

5	h=4	h=3	h=2	h=1 f=108	h=0 f=8
4	h=5	h=4	h=3	h=2 f=10	h=1 f=8
3	h=6 f=106	h=5 f=106	h=4 f=105	h=3 f=110	h=2 f=10
2	h=7 f=8	h=6 f=108	h=5 f=108	h=4 f=110	h=3 f=10
1	h=8 f=108	h=7 f=10	h=6 f=10	h=5 f=110	h=4 f=10
в/к	1	2	3	4	5

Слика 9.19. Корак 8 за одређивање оптималне путање

Оптимална путања робота коју робот треба да пређе приказана је на слици 9.20, и она износи:

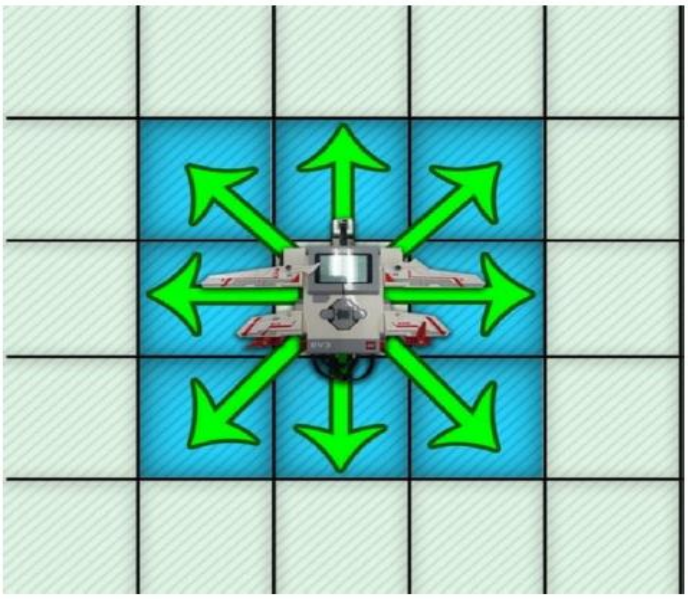
$$s = 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 = 8\,cm \tag{9.8}$$

5	h=4	h=3	h=2	h=1 f=108	h=0 f=
4	h=5	h=4	h=3	h=2 f=10	h=1 f=
3	h=6 f=106	h=5 f=106	h=4 f=105	h=3 f=110	h=2 f=10
2	h=7 f=8	h=6 f=108	h=5 f=108	h=4 f=110	h=3 f=10
1	h=8 f=108	h=7 f=10	h=6 f=10	h=5 f=110	h=4 f=10
в/к	1	2	3	4	5

Слика 9.20 Приказ оптималне путање моблиног робота

9.2 Еуклидска норма

Код еуклидске норме кретање је могуће остварити у четири правца, што значи да се кретање може извршити у један од осам суседних пиксела.



Слика 9.21 Могућност кретања робота по Еуклидској норми

Израз за Еуклидску норму има облик:

$$d_{\mathrm{E}}(\mathrm{n},\ \mathrm{n}_{\text{циљ}}) = \sqrt{(nx - nx_{\text{циљ}})^2 + (ny - ny_{\text{циљ}})^2} \tag{9.9}$$

који се као и код Менхетн норме користи за израчунавање Хеуристике и пређеног пута.

Након тога потребно је одредити вредност f(I,j) према обрасцу (9.7)

Вредности g(I,j) за Менхетн норму износе:

- $g_{\text{вертикално}} = 1$
- $g_{\text{хоризонтално}} = 1$
- $g_{\text{дијагонално}} = 1.41$
- $g_{\text{х-препрека}} = 100$
- $g_{\text{в-препрека}} = 100$
- $g_{\text{д-препрека}} = 100.41$
- $h_{\text{хоризонтално}} = 1$
- $h_{\text{вертикално}} = 1$
- $h_{\text{вдијагонално}} = 1{,}41$

Пример:

6	h=1,41	h=1	h=1,41	h=2,24	h=3,16	h=4,12
5	h=1	h=0 циљ	h=1	h=2	h=3	h=4
4	h=1,41	h=1	h=1,41	h=2,24	h=3,16	h=4,12
3	h=2,24	h=2	h=2,24	h=2,83	h=3,61	h=4,48
2	h=3,16 почетак	h=3	h=3,16	h=3,61	h=4,24	h=5
1	h=4,12	h=4	h=4,12	h=4,47	h=5	h=5,67
В/К	**1**	**2**	**3**	**4**	**5**	**6**

Слика 9.22. Мапа окружења

Корак 1. (слика 9.23)

PP=0

f(3,1) = h(3,1) + g(3,1) + PP = 2,24 + 1 + 0 = 3,24

f(3,2) = h(3,2) + g(3,2) + PP= 2 + 100,41 +0= 102,41

f(2,2) = h(2,2) + g(2,2) + PP= 3 + 1 +0 = 4

f(1,1) = h(1,1) + g(1,1) + PP= 4 ,1+ 1 +0 = 5,1

f(1,2) = h(1,2) + g(1,2) + PP= 4 + 1,41 + 0 = 5,41

6	h=1,41	h=1	h=1,41	h=2,24	h=3,16	h=4,12
5	h=1	h=0 циљ	h=1	h=2	h=3	h=4
4	h=1,41	h=1	h=1,41	h=2,24	h=3,16	h=4,12
3	h=2,24 f=3,24	h=2 f=102,41	h=2,24	h=2,83	h=3,61	h=4,48
2	h=3,16 старт	h=3 f=4	h=3,16	h=3,61	h=4,24	h=5
1	h=4,12 f=5,12	h=4 f=5,41	h=4,12	h=4,47	h=5	h=5,67
В/К	**1**	**2**	**3**	**4**	**5**	**6**

Слика 9.23 Корак 1

Корак 2. (слика 9.24)

PP=0+1

f(4,1) = h(4,1) + g(4,1) + PP = 1,41 + 100 + 1= 102,41

f(4,2) = h(3,2) + g(3,2) + PP= 1 + 100,41 +1= 102,41

f(3,2) = h(1,1) + g(1,1) + PP= 2+100 +1 = 103

f(2,1) = h(1,2) + g(1,2) + PP= 3,16 + 1+ 1 = 5,16

f(2,2) = h(1,2) + g(1,2) + PP= 3 + 1,41 + 1 = 5,41

6	h=1,41	h=1	h=1,41	h=2,24	h=3,16	h=4,12
5	h=1	h=0 циљ	h=1	h=2	h=3	h=4
4	h=1,41 f=102,41	h=1 f=102,41	h=1,41	h=2,24	h=3,16 f	h=4,12
3	h=2,24 f=3,24	h=2 f=103	h=2,24	h=2,83	h=3,61	h=4,48
2	h=3,16 f=5,16	h=3 f=5,41	h=3,16	h=3,61	h=4,24	h=5
1	h=4,12 f=5,12	h=4 f=5,41	h=4,12	h=4,47	h=5	h=5,67
В/К	**1**	**2**	**3**	**4**	**5**	**6**

Слика 9.24 Корак 2

Корак 3. (слика 9.25)

PP=0+1+1,41=2,41

f(3,1) = h(3,1) + g(3,1) + PP = 2,24 + 1,41 + 2,41= 6,06

f(3,2) = h(3,2) + g(3,2) + PP = 2 + 100 +2,41= 104,41

f(3,3) = h(3,3) + g(3,3) + PP = 2,24 + 1,41 +2,41 = 6,06

f(2,1) = h(1,1) + g(1,1) + PP = 3 ,16+ 1 +2,41= 6,57

f(2,3) = h(1,2) + g(1,2) + PP= 3,16 + 1+ 2,41 = 6,57

f(1,1) = h(1,1) + g(1,1) + PP= 4 ,12+ 1,41 +2,41 = 7,94

f(1,2) = h(1,2) + g(1,2) + PP= 4 + 1 + 2,41 = 7,41

f(1,3) = h(1,2) + g(1,2) + PP= 4,12 + 1,41 + 2,41 = 7,94

6	h=1,41	h=1	h=1,41	h=2,24	h=3,16	h=4,12
5	h=1	h=0 циљ	h=1	h=2	h=3	h=4
4	h=1,41 f=102,41	h=1 f=102,41	h=1,41	h=2,24	h=3,16 f	h=4,12
3	h=2,24 f=6,06	h=2 f=104,41	h=2,24 f=6,06	h=2,83	h=3,61	h=4,48
2	h=3,16 f=6,57	h=3 f=5,41	h=3,16 f=6,57	h=3,61	h=4,24	h=5
1	h=4,12 f=7,94	h=4 f=7,41	h=4,12 f=7,94	h=4,47	h=5	h=5,67
В/К	**1**	**2**	**3**	**4**	**5**	**6**

Слика 9.25 Корак 3

Корак 4. (слика 9.26)

PP=0+1+1,41+1,41=3,82

f(4,2) = h(4,2) + g(4,2) + PP = 1 + 100,41 + 3,82= 105,23

f(4,3) = h(4,3) + g(4,3) + PP = 1,41 + 100 +3,82= 105,23

f(4,4) = h(4,4) + g(4,4) + PP = 2,24 + 1,41 +3,82 = 7,47

f(3,2) = h(2,1) + g(2,1) + PP = 2+ 100 +3,82=105,98

f(3,4) = h(2,3) + g(2,3) + PP= 2,83 + 1+ 3,82 = 7,65

f(2,2) = h(1,1) + g(1,1) + PP= 3+ 1,41 +3,82 = 8,23

f(2,3) = h(1,2) + g(1,2) + PP= 3,16 + 1 + 3,82 = 7,98

f(2,4) = h(1,2) + g(1,2) + PP= 3,61 + 1,41 + 3,82 = 8,84

6	h=1,41	h=1	h=1,41	h=2,24	h=3,16	h=4,12
5	h=1	h=0 циљ	h=1	h=2	h=3	h=4
4	h=1,41 f=102,41	h=1 f=105,23	h=1,41 f=105,23	h=2,24 f=7,47	h=3,16 f	h=4,12
3	h=2,24 f=6,06	h=2 f=105,98	h=2,24 f=6,06	h=2,83 f=7,65	h=3,61	h=4,48
2	h=3,16 f=6,57	h=3 f=8,23	h=3,16 f=7,98	h=3,61 f=8,84	h=4,24	h=5
1	h=4,12 f=7,94	h=4 f=7,41	h=4,12 f=7,94	h=4,47	h=5	h=5,67
В/К	**1**	**2**	**3**	**4**	**5**	**6**

Слика 9.26 Корак 4

Корак 5. (слика 9.27)

PP=0+1+1,41+1,41+1,41=5,23

f(5,3) = h(5,3) + g(5,3) + PP = 1 + 1,41 + 5,23= 7,64

f(5,4) = h(5,4) + g(5,4) + PP = 2 + 1 +5,23= 8,23

f(5,5) = h(5,5) + g(5,5) + PP = 3+ 1,41 +5,23 = 9,64

f(4,3) = h(4,3) + g(4,3) + PP = 1,41+ 100 +5,23=106,64

f(4,5) = h(4,5) + g(4,5) + PP= 3,16 + 1+ 5,23 = 9,39

f(3,3) = h(3,3) + g(3,3) + PP= 2,24+ 100,41 +5,23 = 107,88

f(3,4) = h(3,4) + g(3,4) + PP= 2,83+ 100 + 5,23 =108,06

f(3,5) = h(1,2) + g(1,2) + PP= 3,61+ 1,41 + 3,82 = 8,84

6	h=1,41	h=1	h=1,41	h=2,24	h=3,16	h=4,12
5	h=1	h=0 циљ	h=1 f=7,64	h=2 f=8,23	h=3 f=9,64	h=4
4	h=1,41 f=102,41	h=1 f=105,23	h=1,41 f=106,64	h=2,24 f=7,47	h=3,16 f=9,39	h=4,12
3	h=2,24 f=6,06	h=2 f=105,98	h=2,24 f=107,88	h=2,83 f=108,06	h=3,61 f=8,84	h=4,48
2	h=3,16 f=6,57	h=3 f=8,23	h=3,16 f=7,98	h=3,61 f=8,84	h=4,24	h=5
1	h=4,12 f=7,94	h=4 f=7,41	h=4,12 f=7,94	h=4,47	h=5	h=5,67
В/К	**1**	**2**	**3**	**4**	**5**	**6**

Слика 9.27 Корак 5

Корак 6. (слика 9.28)

PP=0+1+1,41+1,41+1,41+1,41=6,64

f(6,2) = h(6,2) + g(6,2) + PP = 1 + 1,41 + 6,64= 9,05

f(6,3) = h(5,4) + g(5,4) + PP = 1,41 + 1 +6,64= 9,05

f(6,4) = h(5,5) + g(5,5) + PP = 2,24+ 1,41 +6,64 = 10,08

f(4,3) = h(4,3) + g(4,3) + PP = 0+ 1 +6,64=7,64

f(4,5) = h(4,5) + g(4,5) + PP= 2+ 1+ 6,64 = 9,64

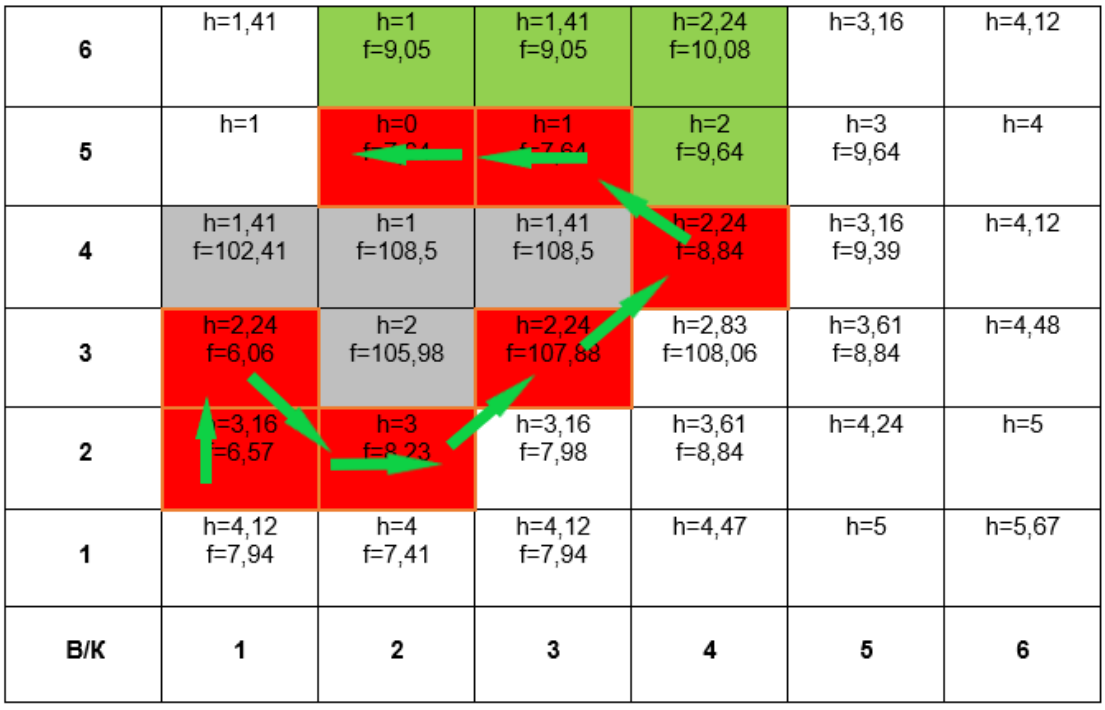
f(3,3) = h(3,3) + g(3,3) + PP= 1+ 100,41 +6,64= 108,05

f(3,4) = h(3,4) + g(3,4) + PP= 1,41+ 100 + 6,64= 108,05

f(3,5) = h(1,2) + g(1,2) + PP= 2,24+ 1,41 + 6,64 = 8,84

6	h=1,41	h=1 f=9,05	h=1,41 f=9,05	h=2,24 f=10,08	h=3,16	h=4,12
5	h=1	h=0 f=7,64	h=1 f=7,64	h=2 f=9,64	h=3 f=9,64	h=4
4	h=1,41 f=102,41	h=1 f=108,5	h=1,41 f=108,5	h=2,24 f=8,84	h=3,16 f=9,39	h=4,12
3	h=2,24 f=6,06	h=2 f=105,98	h=2,24 f=107,88	h=2,83 f=108,06	h=3,61 f=8,84	h=4,48
2	h=3,16 f=6,57	h=3 f=8,23	h=3,16 f=7,98	h=3,61 f=8,84	h=4,24	h=5
1	h=4,12 f=7,94	h=4 f=7,41	h=4,12 f=7,94	h=4,47	h=5	h=5,67
B/K	**1**	**2**	**3**	**4**	**5**	**6**

Слика 9.28 Корак 6



Слика 9.29 Путања којом се робот кретао

9.3 Норма „са часа“

- $g_{вертикално} = 1$
- $g_{хоризонтално} = 1$
- $g_{дијагонално} = 1.4$
- $g_{х\text{-}препрека} = 100$
- $g_{в\text{-}препрека} = 100$
- $g_{д\text{-}препрека} = 100.4$
- $h_{хоризонтално} = 1$
- $h_{вертикално} = 1$
- $h_{вдијагонално} = 1,4$

6	h=1,4	h=1	h=1,4	h=2,4	h=3,4	h=4,4
5	h=1	h=0 циљ	h=1	h=2	h=3	h=4
4	h=1,4	h=1	h=1,4	h=2,4	h=3,4	h=4,4
3	h=2,4	h=2	h=2,4	h=2,8	h=3,8	h=4,8
2	h=3,4 почетак	h=3	h=3,4	h=3,8	h=4,2	h=5,2
1	h=4,4	h=4	h=4,4	h=4,8	h=5,2	h=5,6
В/К	**1**	**2**	**3**	**4**	**5**	**6**

Слика 9.30 Мапа окружења

Корак 1. (слика 9.31)

PP=0

f(3,1) = h(3,1) + g(3,1) + PP = 2,4 + 1 + 0 = 3,4

f(3,2) = h(3,2) + g(3,2) + PP= 2 + 100,4 +0= 102,4

f(2,2) = h(2,2) + g(2,2) + PP= 3 + 1 +0 = 4

f(1,1) = h(1,1) + g(1,1) + PP= 4,4+ 1 +0 = 5,4

f(1,2) = h(1,2) + g(1,2) + PP= 4 + 1,4 + 0 = 5,4

6	h=1,4	h=1	h=1,4	h=2,4	h=3,4	h=4,4
5	h=1	h=0 циљ	h=1	h=2	h=3	h=4
4	h=1,4	h=1	h=1,4	h=2,4	h=3,4	h=4,4
3	h=2,4 f=3,4	h=2 f=102,4	h=2,4	h=2,8	h=3,8	h=4,8
2	h=3,4 почетак	h=3 f=4	h=3,4	h=3,8	h=4,2	h=5,2
1	h=4,4 f=5,4	h=4 f=5,4	h=4,4	h=4,8	h=5,2	h=5,6
В/К	**1**	**2**	**3**	**4**	**5**	**6**

Слика 9.31 Корак 1

Корак 2. (слика 9.32)

PP=0+1=1

f(4,1) = h(4,1) + g(4,1) + PP = 1,4 + 100 + 1= 102,4

f(4,2) = h(3,2) + g(3,2) + PP= 1 + 100,4 +1= 102,4

f(3,2) = h(1,1) + g(1,1) + PP= 2+100 +1 = 103

f(2,1) = h(1,2) + g(1,2) + PP= 3,4 + 1+ 1 = 5,4

f(2,2) = h(1,2) + g(1,2) + PP= 3 + 1,4 + 1 = 5,4

6	h=1,4	h=1	h=1,4	h=2,4	h=3,4	h=4,4
5	h=1	h=0 циљ	h=1	h=2	h=3	h=4
4	h=1,4 f=102,4	h=1 f=102,4	h=1,4	h=2,4	h=3,4	h=4,4
3	h=2,4 f=3,4	h=2 f=103	h=2,4	h=2,8	h=3,8	h=4,8
2	h=3,4 f=5,4	h=3 f=5,4	h=3,4	h=3,8	h=4,2	h=5,2
1	h=4,4 f=5,4	h=4 f=5,4	h=4,4	h=4,8	h=5,2	h=5,6
В/К	**1**	**2**	**3**	**4**	**5**	**6**

Слика 9.32 Корак 2

Корак 3. (слика 9.33)

PP=1+1,4=2,4

f(3,1) = h(3,1) + g(3,1) + PP = 2,4 + 1,4 + 2,4= 6,2

f(3,2) = h(3,2) + g(3,2) + PP = 2 + 100 +2,4= 104,4

f(3,3) = h(3,3) + g(3,3) + PP = 2,4 + 1,4 +2,4 = 6,2

f(2,1) = h(1,1) + g(1,1) + PP = 3 ,4+ 1 +2,4= 6,8

f(2,3) = h(1,2) + g(1,2) + PP= 3,4 + 1+ 2,4 = 6,8

f(1,1) = h(1,1) + g(1,1) + PP= 4 ,4 + 1,4 +2,4 = 8,2

f(1,2) = h(1,2) + g(1,2) + PP= 4 + 1 + 2,4 = 7,4

f(1,3) = h(1,2) + g(1,2) + PP= 4,4 + 1,4 + 2,4 = 8,2

6	h=1,4	h=1	h=1,4	h=2,4	h=3,4	h=4,4
5	h=1	h=0 циљ	h=1	h=2	h=3	h=4
4	h=1,4 f=102,4	h=1 f=102,4	h=1,4	h=2,4	h=3,4	h=4,4
3	h=2,4 f=6,2	h=2 f=104,4	h=2,4 f=6,2	h=2,8	h=3,8	h=4,8
2	h=3,4 f=6,8	h=3 f=5,4	h=3,4 f=6,8	h=3,8	h=4,2	h=5,2
1	h=4,4 f=8,2	h=4 f=7,4	h=4,4 f=8,2	h=4,8	h=5,2	h=5,6
В/К	**1**	**2**	**3**	**4**	**5**	**6**

Слика 9.33 Корак 3

Корак 4. (слика 9.34)

PP=1+1,4 + 1,4=3,8

f(4,2) = h(4,2) + g(4,2) + PP = 1 + 100,4 + 3,8= 105,2

f(4,3) = h(4,3) + g(4,3) + PP = 1,4 + 100 +3,8= 105,2

f(4,4) = h(4,4) + g(4,4) + PP = 2,4 + 1,4 +3,8 = 7,6

f(3,2) = h(2,1) + g(2,1) + PP = 2+ 100 +3,8 = 105,8

f(3,4) = h(2,3) + g(2,3) + PP= 2,8 + 1+ 3,8 = 8,6

f(2,2) = h(1,1) + g(1,1) + PP= 3+ 1,4 +3,8 = 8,2

f(2,3) = h(1,2) + g(1,2) + PP= 3,4 + 1 + 3,8 = 8,2

f(2,4) = h(1,2) + g(1,2) + PP= 3,8 + 1,4 + 3,8 = 9

f(2,4) = h(1,2) + g(1,2) + PP= 3,61 + 1,4 + 3,82 = 8,84

6	h=1,4	h=1	h=1,4	h=2,4	h=3,4	h=4,4
5	h=1	h=0 циљ	h=1	h=2	h=3	h=4
4	h=1,4 f=102,4	h=1 f=105,2	h=1,4 f=105,2	h=2,4 f=7,6	h=3,4	h=4,4
3	h=2,4 f=6,2	h=2 f=105,8	h=2,4 f=6,2	h=2,8 f=8,6	h=3,8	h=4,8
2	h=3,4 f=6,8	h=3 f=8,2	h=3,4 f=8,2	h=3,8 f=9	h=4,2	h=5,2
1	h=4,4 f=8,2	h=4 f=7,4	h=4,4 f=8,2	h=4,8	h=5,2	h=5,6
B/K	**1**	**2**	**3**	**4**	**5**	**6**

Слика 9.34 Корак 4

Корак 5. (слика 9.35)

PP=1+1,4+1,4+1,4=5,2

f(5,3) = h(5,3) + g(5,3) + PP = 1 + 1,4 + 5,2= 7,6

f(5,4) = h(5,4) + g(5,4) + PP = 2 + 1 +5,2= 8,2

f(5,5) = h(5,5) + g(5,5) + PP = 3+ 1,4 +5,2 = 9,6

f(4,3) = h(4,3) + g(4,3) + PP = 1,4+ 100 +5,2=106,6

f(4,5) = h(4,5) + g(4,5) + PP= 4,4 + 1,4+ 5,2 = 11

f(3,3) = h(3,3) + g(3,3) + PP= 2,4+ 1+5,2 = 8,6

f(3,4) = h(3,4) + g(3,4) + PP= 2,8+ 1,4 + 5,2= 9,4

f(3,5) = h(1,2) + g(1,2) + PP= 3,8+ 1,4 + 5,2= 10,4

6	h=1,4	h=1	h=1,4	h=2,4	h=3,4	h=4,4
5	h=1	h=0 циљ	h=1 f=7,6	h=2 f=8,2	h=3 f=9,6	h=4
4	h=1,4 f=102,4	h=1 f=105,2	h=1,4 f=106,6	h=2,4 f=7,6	h=3,4 f=11	h=4,4
3	h=2,4 f=6,2	h=2 f=105,8	h=2,4 f=8,6	h=2,8 f=9,4	h=3,8 f=10,4	h=4,8
2	h=3,4 f=6,8	h=3 f=8,2	h=3,4 f=8,2	h=3,8 f=9	h=4,2	h=5,2
1	h=4,4 f=8,2	h=4 f=7,4	h=4,4 f=8,2	h=4,8	h=5,2	h=5,6
B/K	**1**	**2**	**3**	**4**	**5**	**6**

Слика 9.35 Корак 5

6	h=1,4	h=1 f=9	h=1,4 f=9	h=2,4 f=10,4	h=3,4	h=4,4
5	h=1	h=0 f=7,6	h=1 f=7,6	h=2 f=9,6	h=3 f=9,6	h=4
4	h=1,4 f=102,4	h=1 f=108	h=1,4 f=108	h=2,4 f=10,4	h=3,4 f=11	h=4,4
3	h=2,4 f=6,2	h=2 f=105,8	h=2,4 f=8,6	h=2,8 f=9,4	h=3,8 f=10,4	h=4,8
2	h=3,4 f=6,8	h=3 f=8,2	h=3,4 f=8,2	h=3,8 f=9	h=4,2	h=5,2
1	h=4,4 f=8,2	h=4 f=7,4	h=4,4 f=8,2	h=4,8	h=5,2	h=5,6
B/K	**1**	**2**	**3**	**4**	**5**	**6**

Слика 9.36 Путања којом се кретао робот

9.4 Еуклидска норма кодови

У наставку је описана функција која за задате *x* и *y* координате старта и циља помоћу А звезда алгоритма са еуклидском нормом проналази најкраћи пут кретања робота.

Димензије радног окружења у коме се робот кретао су 150 cm x 100 cm и оно је подељено на пикселе величине 5 cm x 5 cm. Број пиксела дуж *x* и *y* осе дефинисан је у линијама кода 3-5, а окружење је проширено за 2 пиксела чиме су урачунате ивице.

По А звезда алгоритму, за физичке препреке у окружењу потребно је повећати вредност елемента матрице препреке *f* којим је описана позиција те препреке, што је извршено у линијама кода 8-39.

Матрица хеуристике рачуна се за сваки пиксел и представља удаљеност конкретног пиксела од пиксела циља што је у овом случају рачунато преко еуклидског растојања.

Док се стартни и циљни пиксели не поклопе, у *while* петљи одређују се наредне позиције (пиксели) тако да је пређени пут робота минималан. За тренутну позицију робота, одређену координатама *x_start* и *y_start*, додељује се цена помераја околним пикселима на основу одговарајућих елемената матрице хеуристике, матрице препрека, типа кретања (g) и дотадашњег пређеног пута. За пиксел чија је цена помераја минимална, чувају се координате *x* и *y* у вектору *prev_pos*. Тај вектор представља излаз из функције након извршавања алгоритма и садржи информације о координатама свих позиција које је робот заузео.

Да би се избегло да се робот у идућој итерацији врати у претходни положај, додаје се „казна“ тј. мења се вредност тог поља матрице хеуристике за 20 (линија 60), а да робот у тренутној итерацији не би „одлучио“ да остане у истој позицији уколико је та вредност минимална, додељује се „тренутна казна“ у виду повећања цене за 50.

Када се одреди наредни пиксел, пређени пут се пре завршетка тренутне итерације *while* петље повећава за пређени пут до наредног пиксела.

```
1  function [prev_pos] = a_zvezda_euklid(x_start,y_start,x_cilj,y_cilj)
 2
 3 M=30; %broj piksela po y
 4 N=20; %broj piksela po x
 5 m=M+2;
 6 n=N+2;
 7 %matrica prepreka f
 8 f=zeros(m,n);
 9 %formiranje zida
10 f(1,:)=100;
11 f(:,1)=100;
12 f(m,:)=100;
```

```
13 f(:,n)=100;
14
15 % definisanje prepreka
16 % y x
17 f(2:20,17:21) = 100;
18 f(18:31,2:3) = 100;
19 f(24:31,5:21) = 100;
20
21 f(8:13,7:12) = 100;
22 f(8,7)=0;
23 f(9,7)=0;
24 f(8,8)=0;
25 f(8,11)=0;
26 f(8,12)=0;
27 f(9,12)=0;
28 f(12,7)=0;
29 f(13,7)=0;
30 f(13,8)=0;
31 f(13,11)=0;
32 f(13,12)=0;
33 f(12,12)=0;
34
35 f(1:5,2) = 100;
36
37 f(2,1:5) = 100;
38 f(3,3) = 100;
39
40 % matrica heuristike h(cena putanje za svako polje do cilja)
41 for i=1:m
42     for j=1:n
43
44     h(i,j)=sqrt((j-x_cilj)^2+(i-y_cilj)^2);
45
46     end
47 end
```

```
48
49
50 p=0;% predjeni put
51 g=zeros(m,n);%matrica g
52 prev_pos=[y_start;x_start];% vektor prethodnih pozicija
53
54 %kretanje robota ka cilju
55
56  while (y_start~=y_cilj || x_start~=x_cilj)
57   RP=f;
58    minimum=1000;
59    %cena vracanja u prethodnu poziciju
60    h(y_start,x_start)=h(y_start,x_start)+20;
61
62
63     for  x=x_start-1:x_start+1
64         for y=y_start-1:y_start+1
65
66             if x_start~=x && y_start~=y
67                 g(y,x)=1.41;
68             elseif x_start==x && y_start==y
69                 g(y,x)=50;
70             else
71                 g(y,x)=1;
72             end
73
74             RP(y,x)=RP(y,x)+h(y,x)+g(y,x)+p;
75
76             if RP(y,x)< minimum
77                 minimum=RP(y,x);
78                 a=y;
79                 b=x;
80             end
81         end
82     end
```

```
83
84
85      %kontrola trenutne koordinate
86      prev_pos=[prev_pos,[ a;b]];
87
88      %pocetna pozicija za sledece kretanje
89      y_start=a;
90      x_start=b;
91
92      %predjeni put
93      p=p+g(a,b);
94
95  end
96 prev_pos = flip(prev_pos) - 1;
97 prev_pos = prev_pos';
98 end
```

9.5 Менхетн норма кодови

```
function [put] = a_star_menh(i_start, j_start, i_end, j_end, mapa)
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here

[n,m] = size(mapa);

%heuristika
h = zeros(n,m);

for i = 1:n
    for j = 1:m
        h(i,j) = abs(i_end-i) + abs(j_end-j);
    end
end

matrica_putanje = zeros(n,m);
```

Улазни аргументи функције су почетак и крај оптималне путање која се тражи, као и мапу која описује наше окружење.

Креирају се две матрице, једна ће чувати вредности хеуристике у чворовима, а друга ће нумерисати редослед којим ће се чворови посећивати у потрази за путањом.

```
g = mapa;
otvoreni = [];
zatvoreni = [i_start, j_start];

i_cur = i_start; j_cur = j_start;

tekuci = 1;
matrica_putanje(i_start,j_start) = 1;

i_cur = i_start; j_cur=j_start;
```

Креирамо матрицу цена (g). Креира се и отворени и затворени скуп чворова.

```
while(true)
    tekuci = tekuci+1;
    tacke = [i_cur-1 , j_cur; i_cur+1 , j_cur; i_cur , j_cur-1; i_cur ,
j_cur+1];

    for i = 1:size(tacke,1)
        xi = tacke(i,1); yi = tacke(i,2);
        if(korektna_pozicija(xi,yi,n,m) && ...
           (size(otvoreni,1)== 0 || ~vec_u_otvorenim(xi,yi,otvoreni)) && ...
           (size(zatvoreni,1)== 0 || ~vec_u_otvorenim(xi,yi,zatvoreni)))
            if(g(xi,yi)<2000)
                otvoreni = [otvoreni; [xi,yi]];
                g(xi,yi) = g(i_cur,j_cur)+1;
            end
        end
    end

    if(size(otvoreni)==0)
        put = [];
        return;
    end
```

```
    min_xi= -1; min_yi=-1;
    min_g = 999999;
    %disp(otvoreni);
    for i = 1:size(otvoreni,1)
        xi = otvoreni(i,1); yi = otvoreni(i,2);
        %disp([xi,yi]);
        %disp(g(xi,yi) + h(xi,yi));
        if( (g(xi,yi) + h(xi,yi)) < min_g)
            min_xi = xi;
            min_yi = yi;
            min_g = g(xi,yi)+ h(xi,yi);
            min_i= i;
        end
    end
    %disp(otvoreni);
    matrica_putanje(min_xi,min_yi)=tekuci;
    i_cur = min_xi; j_cur=min_yi;
    otvoreni(min_i,:) = [];
    zatvoreni = [zatvoreni; [min_xi, min_yi]];
    if( min_xi == i_end && min_yi == j_end)
        break;
    end
    %disp("uzmimamo ");
    %disp([min_xi, min_yi]);
end
```

У оквиру петље се проналази следећа позиција у потрази за оптималном путањом. Увећава се „*tekuci*“ ради поменуте нумерације. Испитују се суседни чворови, који се убацују у отворену листу у случају да нису примећивани раније. Такође им се додељује вредност цене пута. Менхетн хеуристика има особину конзистентности, тако да за једном затворене чворове знамо да је пронађен најкраћи пут до њих, те не морамо то проверавати. Хеуристика је конзистентна, ако за свака два суседна чвора *m* и *n* важи да је збир цене пута између *n* и *m* и вредности хеуристике у чвору *m* већи од вредности хеуристике у чвору *n*, тј. *g(n,m) + h(m) >= h(n)*. Због ове особине, знамо да ће родитељ чвора бити онај сусед који је раније посећен.

Проверава се да ли је скуп отворених чворова празан, тј. да ли можда пут не постоји. Од свих чворова у отвореном скупу, бирамо онај са најбољом вредношћу функције евалуације. Он постаје тренутни чвор, а претходни избацујемо из скупа отворених и убацујемо га у скуп затворених чворова. Ово се понавља док не стигнемо до циљног чвора.

```
xc=i_end; yc=j_end;

put = [xc, yc];

while(true)
    if(xc==i_start && yc==j_start)
        break
    end
    [xc,yc] = vrati_roditelja(xc,yc,n,m,matrica_putanje);
    put = [put; [xc,yc]];
end

disp(matrica_putanje);
disp(g);
disp(h);
end
```

На крају конструишемо путању крећући се уназад од циљног чвора, преко родитеља, до почетног чвора.

Подфункције

```
function [da_li_je] = korektna_pozicija(i,j,n,m)
%UNTITLED4 Summary of this function goes here
%   Detailed explanation goes here
if i<=n && i>0 && j<=m && j>0
    da_li_je = true;
else
    da_li_je = false;
```

```
end
end
```

```
function [je_li] = vec_u_otvorenim(x,y,otvoreni)
%UNTITLED5 Summary of this function goes here
%   Detailed explanation goes here

if(any(ismember(otvoreni,[x,y],'rows')))
    je_li = true;
else
    je_li = false;
end

end
```

```
function [xr,yr] = vrati_roditelja(x,y,n,m,matr)
%UNTITLED6 Summary of this function goes here
%   Detailed explanation goes here
tacke = [x-1 , y; x+1 , y; x, y-1; x, y+1];

min_v = 10001;

for i = 1:size(tacke)
    xi = tacke(i,1); yi = tacke(i,2);
    if(korektna_pozicija(xi,yi,n,m))
        if(min_v > matr(xi,yi) && matr(xi,yi)>0)
            xr = xi;
            yr= yi;
            min_v = matr(xi,yi);
        end
    end
end
end
```

10. Опис кодова за управљање роботом у MATLAB-у

10.1 Описивање функције *Rotate*

```
function [ r1,r2] = Rotate(PP, mymotor2, mymotor1 )

B=95; %Rastojanje izmedju tockova
D=43;% PRECNIK TOCKA U MM

if PP>0 % UKOLIKO SE KRECE U NAPRED
    a=1;
else   % UKOLIKO SE KRECE U NAZAD
    a=-1;
end
MaxSPEED=30*a;
SPEED =MaxSPEED;% Motor speed

Fi=round(PP*B/D); % Odredjivanje ugla vratila koji za koji treba motor da
rotira

resetRotation(mymotor1);                    % Reset motor rotation counter
resetRotation(mymotor2);

mymotor1.Speed = -SPEED*a;                    % Set motor speed
mymotor2.Speed = SPEED*a;

kP=1;
```

Као улазне променљиве, функција *Rotate* узима жељени угао ротације робота *PP* и функције које покрећу моторе левог и десног точка, *mymotor1* и *mymotor2*, респективно. У зависности од тога да ли је задати угао ротације позитиван или негативан, моторима точкова се прослеђује различита команда. Зависност од знака угла ротације смешта се у варијаблу *a*.

```
start(mymotor1);                            % Start motor
start(mymotor2);
```

На доле приказаном алгоритму може се видети како се моторима точкова додељује брзина у зависности од угла жељене ротације. Да би се робот заротирао за 90 степени, потребно је да се леви точак ротира у математички негативном смеру, а десни точак у математички позитивном смеру и обрнуто за угао ротације робота који износи -90 степени.

```
if a==1
    while 1
        r1 = readRotation(mymotor1);            % Read rotation counter in
degrees
        r2 = readRotation(mymotor2);

        r=(-r1*a+r2*a)/2;

        error=Fi-r;
        SPEED=kP*error;%+kD*error_dot+kI*E

        if SPEED>=MaxSPEED
            SPEED=MaxSPEED;
        elseif SPEED <=5
            break;
        end
%         SPEED
        mymotor1.Speed = -SPEED*a;                  % Set motor speed
        mymotor2.Speed = SPEED*a;
    end
else
        while 1
        r1 = readRotation(mymotor1);            % Read rotation counter in
degrees
        r2 = readRotation(mymotor2);

        r=(+r1*a-r2*a)/2;

        error=Fi-r;
        SPEED=kP*error;%+kD*error_dot+kI*E

        if SPEED<=MaxSPEED
            SPEED=MaxSPEED;
        elseif SPEED >=-5
            break;
```

```
        end
%         SPEED
        mymotor1.Speed = SPEED*a;                    % Set motor speed
        mymotor2.Speed = -SPEED*a;
    end
end
```

У наредном алгоритму приказано је на који начин се управља брзином мотора за два смера ротације.

Након што су мотори укључени помоћу команде *startmymotor*, моторима се додељује максимална брзина и они почињу да се крећу. У *if/else* тврдњама, на основу очитаних углова ротације точкова, r_1 и r_2 одређује се нова брзина мотора. Прво се рачуна средња вредност угла ротације, *r*, због тога што мотори не раде исто. На основу разлике између задатог (Fi) и оствареног угла ротације (r) точка робота, рачуна се брзина, *SPEED*, која се упоређује са максимално могућом брзином, задатом пре петље, *MaxSPEED*. Уколико је брзина мања од максималне, потребно је убрзати мотор и због тога се брзина мотора изједначава са максималном. Ако је брзина већа од -5 или мања од 5 зависно од смера ротације, разлика између траженог и оствареног угла ротације је 5 степени што је задовољавајућа тачност и петља се зауставља, а самим тим и робот.

```
while 1
        r1 = readRotation(mymotor1);            % Read rotation counter in
degrees
        r2 = readRotation(mymotor2);

        r=(+r1*a-r2*a)/2;

        error=Fi-r;

        SPEED=kP*error;%+kD*error_dot+kI*E

        if SPEED<=MaxSPEED
            SPEED=MaxSPEED;
        elseif SPEED >=-5
            break;
        end
%         SPEED
        mymotor1.Speed = SPEED*a;                    % Set motor speed
        mymotor2.Speed = -SPEED*a;
```

```
    end
end
r1=double(r1);
r2=double(r2);
%KRAJ
stop(mymotor1,0);                                   % Stop motor
stop(mymotor2,0);
pause(0.2);
r1 = double(readRotation(mymotor1));
r2 = double(readRotation(mymotor2));
end
```

На крају, функција Rotate за жељену вредност ротације робота, *PP*, враћа вредности ротације точкова, r1 и r2.

10.2 Функција *GoStraight*

Ова функција ради на сличном принципу као и функција Rotate. Разлике су у томе што се овде као улазни параметар уноси жељени пређени пут робота *PP* и мотори точкова подешавају се на брзину истог интензитета и смера, за разлику од ротације где су оне супротног знака, једна је позитивне а друга негативне вредности. Кретање унапред задаје се знаком променљиве *PP*: уколико је потребно да се робот креће унапред, *PP* задајемо као позитивну вредност, и онда се та информација чува у a = 1, а ако је потребно да се креће уназад, *PP* задаје се као негативна вредност и то се по програму смешта у a = -1. Као и код функције *Rotate* за жељени пређени пут, функција *GoStraight* враћа укупне углове ротација левог и десног точка очитане са енкодера мотора, $r_1 r_2$.

```
function [ r1,r2] = GoStraight(PP, mymotor1, mymotor2 )
PP=PP*10;
D=43;% PRECNIK TOCKA U MM
MaxSPEED=30;
if PP>0 % UKOLIKO SE KRECE U NAPRED
    SPEED =MaxSPEED;% Motor speed
    a=1;
else   % UKOLIKO SE KRECE U NAZAD
    SPEED =-MaxSPEED;% Motor speed
    a=-1;
end
```

```
Fi=round(PP*360/(pi*D)); % Odredjivanje ugla vratila koji za koji treba motor
da rotira


resetRotation(mymotor1);                    % Reset motor rotation counter
resetRotation(mymotor2);

mymotor1.Speed = SPEED;                     % Set motor speed
mymotor2.Speed = SPEED;

start(mymotor1);                            % Start motor
start(mymotor2);

_________________
if a==1
    % IDI U NAPRED
    while 1
        r1 = readRotation(mymotor1);            % Read rotation counter in
degrees
        r2 = readRotation(mymotor2);

        r=(r1+r2)/2;

        error=Fi-r;
%         error_dot=error-error_old;
%         E=E+error;
        SPEED=kP*error;%+kD*error_dot+kI*E
%         error_old=error;

        if SPEED>=MaxSPEED
            SPEED=MaxSPEED;
        elseif SPEED <=5
            break;
        end

        mymotor1.Speed = SPEED;                     % Set motor speed
        mymotor2.Speed = SPEED;
    end
```

```
else
    % IDI U NAZAD!
    while 1
        r1 = readRotation(mymotor1);
       r2 = readRotation(mymotor2);

        r=(r1+r2)/2;

        error=Fi-r;
%         error_dot=error-error_old;
%         E=E+error;
        SPEED=kP*error;%+kD*error_dot+kI*E
%         error_old=error;

        if SPEED<=-MaxSPEED
            SPEED=-MaxSPEED;
        elseif SPEED >=-5
            break;
        end

        mymotor1.Speed = SPEED;                    % Set motor speed
        mymotor2.Speed = SPEED;
    end
end
%KRAJ
r1=double(r1);
r2=double(r2);
stop(mymotor1,0);                          % Stop motor
stop(mymotor2,0);
pause(0.2);
r1 =double(readRotation(mymotor1));
r2 =double(readRotation(mymotor2));
end
```

10.3 Управљање мотором

Да би се мотор активирао, користи се код ‘*EV3Motors*’ који се налази у наставку. У њему се користе функције *motor*, *resetRotation* и *readRotation*. Помоћу прве функције улазима B I C додељују се функције мотора и њима се управља променљивима *mymotor1* , *mymotor2*, друга је потребна да би се енкодер вратио на нулту позицију, а помоћу треће се након рада мотора очитава резултат са енкодера и чува у промењиве r_1 и r_2. Пре пуштања било ког кода, потребно је прикључити робот преко USB порта и покренути прву линију у овом коду. Већина функција и команди из овог кода користе се појединачно у кодовима *GoStraight* и *Rotate*.

```
% Uspostavljanje komunkacije sa upravljackom jediniciom
myev3 = legoev3('usb');

%Komunkiacija sa motor, pazite na PORT!!! u koji je motor prikljucen ("A",
"C")
mymotor1 = motor(myev3, 'B');
mymotor2 = motor(myev3, 'C');

% Resetovanje enkodera - Vracanje na 0
resetRotation(mymotor1);
resetRotation(mymotor2);

SPEED =30;

mymotor1.Speed = SPEED; % Podesavanje brzine motora
mymotor2.Speed = -SPEED;

start(mymotor1); % Start motora
start(mymotor2);

  r1 = readRotation(mymotor1); % Ocitavanje sa enkodera
  r2 = readRotation(mymotor2);

stop(mymotor1,0);% Zaustavljanje motora
stop(mymotor2,0);
```

10.4 Управљање сензорима

У овом коду се успоставља веза са физичким сензорима. Помоћу оптичког сензора, могуће је детектовати интензитет светлости (променљива *intensity*) и рефлектовану светлост (што се чува у променљивој *Ref_intensity*). Помоћу прве *for* петље могуће је, у току првог кретања, прикупити информације о рефлектованој светлости на пређеној путањи робота. Када је потребно да сензор детектује боју, покреће се трећа линија кода. Вредност боје чува се у променљивој *color* типа *string*.

На сличан начин, у следећа два сегмента кода успоставља се веза са инфрацрвеним сензором и сензором додира и могуће је помоћу *for* петље очитавати удаљеност од објекта у току кретања, као и активност сензора додира у случају да робот наиђе на препреку.

У овом коду су, помоћу горе описаних функција, прикупљени обучавајући парови за задатак класификације боје површине по којој се робот креће. У последњој *for* петљи, робот се покреће дуж путање (црне или беле боје), зауставља и детектује интензитет светлости у опсегу 0-100. За површину беле боје дуж путање, интензитети се крећу од 50 до 100 а за црну од 0 до 50. Улазни вектор обучавајућег пара чини комбинација „белог“ и „црног“ вектора. Излазни вектор је дефинисан тако да за белу површину, тј. вредност интензитета од 50 до 100, има вредност [1 0] а за црну [0 1]. На тај начин формирани су обучавајући парови и могуће је обучити мрежу за класификацију боје површине на црну или белу. С обзиром на то да се ради о задатку класификације, поред ‘*back propagation’*, могуће је користити и ‘*perceptron*’ вештачку неуронску мрежу.

```
%TEST SENZORA
% Uspostavljanje komunkacije sa upravljackom jediniciom
myev3 = legoev3('usb')

% COLOR SENZOR - OPTICKI SENZOR
%
mycolorsensor = colorSensor(myev3); % Uspostavljanje komunkacije sa color
senzorom

color = readColor(mycolorsensor); %u promeljivu "color" pise naziv boje

%Osvetljenje
intensity = readLightIntensity(mycolorsensor); % Ocitavanje inteziteta
svetlosti

% Vrednost reflektovane svetlosti
Ref_intensity = readLightIntensity(mycolorsensor,'reflected');

for i=1:30
```

```
    Ref_inte=readLightIntensity(mycolorsensor,'reflected')
     pause(0.2)
end

% INFRA RED SENZOR - PROKSIMITI SENZOR - SENZOR DUZINE

myirsensor = irSensor(myev3); %Komunkacija sa IR senzorom

Distance=readProximity(myirsensor); % Ocitanje distance u cm

for i=1:30
    Dis=readProximity(myirsensor)
    pause(0.2)
end


% TOUCH senzor

mytouchsensor = touchSensor(myev3);% Komunkacija

PUSH=readTouch(mytouchsensor); % 1 za pritisnuto
                               % 0 za ne pritisnuto


for i=1:30
    PUSH=readTouch(mytouchsensor)
     pause(0.2)
end

vektor_beli = [];

for i=1:99
    Ref_intensity = readLightIntensity(mycolorsensor,'reflected');
    vektor_beli = [vektor_beli; Ref_intensity];
    GoStraight(15, mymotor1, mymotor2);
    pause(1)
end
```

10.5 Калманов филтер

Следећи задатак је предвиђање стања и матрице коваријанси, тј. корак предикције Калмановог филтера.

Прво је формирана функција *dkp* која за дати вектор положаја, *x, y* и *θ*, и за параметре *ugaoL* и *ugaoD* даје предвиђено стање μ (променљива *mu* у коду). Параметри *ugaoL* и *ugaoD* представљају излаз из функција *GoStraight* или *Rotate*, тј. r_1 *и* r_2, углове очитане са енкодера левог и десног точка. На основу њих се у функцији рачуна пређени пут левог и десног точка и на основу модела кретања одређују предвиђена позиција и оријентација, тј. предвиђени положај.

```
function [xv] = dkp(x,y,teta,ugaoL,ugaoD)
b=10.5; %Rastojanje izmedju tockova
D=4.3;% PRECNIK TOCKA U MM

dsl = ugaoL/360*pi*D;
dsd = ugaoD/360*pi*D;

ds = (dsl+dsd)/2;

x1 = x+ds*cos(teta+(dsd-dsl)/(2*b));
y1 = y + ds*sin(teta+(dsd-dsl)/(2*b));
teta1 = teta + (dsd-dsl)/b;
xv = [x1,y1,teta1]';
end
```

Да би се израчунала матрица коваријанси, потребно је израчунати матрице Јакобијана положаја и управљања. На основу формуле, матрица бројних вредности G_t добија се за дате углове *ugaoL* и *ugaoD* и почетну оријентацију *θ* те је на основу ове чињенице лако конструисати функцију која са овим параметрима враћа матрицу извода по положају, тј. матрицу Јакобијана G_t.

```
function[Gt]=jakobijan_polozaja(ugaoL,ugaoD,teta)

b=10.5; %Rastojanje izmedju tockova
D=4.3;% PRECNIK TOCKA U MM

dsl = ugaoL/360*pi*D;
dsd = ugaoD/360*pi*D;

ds = (dsl+dsd)/2;
Gt=zeros(3);
```

```
Gt(1,:)=[1,0,-ds*sin(teta+(dsd-dsl)/(2*b))];
Gt(2,:)=[0,1,ds*cos(teta+(dsd-dsl)/(2*b))];
Gt(3,:)=[0,0,1];
end
```

На сличан начин се конструкцијом функције *jakobijan_upravljanje* добија матрица бројних вредности *Vt* која је матрица извода по управљачким величинама, *dsl* и *dsd*.

```
function[Vt]=jakobijan_upravljanje(ugaoL,ugaoD,teta)
b=11.1; %Rastojanje izmedju tockova
D=4.3;% PRECNIK TOCKA U MM

dsl = ugaoL/360*pi*D;
dsd = ugaoD/360*pi*D;

Vt = [ cos(teta + (dsd - dsl)/(2*b))/2 + (sin(teta + (dsd -
dsl)/(2*b))*(dsd/2 + dsl/2))/(2*b), cos(teta + (dsd - dsl)/(2*b))/2 -
(sin(teta + (dsd - dsl)/(2*b))*(dsd/2 + dsl/2))/(2*b);
 sin(teta + (dsd - dsl)/(2*b))/2 - (cos(teta + (dsd - dsl)/(2*b))*(dsd/2 +
dsl/2))/(2*b), sin(teta + (dsd - dsl)/(2*b))/2 + (cos(teta + (dsd -
dsl)/(2*b))*(dsd/2 + dsl/2))/(2*b);

-1/b, 1/b];

end
```

За рачунање предвиђене матрице коваријанси, потребно је израчунати матрицу шума за шта је потребно да се добија матрица шума *Mt*.

```
%racunamo predvidjenu matricu kovarijansi

function[Mt]=matrica_suma( ugaoL,ugaoD)
b=9.5; %Rastojanje izmedju tockova
D=4.3;
dsl = ugaoL/360*pi*D;
dsd = ugaoD/360*pi*D;

%Mt=[(0.5*abs(dsl)+0.5*abs(dsd))^2, 0;
%   0, (0.5*abs(dsl)+0.5*abs(dsd))^2];
```

```
Mt = 0.1*abs([dsd 0;
      0 dsl]);
end
```

У функцији предикција, позивају се претходно конструисане функције и на основу формула:

$$\overline{\mu_t} = g\left(u_t, \mu_{t-1}\right)$$

$$\overline{\Sigma_t} = G_t \overline{\Sigma_{t-1}} {G_{t-1}}^T + V_t M_t {V_{t-1}}^T$$

рачунају се предвиђено стање и предвиђена матрица коваријанси.

```
% sad cemo da racunamo tacku 7, tj predvidjeno stanje
%to je formula 7 na slajdu 36 na prezentaciji PA1

% predvidjeno stanje, koje se racuna preko modela kretanja, je u sustini
% rezultat funkcije dkp tako da cemo samo sacuvati taj rezultat
function[Sigma_t,mu]=predikcija(ugaoL,ugaoD,x,y,teta, Sigma )
b=10.5; %Rastojanje izmedju tockova
D=4.3;
mu=dkp(x,y,teta,ugaoL,ugaoD);


% konkretno predvidjeno stanje (t) dobicemo tako sto cemo u zagrati uneti
% parametre pocetno stanja (t-1)

Mt=matrica_suma(ugaoL,ugaoD);
Gt=jakobijan_polozaja(ugaoL,ugaoD,teta);
Vt=jakobijan_upravljanje(ugaoL,ugaoD,teta);

Sigma_t=Gt*Sigma*(Gt')+Vt*Mt*(Vt');
end
```

10.6 Опис кода главног програма

На основу дефинисаних улазаних података, а покретањем следећег алгоритма прво се израчунава оптимална путања применом алгоритма претраживања A* од почетне тачке па до прве међутачке узимајући је као да је циљна. Излаз из алгоритма претраживања A* је матрица са кординатама додатних међутачака које одређују трајекторију, *Putanja*. Ова петља ивршаваће се све док не дође до краја вектора *xc* што значи кроз све наведене пикселе који дефинишу трајекторију.

```
for i=1:(size(xc,2)-1)
  Putanja=a_zvezda_euklid(xc(i)+1,yc(i)+1,xc(i+1)+1,yc(i+1)+1);

  pathx = Putanja(1:end,1)*Xpiksel;% %svi redovi u prvoj koloni
  pathy = Putanja(1:end,2)*Ypiksel;% %svi redovi u drugoj koloni
```

За сваку од тих међутачака из наведене матрице, упоређујући тренутну позицију и оријентацију са наредном тачком, у наставку алгоритма, израчунава се растојање *d_wp* и потребна оријентација робота, *fi_steer*. Испод је приказан део управљачког алгоритма који извршава ову операцију. Ова петља ивршаваће се за све тачке у које робот мора да дође између два задата пиксела трајекторије, предвиђене А звезда алгоритмом.

```
for k = 1 : sizpath

    d_wp = ITS_compute_distance(x, path(k,:));
    fi_steer = ITS_compute_direction(x,path(k,:))*57.3;
```

Ако се пре почетка кретања тренутна орјентација робота разликује за више од *fi_steer* тада се прво изврши ротација робота пре кретања унапред. Тада ротацију вратила мотора одређује вештачка неуронска мрежа (ВНМ) која је обучена за ову намену.

```
    alrot = round((sim(net_rotacija,(fi_steer+153)/313))*360-180);
```

Када се изврши ротација, тј. ако је и има, врши се наредно кретање унапред све док се растојање до наредне тачке не смањи за мање или једнако од вредности *d_wp*. Део управљачког кода који је покривен овим објашњењем приказан је испод.

Све време, од прве задате тачке до друге, дуж путање израчунате А звезда алгоритмом, извршава се наредни алгоритам управљања све док се не достигне положај наредне задате тачке.

1. Уколико је достигнута позиција али не и оријентација извршава се код описан у наставку.

```
if abs(fi_steer) >= 15 && d_wp < dmax

            s = readLightIntensity(mycolorsensor,'reflected');
            Yp =sim(net_boja,double(s)/100);
            [~,Yp] = max(Yp);
            yp= [yp;Yp];

            alrot = round((sim(net_rotacija,(fi_steer+153)/313))*360-180);

            [MC,MB]=Rotate( alrot,mymotor1,mymotor2);
            %mymotor1=B=desni
            %mymotor2=C=levi



            stop(mymotor1,0); % Stop motor
            stop(mymotor2,0);

            dsl = MC/57.3*r; dsr = MB/57.3*r;
            M = [10^-5*abs(dsl) 0;0 10^-5*abs(dsr)];
            [C,x] = predikcija(MC,MB,x(1),x(2),x(3), C ); % dodati svoj model
kretanja

            %plot_simulation_data(x, C),hold on;



            s = readLightIntensity(mycolorsensor,'reflected');
            Yp =sim(net_boja,double(s)/100);
            [~,Yp] = max(Yp);
```

```
        yp= [yp;Yp];

    end
```

Вештачка неуронска мрежа на основу угла *fi_steer*, који представља разлику оријентација у тренутном и наредном положају, предвиђа угао за који је потребно да се заротира робот, *alrot*. На основу њега робот се ротира, помоћу функције *Rotate*, и вредности угла ротације очитаних са енкодера левог и десног точка користе се за рачунање следећег положаја *x* и предвиђене вредности матрице коваријансе *C* на основу модела кретања помоћу функције *predikcija*.

2.Уколико услов у претходној тачки није испуњен, улази се у петљу све док се не постигне жељена тачност позиције

```
while d_wp > dmax
        s =readLightIntensity(mycolorsensor,'reflected');
        Yp =sim(net_boja,double(s)/100);
        [~,Yp] = max(Yp);
        yp= [yp;Yp];
        if Yp==1

            [x,C]=korigovanje_polozaja(x,C,Q,mapa1);

        end

        [MB,MC]=GoStraight( 2,mymotor1,mymotor2);
        stop(mymotor1,0);
        stop(mymotor2,0);

        dsl =MC/57.3*r; dsr = MB/57.3*r;

        M = [10^-5*abs(dsl) 0;0 10^-5*abs(dsr)];

        [C,x] = predikcija(MC,MB,x(1),x(2),x(3), C );

        plot_simulation_data(x, C),hold on;
```

У овој петљи се на сличан начин, робот креће помоћу функције *GoStraight*, подаци са енкодера о оствареном углу ротације при праволинијском кретању прослеђују се функцији модела кретања, predikcija, и подаци о предвиђеном положају и матрици коваријанси чувају се у променљивама *x* и *C*.

3. Поред тога, у свим петљама и исказима кроз које програм пролази константно се врше очитавања интензитета рефлектоване светлости и чувају се у променљивој *s*. На основу те вредности и обучене мреже за класификацију детектоване светлости на црну и белу, у променљиву *Yp* смешта се вредност 0 или 1.

```
s = readLightIntensity(mycolorsensor,'reflected');
Yp =sim(net_boja,double(s)/100);
[~,Yp] = max(Yp);
yp= [yp;Yp];
```

У истој *while* петљи се проверава да ли је вредност *Yp* променљиве 1. Уколико јесте, значи да се ради о црној површини, што даље значи да је потребно да робот изврши корекцију положаја на основу Калмановог филтера, што се ради помоћу функције *korigovanje_polozaja*.

4. Уколико дође до кориговања положаја потребно је поново израчунати *d_wp* и *fi_steer* и уколико је испуњен услов за нову вредност *d_wp* изаћи из петље. Уколико није, и уколико је угао *fi_steer* већи од задате тачности, понавља се процедура описана у тачки 1.

```
if Yp==1

    [x,C] = korigovanje_polozaja(x,C,Q,mapa1);

end

d_wp = ITS_compute_distance(x, path(k,:));
fi_steer = ITS_compute_direction(x,path(k,:))*57.3;

if d_wp <dmax
    break;
end
if abs(fi_steer) >= 30
```

```
        s = readLightIntensity(mycolorsensor,'reflected');
        Yp =sim(net_boja,double(s)/100);
        [~,Yp] = max(Yp);
        yp= [yp;Yp];

        alrot = round((sim(net_rotacija,(fi_steer+153)/313))*360-180);

        [MC,MB]=Rotate( alrot,mymotor1,mymotor2);

        % Stop motor
        stop(mymotor1,0);
        stop(mymotor2,0);

        dsl = MC/57.3*r; dsr = MB/57.3*r;
        M = [10^-5*abs(dsl) 0;0 10^-5*abs(dsr)]; % control uncertainty
        %[x,C] = model_kretanja_grupa4(x, C, dsr, dsl, b, M);
        [C,x] = predikcija(MC,MB,x(1),x(2),x(3), C );

        if Yp==1


            [x,C] = korigovanje_polozaja(x,C,Q,mapa1);

        end
    end
    s = readLightIntensity(mycolorsensor,'reflected');
    Yp =sim(net_boja,double(s)/100);
    [~,Yp] = max(Yp);
    yp= [yp;Yp];

    d_wp = ITS_compute_distance(x, path(k,:));
    fi_steer = ITS_compute_direction(x,path(k,:))*57.3;
    pause(0.2)
end
```

11. Опис експерименталних резултата

Трајекторија 1

У овом делу пројекта, кориштен је А звезда алгоритам са еуклидском нормом за кретање робота по путањи бр. 1. У линијама кода 6 и 7 дефинисани су пиксели кроз које робот треба да прође при кретању и тиме је задата путања.

```
1  clc,clear,close all
2
3 load('net-rotacija.mat');
4 load('net-boja.mat');
5 load('mapa_okruzenja.mat');
6 xc=[3 12 12 11];
7 yc=[5 7 12 21];
8
9 Xpiksel=5;
10 Ypiksel=5;
```

Путања је прилагођена положајима најближих карактеристичних објеката (приказаних у променљивој *mapa1* у cm), да би робот имао прилику да по проласку кроз карактеристичне објекте изврши корекцију положаја и достигне позицију што ближу циљу. На пример, други задати пиксел је (12,7) што значи да се он налази у опсегу 55-60cm по *x* оси и 30-35 cm по *y* оси, а то одговара координатама првог карактеристичног објекта.

```
mapa1 = [55.5 30.5; 25 22; 63 57; 15 63; 65 102.5];      (11.1)
```

За описано окружење вршено је 5 понављања и графички прикази налазе са на сликама (11.1-11.5) а резултати експеримента у табели 8.

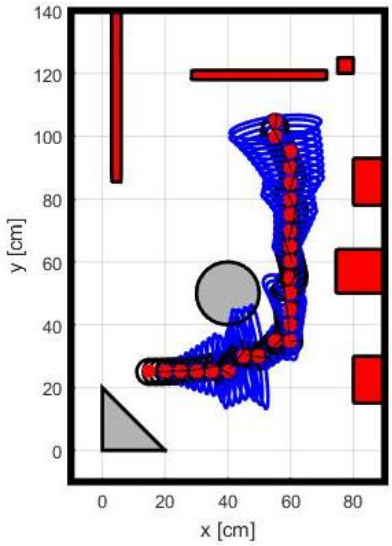
Табела 8. Експериментални резултати за трајекторију 1

редни бр. експеримента	1	2	3	4	5
x предвиђено (cm)	55.1404	55.0797	55.0934	55.1713	55.6114
y предвиђено (cm)	102.88	102.507	102.7718	102.4803	102.4677
θ предвиђено (°)	82.31	82.12	83.33	95.15	97.09
x остварено (cm)	54	57.102	53.5	53.2	58.8
y остварено (cm)	101.5	102.7	103.5	103.6	102.2
θ остварено (°)	80	75	80	92.5	95

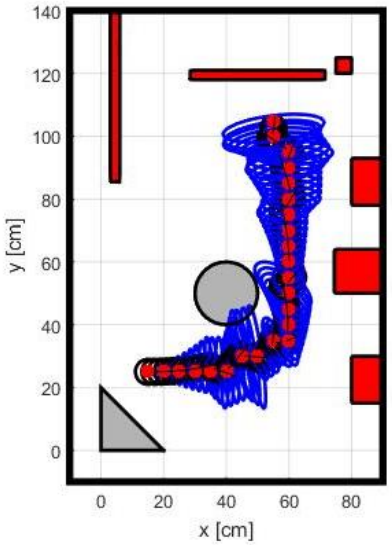
Као резултат кретања робота бележене су стварно остварена позиција и оријентација робота (у односу на координатни почетак окружења, уз помоћ метра и угломера мерене су позиција и оријентација центра масе робота), *x остварено (cm)*, *y остварено (cm)* и *θ остварено (°)*. Такође је разматрано предвиђање позиције и оријентације робота на основу модела кретања, тј. положај у који робот „мисли“ да је стигао. На основу претходно описаног кода за модел кретања, *predikcija,* као излаз из функције добија се предвиђена матрица коваријанси *C* и вектор положаја *x*. Када робот заврши кретање, он претпоставља да се налази у позицији и оријентацији описаним вектором положаја *x*. Координате вектора положаја у којима је робот моделом кретања предвидео да се налази бележене су у променљивима *x предвиђено (cm), y предвиђено (cm)* и *θ предвиђено (°)*.

```
[C,x] = predikcija(MC,MB,x(1),x(2),x(3), C );        (11.2)
```

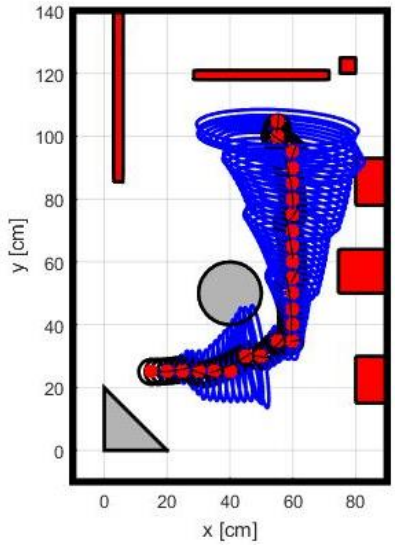
Са слика (11.1-11.5) се може приметити да је у свих 5 покушаја извршена корекција при проласку робота кроз карактеристични објекат. Квалитет остварене путање мери се на основу разлике остварених и предвиђених вектора положаја.



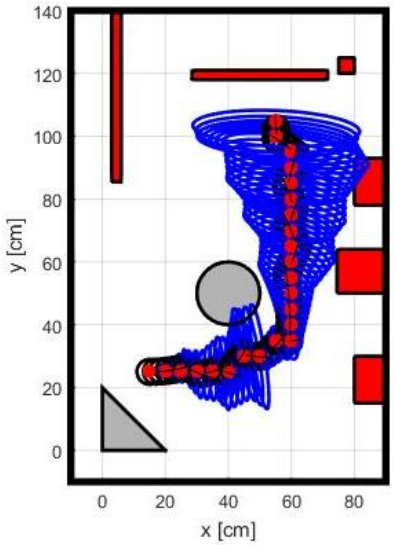
Слика 11.1 Прво понављање



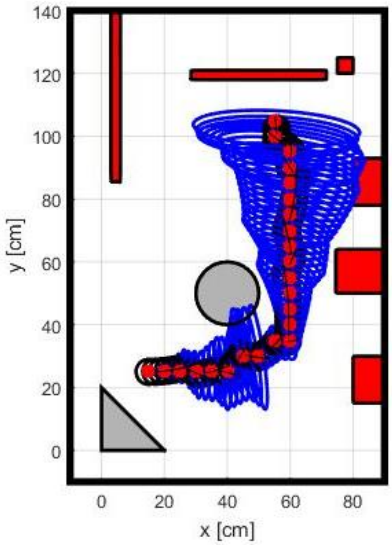
Слика 11.2 Друго понављање



Слика 11.3 Треће понављање



Слика 11.4 Четврто понављање



Слика 11.5 Пето понављање

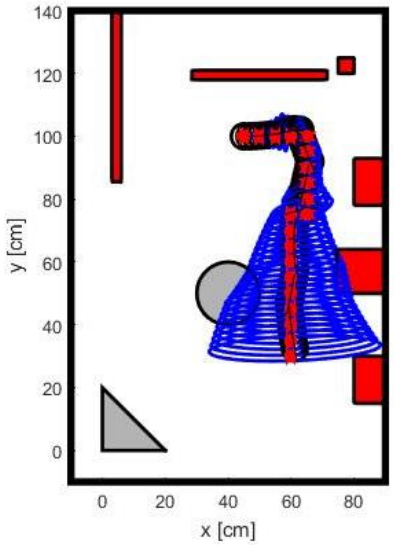
Трајекторија 2

У овом делу пројекта, кориштен је А звезда алгоритам са менхетн нормом за кретање робота по путањи бр. 2. У линијама кода 7 и 8 дефинисани су пиксели кроз које робот треба да прође при кретању и тиме је задата путања.

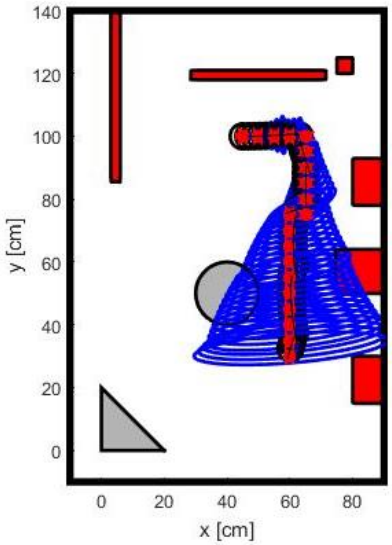
```
1  clc,clear,close all
2
3 load('net-rotacija.mat');
4 load('net-boja.mat');
5 load('mapa_okruzenja.mat');
6
7 xc=[9,13,13,12,12];
8 yc=[20,20,15,11,6];
9
10 Xpiksel=5;
11 Ypiksel=5;
```

Табела 9. Експериментални резултати за трајекторију 2

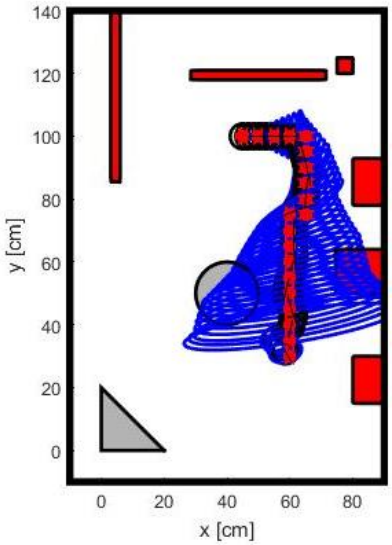
редни бр. експеримента	1	2	3	4	5
x предвиђено (cm)	61.16228585	60.15788387	58.58295269	60.17379038	60.71832551
y предвиђено (cm)	32.5098571	32.50765442	31.50939748	31.20161476	32.35310024
θ предвиђено (°)	-79.12924702	-106.4091322	-61.22148955	-88.30792625	-82.55136299
x остварено (cm)	63	60	61	73	66
y остварено (cm)	30	29	35.5	32	29
θ остварено (°)	-90	-100	-80	-85	-80



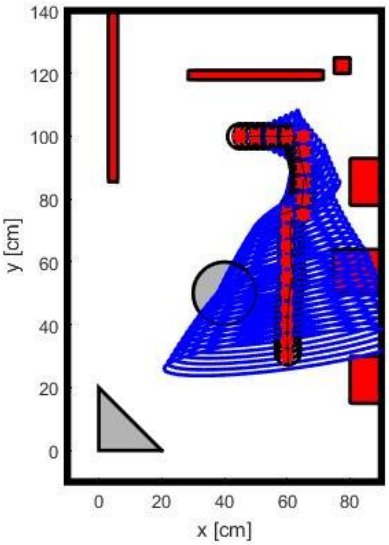
Слика 11.6 Прво понављање



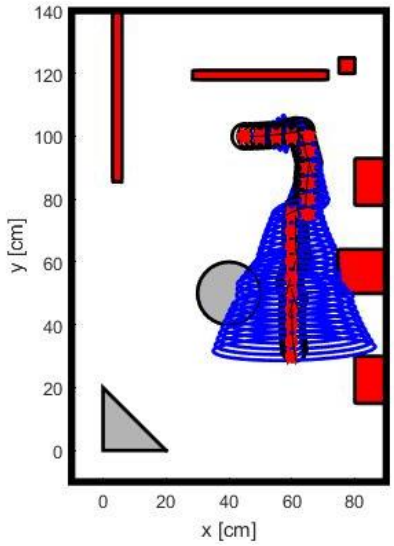
Слика 11.7 Друго понављање



Слика 11.8 Треће понављање



Слика 11.9 Четврто понављање



Слика 11.10 Пето понављање

С обзиром на то да је за ову трајекторију као један од пиксела одабран пиксел [13, 20] што одговара координатама [65, 100] видимо да је робот требало да прође кроз један од карактеристичних објеката и изврши корекцију положаја и смањи елипсу несигурности. Карактеристични објекат је на почетку кретања па се може променити релативно мало смањење елипсе пре скретања. Иако је корекција извршена, с обзиром да робот до краја путање не наилази ни на један други карактеристични објекат он не врши корекцију, а резултат тога је значајно већа елипса несигурности. Да би се смањила елипса несигурности робота, потребно би било увести још карактеристичних објеката на путањи којом се робот креће. У односу на еуклидску норму, менхетн норма не дозвољава дијагоналне пролазе из једног у други пиксел. У оквиру ове путање, робот је на два места требало да се заротира за тачно 90 степени, међутим као што је већ описано, робот се услед грешака које прави ретко окреће за тачно задату вредност чак иако се користила вештачка неуронска мрежа за предвиђање потребних углова за које робот треба да се заротира да би испунио захтев. У овом експерименту, дешавале су се релативно велика одступања у крајњој оријентацији коју је робот заузео и у којој мисли да јесте. Ако се погледа слика 11.8 и размотри правац путање задате А звезда алгоритмом (црвене звездице) и путање којом робот мисли да се креће види се да се оне значајно разликују у оријентацији што се одражава и на велику разлику између остварене и предвиђене оријентације приказане у табели 9.

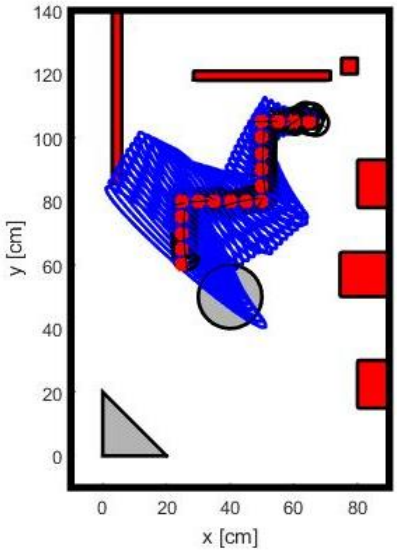
Трајекторија 3

У овом делу пројекта, кориштен је А звезда алгоритам са менхетн нормом за кретање робота по путањи бр. 3. У линијама кода 7 и 8 дефинисани су пиксели кроз које робот треба да прође при кретању и тиме је задата путања.

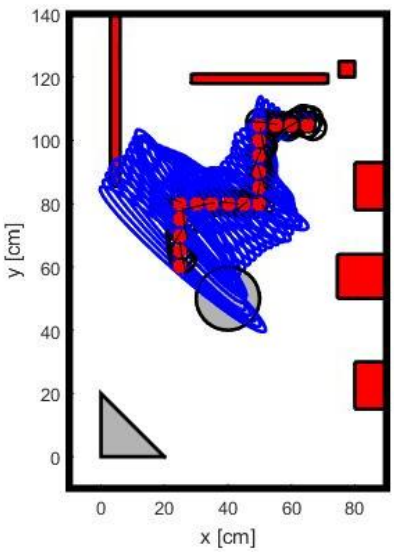
```
clc,clear,close all

load('net-rotacija.mat');
load('net-boja.mat');
load('mapa_okruzenja.mat');

xc = [13 11 10 7 5 5];
yc = [21 21 16 16 16 12];

Xpiksel=5;
Ypiksel=5;
```

Табела 10. Експериментални резултати за трајекторију 3

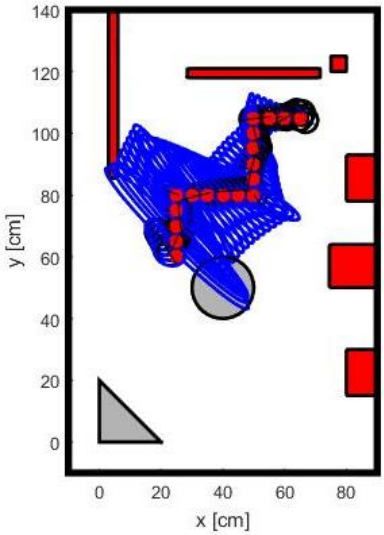
редни бр. експеримента	1	2	3	4	5
x предвиђено (cm)	26.0642	25.795	22.7153	22.9903	22.6383
y предвиђено (cm)	62.4278	62.4581	61.2349	60.3179	61.1009
θ предвиђено (°)	-83.37898089	-77.85286624	-56.68853503	-50.42292994	-56.96369427
x остварено (cm)	20.5	18	23.5	23	22.4
y остварено (cm)	63	65	60	59.5	60.5
θ остварено (°)	-96	-94.5	-50	-45	-51



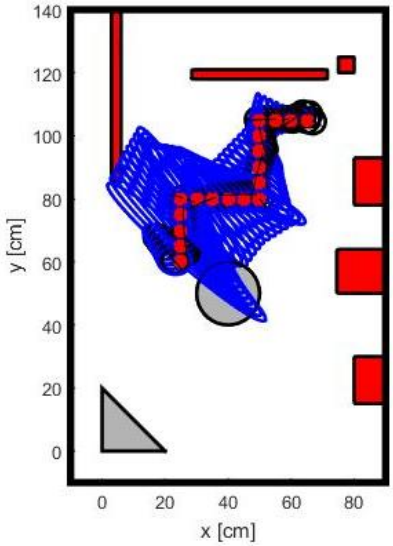
Слика 11.11 Прво понављање



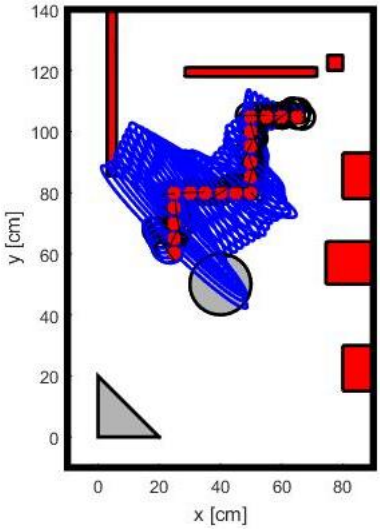
Слика 11.12 Друго понављање



Слика 11.13 Треће понављање



Слика 11.14 Четврто понављање



Слика 11.15 Пето понављање

У односу на претходне две трајекторије, ова трајекторија због менхетн норме, која не дозвољава кретање по дијагонали, има 3 ротације. Карактеристични објекат је такође постављен на почетку кретања те робот до краја кретања није успео да искоригује свој положај. Иако одступања од позиције нису значајно велика, из табеле 10 види се да одступања од задате оријентације јесу, поготово уколико упоређујемо са претходна два кретања. Овакви резултати говоре да је, с обзиром да су углови одступања *fi_steer* на основу којих се ради корекција оријентације у крајњем положају релативно мали у односу на опсег углова за који је коришћена мрежа обучавана, потенцијално решење у проблему оријентације додатно обучавање мреже за што више углова у опсегу од 0 до 20 степени.

Такође се из последњег експеримента види да је менхетн норма у односу на еуклидску мање погодна уколико се проблем великог одступања у оријентацији не може решити.

Поред потенцијалног решења у повећању броја обучавајућих парова, могуће објашњење се налази у чињеници да код функције *Rotate* и *GoStriaght* робот зауставља кретање уколико је разлика између жељене и остварене вредности 5 степени.

12. Закључак

У овом пројектом задатку можемо видети да су неуронске мреже есенцијални део интелигентног понашања овог мобиног робота. Функционална апроксимација, била је једна од кључних примена мрежа, јер је она искоришћена да би робот „знао“ какво управљање да зада моторима како би се заротирао за потребни угао. Друга врло важна примена је класификација, која је имплементирана при разумевању података очитаних са оптичког сензора. Коришћењем модела кретања и очитавања са унутрашњих сензора (енкодера), омогућено је познавање позиције и оријентације робота у свим тренуцима његовог кретања. Уз помоћ Калмановог филтера вршена је корекција положаја, тј. он је уклањао неусаглашености између израчунатог положаја и реално оствареног положаја. Преко A* алгоритма генерисана је оптимална путања између почетних и крајних тачака, уз избегавање препрека у окружењу.

Након више извршених лабораторијских експеримената, добијени резултат представља интелигентног мобилног робота који може извршити потребну оријентацију и задато кретање у датом технолошком систему, без асистенције човека, шга чини и аутономним.

Међутим, може се приметити да тачност извођења кретања, која је у толерантним вредностима, ипак одступа од задатих вредности. То се може приписати квалитету и тачности израде самих компоненти од којих је робот израђен, па ако би робот био израђен од поузданијих компоненти и тачност извођења кретања била би много виша.

Литература

[1] H.H. Yu,J-H. Hwang, *Handbook of neural network signal processing.* London : CRCPress, 2002.

[2] E. Alpaydin, (2004) Introduction to Machine Learning, The MIT Press, Cambridge, Massachusetts London, England

[3] Роботика и вештачка интелигенција, З. Миљковић, Н. Славковић, М. Петровић, (2020) Предавања за сваку лекцију (handouts). Машински факултет, интерна скрипта Мастер 4.0

[4] John M.Holland, *Designing Autonomous mobile Robots, Inside the Mind of an Intelligent machine,* PDF књига преузета са сајта https://doc.lagout.org/science/ , дана 9.03.2021.

[5] Машинско учење интелигентних роботских система, З. Миљковић, М. Петровић, (2021) Предавања за сваку лекцију (handouts). Машински факултет, интерна скрипта Мастер 4.0

[6] http://www.wikipedia.com , последњи пут приступљено 9.06.2021.

[7] https://www.mathworks.com/products/matlab.html, последњи пут приступљено 15.05.2021.

[8] https://education.lego.com/ , последњи пут приступљено 17.04.2021.

[9] http://www.legoengineering.com/ev3-sensors/ , последњи пут приступљено 17.04.2021.

[10] Марко Кованџић, Семинарски рад на тему: *Калманов филтер,* PDF верзија преузета са сајта http://www.marko.rs/author/files/KALMAN.pdf , дана 2.05.2021.

[11] Бојан Р. Бабић, *Рачунарски интегрисани системи и технологије,* Машински факултет, Београд, 1. издање (2017)