Универзитет у Београду
Машински факултет

Мастер академске студије

Модул за производно машинство

ИНТЕЛИГЕНТНИ ТЕХНОЛОШКИ СИСТЕМИ

П Р О Ј Е К А Т

Оцена пројектног задатка:	Предметни наставници:	проф. др Зоран Миљковић и проф. др Бојан Бабић		
	Предметни сарадници:	доц. др Божица Бојовић, Најдан Вуковић, дипл. маш. инж., Марко Митић, дипл. маш. инж., Милица Петровић, дипл. маш. инж.		
	Група: 4			
Потпис наставника:	РБ	Презиме и име:	Бр.инд.	Потпис:
	1.	Драмићанин Предраг	1196/09	
	2.	Модец Виктор	1033/09	
	3.	Остојић Лазар	1034/09	
	4.	Живковић Бојан	1192/09	
	5.	Вучетић Мирко	1195/09	

Школска година: 2010/2011.

УНИВЕРЗИТЕТ У БЕОГРАДУ – МАШИНСКИ ФАКУЛТЕТ
Дипломске академске студије – 2. година
Модул: **ПРОИЗВОДНО МАШИНСТВО**, шк. год. 2010/2011.
Предмет: **ИНТЕЛИГЕНТНИ ТЕХНОЛОШКИ СИСТЕМИ (ПРО220-0131)**
Предметни наставници: **проф. др Зоран Миљковић** и **проф. др Бојан Бабић**

ПРОЈЕКТНИ ЗАДАТАК
(1/2)

Ради успостављања унутрашњег транспорта материјала, сировина и готових делова у оквиру експерименталног модела технолошког окружења „XY" применом интелигентних мобилних робота, на мобилном роботу *LEGO Mindstorms NXT* урадити следеће:

1. Формирати конфигурацију мобилног робота;

2. Развити и имплементирати модел кретања у *Matlab* окружењу;

3. Развити и имплементирати опсервациони (сензорски) модел мобилног робота применом система вештачких неуронских мрежа;

4. Применити алгоритам Калмановог филтера у циљу одређивања положаја мобилног робота у окружењу;

5. Имплементирати *A** алгоритам претраге;

6. У експерименталном моделу технолошког окружења верификовати резултате.

За дату диспозицију технолошког окружења „XY":

1. Развити симулациони модел у *Anylogic* окружењу;

2. На основу резултата симулације предложити нови диспозициони план технолошког окружења (задржати исте машине) и формирати „троугаону" матрицу за нови модел;

3. Упоредити два модела диспозиционог плана на основу резултата симулације и дати закључак.

Решењем пројектног задатка обухватити:

1. Основни циљ пројекта;
2. Теоријску поставку проблема и анализу;
3. Тестирати и верификовати перформансе експерименталних и симулационих решења;
4. Дискутовати резултате и дати закључак;

Напомене:

1. Пројекат ће бити позитивно оцењен ако и само ако приликом одбране пројектних задатака пројектно решење омогући несметано функционисање мобилног робота у окружењу;
2. Студенти су у обавези да на предавања и вежбе дођу припремљени јер ће у супротном коначан исход пројектних активности бити негативан;
3. Иницијатива студената у погледу предлога решења проблема, као и у погледу рада на додатним проблемима је више него пожељна, па ће стога сваки додатни рад бити узет у обзир приликом формирања завршне оцене;
4. Рокови израде појединачних пројектних целина дефинисани су планом и програмом предмета (Course Outline);
5. Сва питања, сугестије и евентуалне проблеме предочити у директном контакту са предметним наставницима, проф. др Зораном Миљковићем и проф. др Бојаном Бабићем, као и путем електронске поште на zmiljkovic@mas.bg.ac.rs, bbabic@mas.bg.ac.rs, а посебно у разговору са сарадницима у настави и на е-пошту: nvukovic@mas.bg.ac.rs, bbojovic@mas.bg.ac.rs, mmpetrovic@mas.bg.ac.rs и mmitic@mas.bg.ac.rs.

Задатак издао: _____

(Најдан Вуковић)

Предраг Драмићанин[1], Виктор Модец[2], Лазар Остојић[3], Бојан Живковић[4], Мирко Вучетић[5]

Резиме

У овом пројектном раду задатак је био да се у експерименталном моделу технолошког окружења изврши имплементација интелигентног мобилног робота, који за циљ има да замени виљушкар којим се обављао транспорт у технолошком окружењу. Решење овог проблема заснива се на употреби мобилног робота *LEGO Mindstorms NXT*, а за његово управљање коришћен је *RWTH Toolbox* уз софтверско окружење *MATLAB*. За апроксимацију примљене сензорске информације коришћене су вештачке неуронске мреже [3]. Применом Калмановог филтера решен је проблем локализације робота, како би робот могао знати где се налази. За одабир оптималне путање за задату почетну и циљну тачку коришћен је *A** алгоритам претраге. Софтверским пакетом *AnyLogic* симулирани су транспортни токови материјала унутар технолошког окружења *layout* производно-монтажног погона L3 Фабрике Металних Производа (ФМП). За обављање постављеног задатка мобилног робота, на основу успешних експерименталних и симулационих резултата, добијена је задовољавајућа тачност транспорта материјала у оквиру технолошког окружења.

Кључне речи: *интелигентни технолошки системи, интелигентни мобилни робот, вештачке неуронске мреже, Калманов филтер, A* алгоритам претраге, технолошко окружење.*

[1] **Предраг Драмићанин 1196/09**, Универзитет у Београду – Машински факултет, студент друге године Мастер академских студија.
Е-пошта: pdramicanin@gmail.com
[2] **Виктор Модец 1033/09**, Универзитет у Београду – Машински факултет, студент друге године Мастер академских студија.
Е-пошта: viktor23@ymail.com
[3] **Лазар Остојић 1034/09**, Универзитет у Београду – Машински факултет, студент друге године Мастер академских студија.
Е-пошта: l.ostojic33@yahoo.com
[4] **Бојан Живковић 1192/09**, Универзитет у Београду – Машински факултет, студент друге године Мастер академских студија.
Е-пошта: bozixxx@gmail.com
[5] **Мирко Вучетић 1195/09**, Универзитет у Београду – Машински факултет, студент друге године Мастер академских студија.
Е-пошта: mirkovucetic@gmail.com

Списак слика:

Списак табела:

Садржај:

1. Увод

Пројектни задатак обухвата решавање проблема транспорта материјала, сировина, полуфабриката или готових делова у фабрици "XY".

Овај задатак се односио на програмирање робота (конфигурисаног од компонената "*LEGO"* комплета) који ће се кроз задато окружење кретати путањама које су јасно дефинисане, како не би дошло до колизије са препреком. Робот који се користи је *LEGO Mindstorm NXT* [7], а његово управљање се извршава помоћу функције *RWTH Toolbox-a* [8] и алгоритама развијених и имплементираних у софтверском пакету *MATLAB* [9].

Друго поглавље овог рада обухвата поставку проблема која се односи на опис технолошког окружења, где је описан распоред машина и ток материјала. Дат је кратак опис робота, сензора и мерног система робота.

Треће поглавље се односи на приказ концепцијског решења постављеног задатка у виду блок дијаграма и обухвата конфигурисање мобилног робота, модел кретања, вештачке неуронске мреже, Калманов филтер, A* алгоритам и симулацију рада технолошког система у софтверу *AnyLogic*.

Четврто поглавље се односи на опис *LEGO* компоненти и конфигурисање мобилног робота.

У петом поглављу је описан математички модел кретања мобилног робота.

Шесто поглавље представља употребу вештачких неуронских мрежа, њихово дефинисање, обучавање и одабир за препознавање боје и управљање кретањем.

Седмо поглавље се односи на коришћење Калмановог филтера, за решавање проблема локализације робота.

У осмом поглављу се говори о алгоритмима претраживања, опис одабраног A* алгоритма и његова имплементација у пројектно решење.

Девето поглавље описује употребу *AnyLogic* софтверског пакета за сврхе симулације приказаног производног процеса посматраног предузећа, представљен је опис транспортних токова, диспозиционог плана и предложеног новог окружења након извршене симулације.

У десетом поглављу је извршена експериментална верификација пројектног задатка.

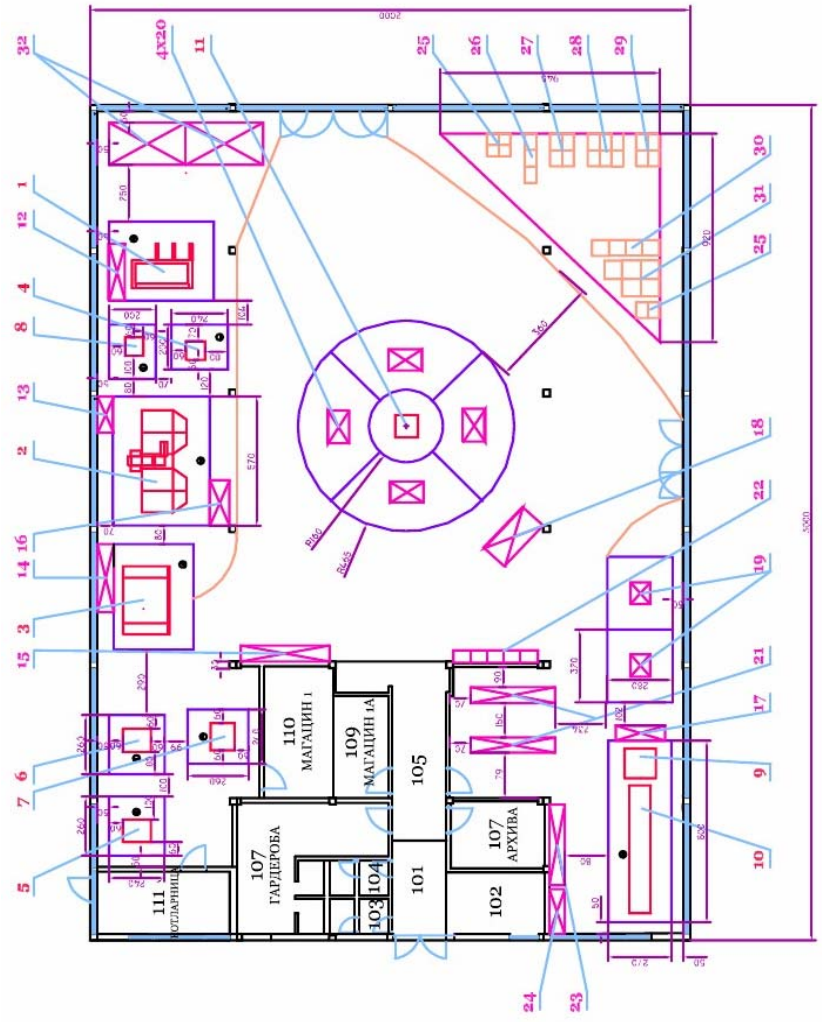
Једанаесто поглавље преставља закључак целокупног пројектног задатка.

2. Поставка проблема

2.1. Опис технолошког окружења

У овом раду за реализацију задатка коришћено је реално предузеће „XY“ које се бави производњом и монтажом *SIVACON* електро-ормана, где се целокупни процес од производње делова до монтаже одвија према *Siemens AG* [10] технолошком поступку.

Целокупни распоред машина у предузећу разврстан је према редоследу технолошких операција, што подразумева да се машине и уређаји постављају у линијском распореду односно међузависности машина и уређаја које се користе у том погону. Иако је производна хала изграђена пре дефинисања пројектног решења, добијени су задовољавајући резултати у погледу распореда машина, односно у погледу оптимизације транспортних путева унутар погона [4].



Слика 2.1.1: *Layout* производног монтажног предузећа "XY"

У табели 1, описан је *Layout*-а производног монтажног предузећа "XY".

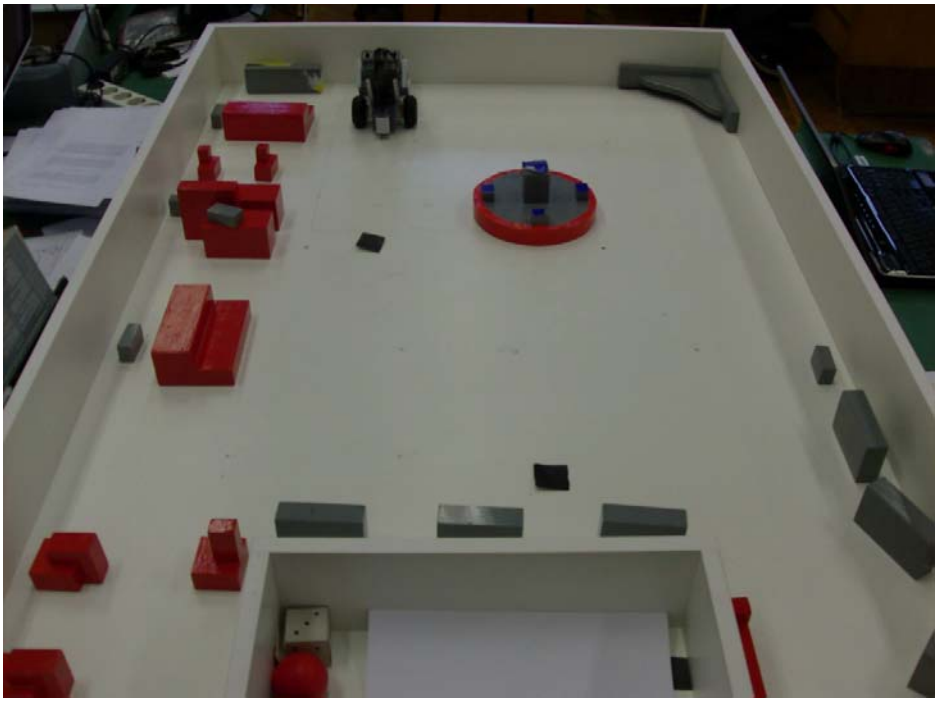
Табела 1: Опис *Layout*-а производног монтажног предузећа "XY"	
Позиција	Назив
1	Маказе за сечење
2	CNC машина за пробијање и просецање
3	CNC хидраулична „апкант“ преса
4	Машина за исецање профила
5	Стубна бушилица
6	Стубна бушилица
7	Кружна тестера
8	Оштрилица
9	Линија за обраду делова од лима
15	Међускладиште
18	Радни сто за формирање подсклопова
19	Монтажна целина са два радна места
20	Монтажна целина са четири радна места
21	Складиште готових делова од лима (полупроизводи)
22	Складиште вијцане робе и компонената за монтажу
23	Складиште готових делова од бакра
25	Складиште готових производа (SIVACON електро-ормана)
31	Складиште готових производа (SIVACON електро-ормана)

Из производног монтажног предузећа "XY" слика 2.1.1, издвојене су основне машине приказане у табели 2.

Табела 2: Машине у погону предузећа	
Позиција	Опис
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

На основу пројектованог *layout*-а датог на слици 2.1.1. може се уочити да су у линијском распореду битне машине које се користе за обраду челичног лима, машине алатке М1, М2, М3 и М4, а да су издвојене машине М5, М6 и М7, које су у мањој употреби, као и то да је линија за обраду бакра (М9) потпуно дислоцирана у односу на све друге обрадне системе, је технолошки процес израде делова од бакра који је издвојен у односу на остале технолошке пступке. Оштрилица алата, М8, највише се користи за оштрење алата који су везани за машину М2, тако да је позиционирана баш уз њу.

Layout производно-монтажног погона представљен је лабораторијским моделом, слика 2.1.2, где је једина разлика у томе што позиција 18, односно сто за монтажу подсклопова у оквиру монтаже великих ормана није био постављен у модел окружења.



Слика 2.1.2: Лабораторијски модел окружења

2.2. Индустријски и мобилни роботи, сензори и мерни системи робота

2.2.1. Индустријски роботи

Већ је познато да реч "Робот" не датира из научних дела већ је своје корене нашла у научно фантастичној драми чешког писца Карела Чапека који је овај назив извео из чешке речи „*робота”* која се односи на присилан рад. Применом овог имена и лаик може наслутити које су његове функције. Али, тачне и детаљне одговоре на ову тему не можемо тражити у белетристици већ одговоре и тачне дефиниције налазимо у стручној литератури.

Наводе се две дефиниције индустријских робота [1]:

1. RIA[1] дефиниција:

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

2. ISO дефиниција:

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.

Иако не постоји савршена дефиниција индустријског робота уочавамо да фигуришу две речи: програмабилност и флексибилност, уз недостатак још две: интелигенција и аутономност о којима ће бити речи у оквиру подпоглавља о интелигентним технолошким системима. Програмабилност нам говори да се програмирана кретања и помоћне функције могу мењати без физичких интервенција, док флексибилност омогућава прилагођавање различитим применама са или без физичких интервенција.

Ако се сада осврнемо на разлог развоја оваквих система у исто време приближићемо се окосници флексибилне аутоматизације и порекло речи "робот". Наиме, принудни и тежак рад, тачније његово олакшавање и смањење ризика по човека, оно је што је инспирисало човека да тежи ка развоју ове области. И постепено стигло се 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, на пример узимање радног предмета са одређеног места и постављање на машину, као и послове који се обављају у нездравим условима (просторије са високим температурама, просторије контаминиране радиоактивним материјалом и слично). Уопште, то су послови од којих је човека пожељно ослободити, па тако роботи имају одређену улогу у хуманизацији рада. Илустративно се ови послови описују са три D и три H:

- **dull**-досадан монотон,
- **dirty**-прљав,
- **dangerous**-опасан.
- **hot**-врућe,
- **heavy**-тешко и
- **hazardous**-рискантно.

[1] RIA - **R**obotic **I**nstitute Of **A**merica

Ипак роботи су веома продуктивни, раде у више смена, раде уједначеним ритмом, праве мало шкарта те је у капиталистичком друштву које нас окружује главни мотив за њихову примену првенствено економски. Применом робота избегавају се појаве у производњи које се у аутомобилској индустрији наводе као "monday morning car".

Индустријски роботи се данас примењују у широком спектру области и функција од којих издвајамо:

- Манипулацију (опслуживање машина, палетизацију/депалетизацију),
- Обављање процеса (заваривање, обрада резањем, сечење ласером, воденим млазом),
- Манипулација и обављање процеса (монтажа) и
- Специјални задаци (мерење и контрола)

Индустријске роботе можемо поделити на различите начине према:

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

Према JARA[2] класификацији индустријски роботи се по типу управљања деле на следеће класе:

- Ручни манипулациони уређаји: то су уређаји са неколико степени слободе којима управља оператер,
- Секвенцијални роботи: манипулациони уређаји са фиксним и променљивим секвенцијалним управљањем,
- Понављајући роботи: оператор вођењем или са пулта, извршава задатак водећи робот који меморише трајекторије и касније их понавља,
- Нумерички управљани роботи: програмирају се текстуалним језицима и
- Интелигентни роботи: на бази информација са сензора и техникама вештачке интелигенције разумеју задатак и околину и доносе одлуке у реалном времену.

Роботи се даље деле и на генерације:

- Прва генерација - програмски роботи,
- Друга генерација - адаптивни роботи и
- Трећа генерација - интелигентни роботи.

[2] JARA - **Ja**pan **R**obot **A**ssociation

2.2.2. Мобилни роботи

Мобилни робот је манипулативан физички систем, који се креће кроз неструктурирани простор, остварујући притом интеракцију са људским бићима или обављајући неки посао уместо њих.

Једна од класификација мобилних робота може се извршити с обзиром на број додатних степени слободе и то на оне са:

- једним степеном слободе (врши линеарно кретање тзв. линеарна мобилност);
- два степена слободе (врше кретање у равни тзв. равански роботи);
- три степена слободе (врше кретање у простору тзв. просторна мобилност).

Мобилни роботи се могу класификовати и према уређајима помоћу којих остварују кретање и то:

- кретање помоћу ногу тзв. "*legged"* робота , који још могу бити хуманоидни и анималоидни;
- котрљање помоћу точкова, гусеница;
- кретање по шинама.

Следећа подела остварена је на основу окружења у којима се мобилни роботи крећу и извршавају своје задатке.

- роботи који се крећу по подлози;
- роботи који егзистирају у ваздуху;
- роботи који егзистирају у воденом окружењу.

Оправданост примене и коришћења мобилних робота:

- Већа расположивост робота, будући да немају прописано радно време,
- Манипулација већим теретима и бржи рад у односу на човека (резање, фарбање, брже кретање),
- Смањење употрбе скупих ресурса,
- Повећање квалитета призвода,
- Обављање послова који су за човека заморни, досадни...

Примена мобилних робота:

- Медицинске услуге (синтеза биолошког и механичког састава је киборг),
- Опасни послови и додир са струјом,
- Рударство,
- Грађевински послови,
- Пољопривреда,
- Подводни свет,
- Свемир,
- Војне сврхе,
- Превоз, складиштење, руковање материјалом

2.2.3. Сензори

Сензори су различити мерни уређаји и системи којима робот добија информације о себи и околини. Данас су то уређаји за мерење угаоног и транслаторног померања, различити сензори додира, уређаји за мерење растојања, силе, убрзања и сл.

Сензори у роботици омогућавају [1]:

- Мерење параметара у управљачким повратним спрегама,
- Налажење објеката (препознавање, позиција и оријентација објеката),
- Корекцију грешака у моделу робота и околине,
- Откривање и решавање проблема у погрешним ситуацијама,
- Мониторинг интеракције са околином,
- Осматрање промена у околини које могу утицати на задатак и
- Инспекцију резултата процеса.

Постоји више критеријума на основу којих се сензори класификацију у зависности од области у оквиру које се разматрају. Од посебног значаја је и подела сензора на:

- унутрашње, који контролишу унутрашње стање робота одређено његовим унутрашњим величинама;
- спољашње, који генеришу мерне сигнале о спољашњим величинама из окружења робота.

Унутрашњи сензори дају информације о позицији, брзини, убрзању, сили и моментима. Спољашње сензоре предствљају сензоре околине у које спадају тактилни сензори, сензори близине и сензори удаљености [1].

Такође, не смемо заборавити ни следећу поделу сензора која узима у обзир енергетски утицај окружења на систем и обрнуто. У том погледу разликујемо [2]:

- 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 ову групи спадају ласерски и ултразвучни сензори [2].

2.2.4. Мерни системи

Енкодери представљају једну широку класу дигиталних сензорских система који се користе за мерења линијских и угаоних помераја. Већ из самог назива се види да се ради о чисто дигиталним системима, који кодирају (енг. encoding) угаони или линијску позицију, коришћењем одговарајућих електромеханичких склопова [5].

Енкодери се класификују у две велике групе:

1. Апсолутни енкодери, и
2. Инкрементални енкодери (које смо ми користили)

Апсолутни енкодери мере апсолутну позицију, која је дефинисана конструктивним решењем склопа у оквиру кога функционишу.

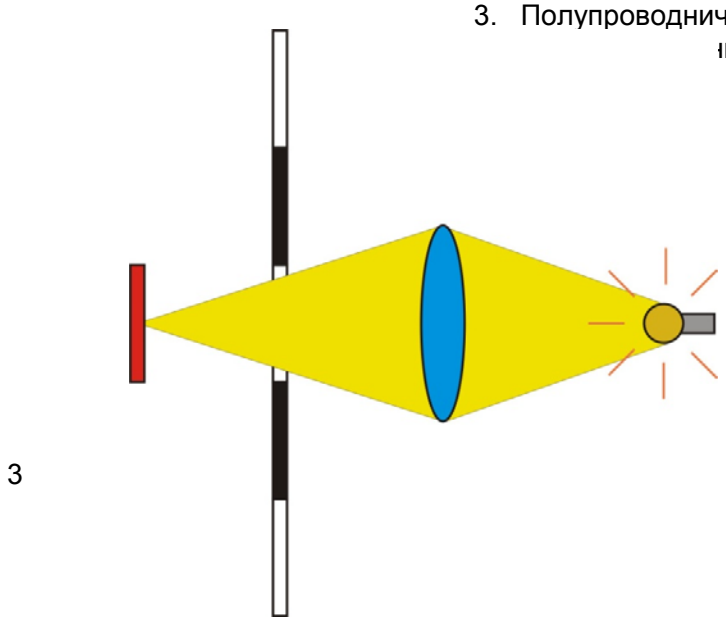
Инкрементални енкодери мере релативни положај у односу на неку унапред познату иницијалну координату (угаону или линијску).

Енкодери се најчешће изводе коришћењем оптичких и магнетних електромеханичких претварача. Доминантна технологија у савременој индустријској пракси је базирана на оптоелектронским претварачима.

Оптички енкодери поседују оптичке елементе, фото диоде и фото транзисторе, од којих је један извор, а други пријемник. Поред оптичких елемената, енкодер садржи и маску на којој се налазе прозирна и непрозирна поља. Ова поља наизменично прекидају оптички пут између извора и пријемника. Величина ових поља је у директној сразмери са механичком резолуцијом енкодера.

Легенда:

1. Извор светлости (полупроводничка диода или ласер)
2. Маска
3. Полупроводнички
ии елемент

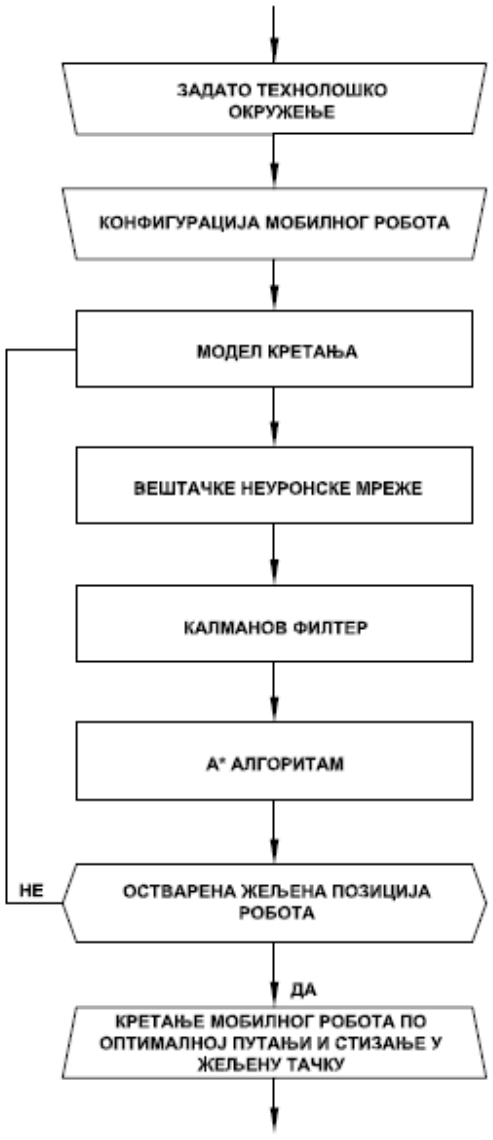


Слика 2.2.4.1: Принцип рада енкодера

3. Концепцијско решење пројектног задатка

Концепцијско решење пројектног задатка обухвата конфигурисање мобилног робота, модел кретања, вештачке неуронске мреже, Калманов филтер, A* алгоритам и симулацију рада технолошког система у софтверу *AnyLogic*. Детаљан опис ће бити приказан у следећем поглављу.

Решење овог пројектног задатка приказана је упрошћено помоћу блокова алгоритма датог на слици 3.1.



Слика 3.1: Алгоритам концепцијског решења пројектног задатка

4. Конфигурација мобилног робота

4.1. Опис расположивих компоненти LEGO MINDSTORMS NXT пакета за едукацију

За конфигурисање мобилног робота коришћењене су компоненте из *LEGO MINDSTORMS NXT* [6] пакета за едукацију. Свако паковање се састоји из сензора (за додир, звук, светлост и ултразвучно одређивање раздаљине), мотора, програмабилног микроконтролера и блокова за основну конструкцију.

Слика 4.1.1: Приказ улазно-излазних портова Lego Mindstorm NXT

Основне компоненте LEGO MINDSTORMS NXT пакета [7]:

1. Управљачка јединица
2. Сензор додира
3. Сензор звука
4. Сензор светлости
5. Ултразвучни сензор
6. Серво мотори

Управљачка јединица. Она је главни део задужен за снимање програма, корисничких датотека, комуникацију са рачунаром и управљање сензорима. Комуникација са рачунаром може се остварити преко *USB* прикључка или *Bluetooth*-а. Користећи програмску подршку која се добије у *NXT* пакету може се направити и уписати готов програм у меморију управљачке јединице (УЈ) па га покренути на самој УЈ. Други начин управљања је слање команде и очитавање једног по једног сензора и покретање актуатора, на пример програмирањем у C++ програмском језику или MATLAB софтверском пакету.

Сензор додира. Сензор додира (eng. touch sensor) је тастер који детектује притискање или отпуштање типке. Очитавање са овог сензора могуће је добити у bool или raw modu. У bool modu очитавање је 0 када типка није притиснута, а 1 када је притиснута. У raw моду, очитање је 1023 када типка није притиснута, а 183 када је притиснута.

Сензор звука. Сензор звука је микрофон који детектује јачину звука у децибелима (dB) или прилагођеним децибелима (dBA). При детекцији звука у прилагођеним децибелима, сензор са појачаном осетљивости мери звукове које људско ухо може чути. Ако се детектује у стандардним децибелима, сви звукови мери ће се са једнаком осетљивости. Највећа јачина звука која се може мерити износи 90 dB, што је отприлике бука косачице. Очитавања се на NXT-у приказују у процентима [%]. Што је нижи проценат, звук је тиши.

Сензор светла. Сензор светла може радити у два основна начина рада: активном и неактивном. У активном начину рада, пали се усмерени зрак светлости на самом сензору (LED), а пријемни сензор детектује количину рефлектоване светлости од неког објекта. Може се користити такође као сензор за удаљеност. У неактивном начину рада, извор светлости на самом сензору не ради, већ пријемни сензор детектује интензитет обојених површина амбијенталног осветљења. Очитавања су у процентима или raw modu.

Ултразвучни сензор. Овај сензор се користи за мерење удаљености од објеката тако да рачуна време које је потребно звучном таласу да удари у објекат и врати се, као бука. Може вредност иказивати у инчима или у милиметрима.

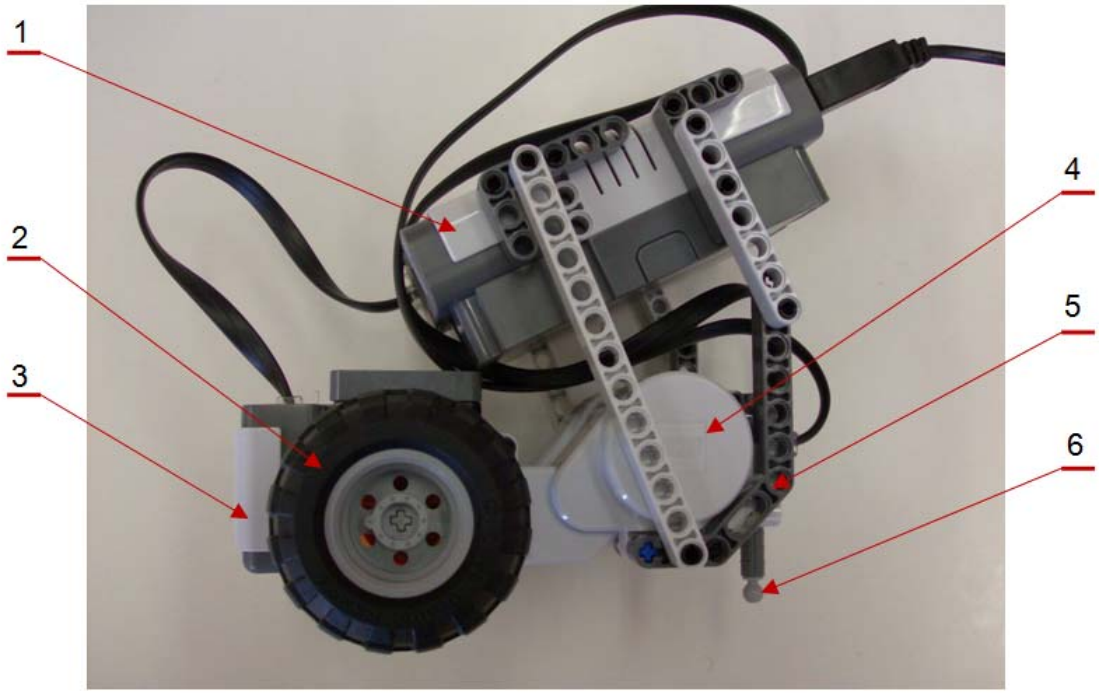
Серво мотори. Мотори служе за покретање робота. У сваком мотору уграђен је сензор уз помоћ којега је могуће добити угаону вредност обртања мотора, одређени смер обртања, подешену брзину обртања и жељени угао закрета те их гасити на начин да се мотори заблокирају или само угасе.

4.2. Формирање конфигурације мобилног робота

У овом пројекту *Lego Mindstorms NXT* робот мора да има процесор, који обрађује информације, сензоре који скупљају информације из околине у којој се налази робот, делове помоћу којих се врши кретање робота и извор енергије који неопходан за напајање компоненти робота.

За формирање конфигурације мобилног робота коришћени су следећи елементи:

1. Управљачка јединица са интерфејсима за моторе;
2. Погонски точак (два точка);
3. Сензор (светлосни - *light sensor*);
4. Погонски модул (чине га два мотора);
5. Носећа конструкција;
6. Куглица за ослањање мобилног робота



Слика 4.2.1: Конфигурација мобилног робота

Приликом конфигурисања мобилног робота, заједно са функционалним захтевима пројекта, усвојено је коришћење оптичког сензора, као и интегрисаних енкодера у сервомоторима. Налажење најбоље локације за светлосни сензор било је од изузетног значаја за пројектовање целокупне конфигурације мобилног робота. Овај сензор је било потребно сместити на локацију где би првенствено могао да обавља своју функцију - детекцију црних тачака (маркера) у оквиру макете технолошког окружења. Такође, требало је водити рачуна о „количини шума“, у овом случају спољашње светлости, која би отежавала, или пак онемогућавала вршење мерења.

Пројектована конфигурација мобилног робота поседује неколико мана, које могу исправити евентуалним унапређењем конструкције у будућности, уколико се за то буде појавила прилика.

Носећа конструкција је састављена тако да буде довољно крута са што мањом масом и прилагођена самој радњи робота. Носећу конструкцију носе два погонска точка која су постављена на погонски модул и куглица за ослањање робота која клизи по равној и глаткој површини. На самој конструкцији је постављена управљачка јединица (процесор *NXT*) и сензор.

Погонски модул се састоји из два серво мотора који поседују енкодере помоћу којих се мери пређени пут. Они су постављени са обе стране, где су точкови. Мотори се управљају помоћу PWM - Pulse Width Modulation, тј. принципа ширинске модулације импулса, што представља начин управљања савременим серво системима. Мотори дају контролној јединици сигнале повратне спреге помоћу 12bit-ог инкременаталног енкодера. Диск енкодера је назубљен како би био у спрези са вратилом мотора. Мотор има шестостепени зупчасти пренос од вратила мотора до осовине преко које се остварује кретање точка робота.

Сервомотор се састоји из погона – мотора, осам зупчаника, ротационог сензора и главе сервомотора (слика 4.2.2.)

Слика 4.2.2: Конструкција сервомотора

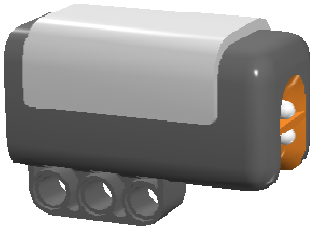
Због преносног односа зупчаника, ротациони сензор може да детектује 1° ротације главе сервомотора. Детаљан изглед сензора ротације је дат на слици 4.2.3.

Слика 4.2.3: Сензор ротације сервомотора

Технички параметри мотора:

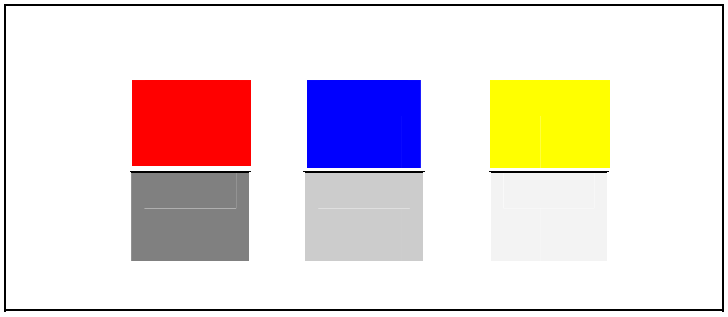
- Напајање 9V
- Угаона брзина 177 min-1
- Обртни момент 16.7 Ncm
- Снага 2.03 W
- Ефикасност 41%

Сензор (светлосни - light sesor) има основну функцију да омогући да се направи разлика између светла и таме, да мери интензитет светлости у просторији и да се мери интензитет светла на некој обојеној површини.



Слика 4.2.4: Изглед сензора за детекцију светлости

Сензор не препознаје саме боје, већ искључиво препознаје интензитет рефлексије светлости неке боје. То у преводу значи да су све боје у ствари само нијансе сиве, односно приближно црна или приближно бела боја, као што је приказано на слици 4.2.5.



Слика 4.2.5: Препознавање боје у зависности од очитавања са оптичког сензора

Моторе са енкодерима и светлосни сензор повезује, напаја и њима управља LEGO Mindstorms управљачка јединица слика 4.2.6 преко које је могуће остварити комуникацију са рачунаром (USB или Bluetooth порт).

Слика 4.2.6: Управљачка јединица LEGO Mindstorms

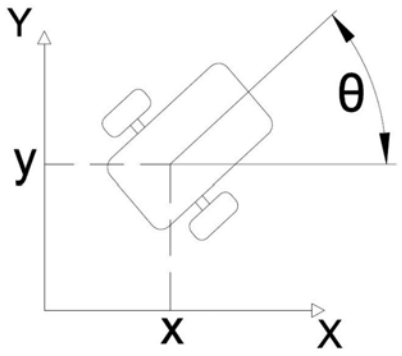
5. Модел кретања мобилног робота

Задатак који треба да изврши програмирани мобилни робот је да се креће по тачно одређеној путањи и по унапред познатом моделу кретања. Одабрани модел кретања мора да задовољава тражене услове тачности кретања мобилног робота како не би дошло до колизије мобилног робота са окружењем, односно са другим машинама које се налазе у његовом окружењу. Модел кретања може бити приказан математичком формулацијом, односно:

$$X_t = (x, y, z, \theta, \psi, \varphi)^T \tag{5.1}$$

Из наведене једначине видимо да је вектор стања мобилног робота описан одређеним променљивама. Те променљиве дефинишу положај и оријентацију моблиног робота. Ово је општи приказ вектора стања мобилног робота и представља његово стање када се налази у простору. Постоји и равански случај, који је једноставнији од горе наведеног и и њему егзистирају само променљиве $x, y, \theta.$, вектор стања је приказан једначином:

$$X_t = (x, y, \theta)^T \tag{5.2}$$



Слика 5.1: Мобилни робот у (x, y) равни

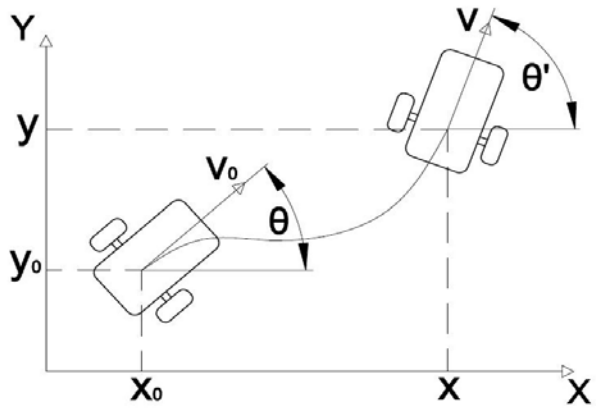
Променљиве које су коришћене у претходним једначинама су:

- x – компонента вектора дуж осе X
- y – компонента вектора дуж осе Y
- z – компонента вектора дуж осе Z
- θ – угао ротације око осе X
- φ – угао ротације око осе Y
- ψ – угао ротације око осе Z

Модел кретања којим се врши локализација мобилног робота може бити :

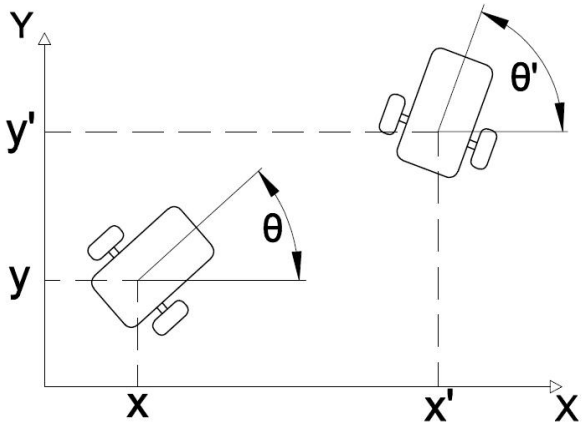
- ***Модел кретања на основу брзина (Брзински модел кретања) тј. "Velocity based motion model"***
- ***Модел кретања на основу пређеног пута (Одометрија) тј. "Odomety"***

Брзински модел кретања се заснива на претпоставци да се кретање моблиним роботом, односно управљање угаоном и транслаторном брзином може директно управљати. У овом моделу транслаторна и угаона брзина кретања мобилног робота представљају управљање x(t) које одређују промену позиције и оријентације током експлоатације. Овај модел није коришћен у задаку због тога што је при експлоатацији овог решења неопходно увести сензоре за очитавање брзине точкова. На следећој слици је приказ брзински модел кретања мобилног робота у равни:



Слика 5.2: Брзински модел кретања мобилног робота у равни

Модел кретања на основу пређеног пута се заснива на мерењу помоћу енкодера од неког положаја у коме је мобилни робот био у тренутку t-1 до неког положаја у којем се робот тренутно налази у тренутку t. Робот погоне два точка од којих сваки има посебан мотор за погон, тј. точкови се обрћу независно један од другог. Овим се закључује да се управљање смером обртања точкова контролише скретање мобилног робота. На осовинама точкова се налазе енкодери који мере углове ротације (обртање точкова). Очитане вредности са енкодера се уводе у једначине, и њиховим решавањем добијамо информацију о позицији робота након оствареног кретања. Пређени пут сваког точка се означава Δ_{Sd} за десни точак, и Δ_{Sl} за леви точак.



Слика 5.3: Промена позиције и оријентације два узастопна положаја мобилног робота

Вектор положаја се може описати са наведеним једначинама у којима важи зависност :

$$x' = \{x\ y\ \theta\}^T \tag{5.3}$$

$$x' = f(x, y, \theta, \Delta s_{\mathrm{d}}, \Delta s_{\mathrm{l}}) \tag{5.4}$$

Остале променљиве које се појављују приликом коришћења овоог модела су величине које одређују прираштај, тј.:

$\Delta x = \Delta s \cdot cos(\theta + \frac{\Delta\theta}{2})$ - промена вредности за осу x (5.5)

$\Delta y = \Delta s \cdot sin(\theta + \frac{\Delta\theta}{2})$ - промена вредности за осу y (5.6)

$\Delta\theta = \frac{\Delta s_{\mathrm{d}} - \Delta s_{\mathrm{l}}}{b}$ - промена угла ротације (5.7)

$\Delta s = \frac{\Delta s_{\mathrm{d}} + \Delta s_{\mathrm{l}}}{2}$ - промена лучне координате (5.8)

Δs_{d}, Δs_{l} - пређени пут десног и левог погонског точка респективно

б - растојање између точкова.

Тренутна позиција и оријентација мобилног робота се рачуна на основу следећег израза:

$$x' = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \Delta s \cdot cos(\theta + \frac{\Delta\theta}{2}) \\ \Delta s \cdot sin(\theta + \frac{\Delta\theta}{2}) \\ \Delta\theta \end{bmatrix} \tag{5.9}$$

Односно када се изврши смена :

$$x' = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{\Delta s_{\mathrm{d}} + \Delta s_{\mathrm{l}}}{2} \cdot cos(\theta + \frac{\Delta s_{\mathrm{d}} - \Delta s_{\mathrm{l}}}{2b}) \\ \frac{\Delta s_{\mathrm{d}} + \Delta s_{\mathrm{l}}}{2} \cdot sin(\theta + \frac{\Delta s_{\mathrm{d}} - \Delta s_{\mathrm{l}}}{2b}) \\ \frac{\Delta s_{\mathrm{d}} - \Delta s_{\mathrm{l}}}{2b} \end{bmatrix} \tag{5.10}$$

$$x' = x + \frac{\Delta s_{\mathrm{d}} + \Delta s_{\mathrm{l}}}{2} \cdot cos(\theta + \frac{\Delta s_{\mathrm{d}} - \Delta s_{\mathrm{l}}}{2b}) \tag{5.11}$$

$$y' = y + \frac{\Delta s_{\mathrm{d}} + \Delta s_{\mathrm{l}}}{2} \cdot sin(\theta + \frac{\Delta s_{\mathrm{d}} - \Delta s_{\mathrm{l}}}{2b}) \tag{5.12}$$

$$\theta' = \theta + \frac{\Delta s_{\mathrm{d}} - \Delta s_{\mathrm{l}}}{2b} \tag{5.13}$$

6. Систем вештачких неуронских мрежа

Да бисмо обавили основни задатак пројекта, конфигурисали интелигентни мобилни робот, који ће заменити класичне виљушкаре, и направили уштеду у транспорту у производном погону, морамо пројектовати сензоре и управљање мобилног робота да би се сналазио у производном погону без интервенције човека. Ови сензори ће на директан или индеректан начин имати комуникацију са окружењем. Комуникација са окружењем се своди на препознавање реалног простора и ситуације, тј. у овом случају позиције и оријентације мобилног робота у односу на затечене услове у радном простору. Слањем правих информација моторима постижесе управљање мобилног робота који као и сваки склоп или машина има одступања која се претварају у грешке због немогућности да се идеализује систем. Уместо идеализовања компоненти, чија производња тражи велика финансиска средства, у мобилни робот су уграђени мотори који на свом вратилу имају енкодере, тако да уз корекцију имамо тачну позицију и оријентацију. У склопу мобилног робота који смо конфигурисали, уградили смо и сензор који мери интензитет рефлектоване светлости о подлогу, ради препознавања референтних тачака, које нам служе за нулирање грешке која се слаже у току кретања и поред употребе енкодера на вратилу мотора.

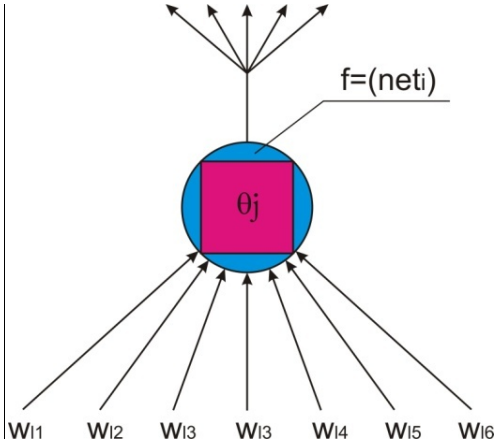
Као парадигму вештачке интелигенције коришћене су вештачке неуронске мреже које служе за обучавање система који је несавршен (и поред грешки које прави никад не зна где се тачно налази). Грешке које настају су пореклом и од самог склопа мобилног робота као и од утицаја окружења у којем мобилни робот ради, а то су грешке услед варијације у напону струје у електричном мотору, зазора у компонентама, деформације под којим је мобилни робот оптерећен имају негативан утицај на претходна два утицаја које вишеструко повећева. Поред грешки за остваривање и мерење кретања постоје и неповољни утицаји и на сензор за мерење интензитета одбијене светлости, а то су различита осветљеност у погону од несавршености вештачког осветљења до утицаја у дневној светлости, материјали који немају исти коефицијент рефлекције светлости, што због саме врсте материјала и храпавости истог.

6.1. Основе вештачких неуронских мрежа

Према дефиницији из [4]: Вештачка неуронска мрежа је парадигма вештачке интелигенције која се дефинише као конективни модел за резоновање заснован на аналогији са мозгом, уз наглашену когнитивну способност да учи и врши генерализацију стеченог знања.

Вештачке неуронске мреже имају способност да препознају смисао у компликованим или непотпуним подацима, да препознају обрасце који су непрепознатљиви људима због своје сложености и који се не могу добити коришћењем других компјутерских техника. Вештачке неуронске мреже су софистициране технике моделирања, способне да моделирају веома комплексне функције.

Неуронске мреже су добре у обављању задатака када су подаци хетерогени, несређени, несигурни, па чак и неконзистентни и када не постоје савршена решења за практичне проблеме.



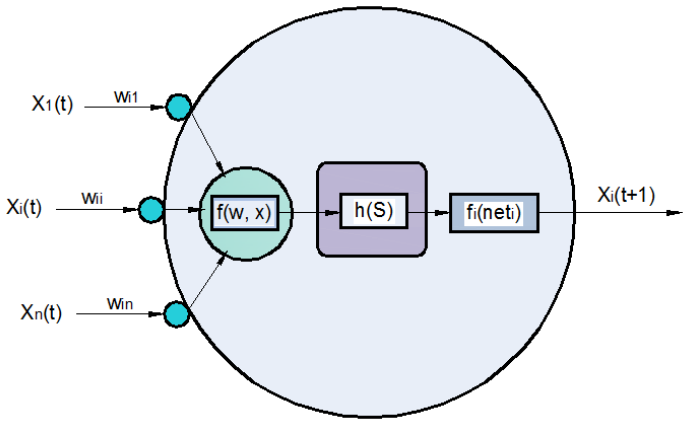
Слика 6.1.1: Неурон са својим окружењем

Основи елементи неурона су [3] (слика 6.1.2):

- **Улазни оператор** *f(w,x)* обезбеђује улазе и тежинске односе међусбних веза. Такође спрема улаз за фукцију преноса која се може представити као $s=f(w,x)=w^Tx$
- **Фукција преноса** *h(s)* обрађује излаз из улзаног оператора, а затим врши интеграцију и тако формира улаз за активациону функцију.
- **Активациона функција** $f_i=(net_i)$ обрађује излаз функције преноса и обезбеђује излазну вредност неурона

Активационе функције су најважније за правилно функционисање неурона. За различите моделе вештачких неуроских мрежа се користе и различите активационе функције. Неке од активационих функција се могу сврстати у неколико група [3]:

- Линеарне
- Бинарне
- Сигмоидне
- Компетитивне
- Гаусове.

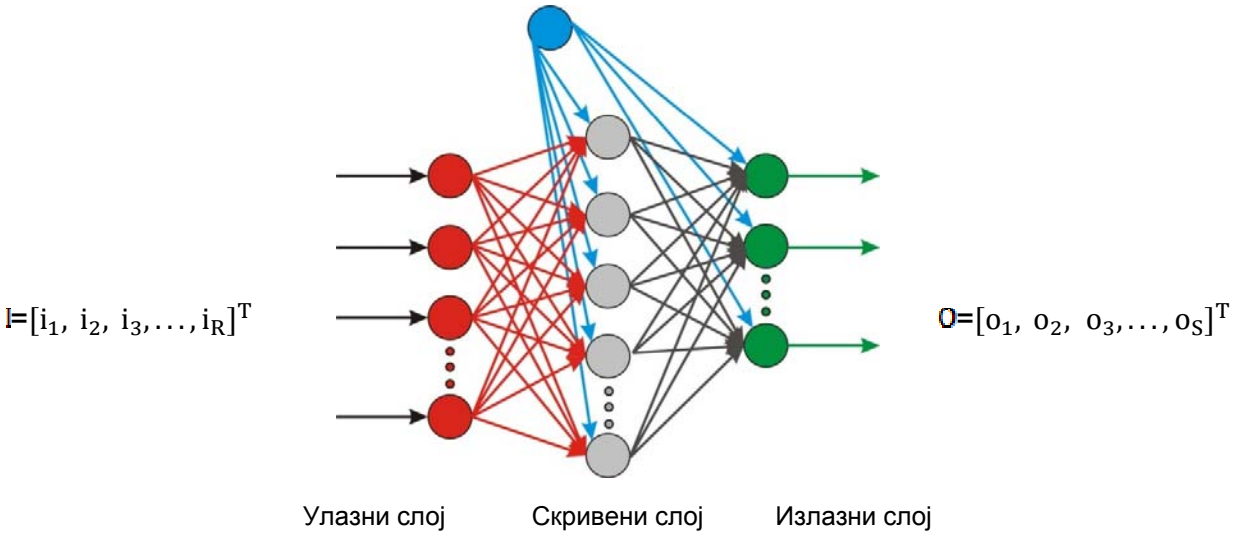


Слика 6.1.2: Приказ типичног процесуирајућег елемента – неурона

Најраспрострањенији развијени модели вештачких неуронских мрежа су [3]:

- Perceptron
- **Backpropagation (BP) неуронска мрежа**
- Асоцијативне неуронске мреже
- Hopfield-ове неуронске мреже
- ART неуронске мреже
- Fuzzy асоцијативне неуронске мреже
- Самоорганизујуће неуронске мреже.

Основни облик архитектуре BP неуронске мреже има један улазни, скривени и излазни слој, док број неурона варира од проблема који се решава применом ове вештечке неуронске мреже, слика 6.1.3. Исто важи и за број скивених слојева. Најчешће постоји само један скривени слој, јер је BP мрежа са таквом структуром у стању да обезбеди репродуковање скупа захтеваних излазних облика за све обучавајуће парове, али не тако ретко се сусреће и архиктектура са више скивених слојева.



Слика 6.1.3: Основни облик архитектуре BP неуронске мреже

Неуронске мреже су у стању да пронађу везе између појава које измичу људском интелектуалном апарату потпомогнутом класичним софтверским алатима.

Неуронске мреже имају и могућност *толеранције недостатака* – мрежа се састоји од више елемената процесирања, па може да функционише и ако дође до оштећења дела мреже.

Способне су да генерализују, па ако им се презентује некомплетан скуп улазних података, мрежа ће ипак бити у стању да да излаз.

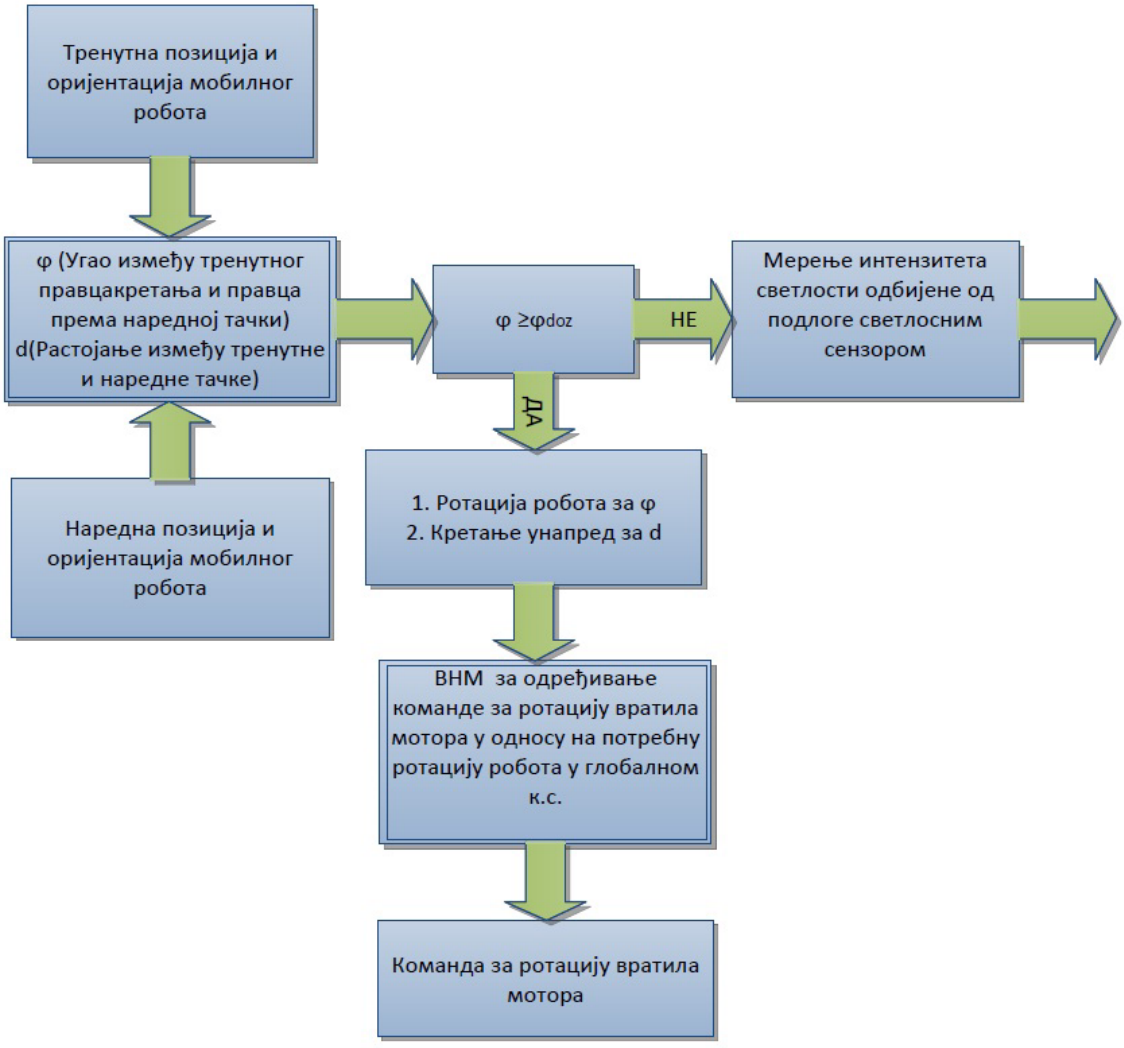
Иако имају одличну моћ предвиђања, имају слабу способност објашњавања. На слици 6.1.4. је приказан дијаграм зависности моћи предвиђања и објашњавања. Видимо да неуронске

мреже одлично предвиђају, а слабо објашњавају, потпуно супротно од нпр. стабала (дрвета) одлучивања. Неуронска мрежа не може кориснику да објасни како је дошла до одређеног решења.

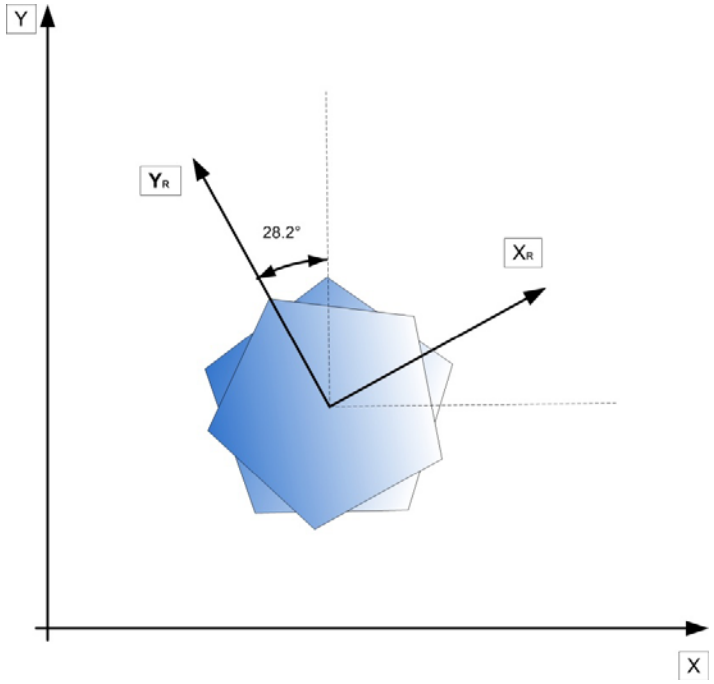
Слика 6.1.4: Дијаграм зависности моћи предвиђања и објашњавања

6.2. Вештачка неуронска мрежа за одређивање команде ротације вратила мотора у односу на потребну ротацију робота у глобалном координатном систему

Прва фаза је формирање скупа обучавајућих парова. Овај скуп је формиран у експерименталном поступку задавања команди за кретање робота, у циљу одређивања вредности које је потребно задати како би се робот ротирао за жељени угао.



Слика 6.2.1: Ток информација кроз развијени неуронски модел за одређивање команде за ротацију вратила мотора, у реалном времену



Слика 6.2.2: Пример ротације робота у месту за 28,2°

Табела 3: Скуп обучавајућих парова ВНМ за ротацију робота током кретања у технолошком окружењу												
Редни број обучавајућег пара	1	2	3	4	5	6	7	8	9	10	11	12
Улаз (угао ротације робота)	15	-15	30	-30	45	-45	60	-60	75	-75	90	-90
Излаз (команда за ротацију мотора)	15	15	50	50	90	82	120	105	150	115	178	135
Редни број обучавајућег пара	13	14	15	16	17	18	19	20	21	22	23	24
Улаз (угао ротације робота)	105	-105	120	-120	135	-135	150	-150	165	-165	180	-180
Излаз (команда за ротацију мотора)	200	150	240	-175	270	190	300	225	330	247	351	270

Прегледом вредности у табели може се уочити да пораст задате вредности за ротацију вратила мотора није пропорционалан порасту угла. Применом вештачких неуронских мрежа ове нелинеарности би требало да буду решене.

```
% odredjivanje AngleLimit i TurnRatio na osnovu NN modela

clc,clear
close all
% SetPower = const. = 30 => uvek!
% TurnRatio = 100 => pozitivan mat. smer = -100
%                    negativan mat. smer = 100

% Merenja su uradjena za sledece uglove:
% +/- [15 -15 30 -30 45 -45 60 -60 90 -90 120 -120 150 -150 180 -180]


AL =    [15 50 90 120 150 178 200 240 270 300 330 351 270 247 225 190 175 150 135
115 105 82 50 15]
input = [15 30 45 60 75 90 105 120 135 150 165 180 -180 -165 -150 -135 -120 -105 -
90 -75 -60 -45 -30 -15]

%
% %

output = [AL]

SCnet_PR = newff(input,output,[10],{'tansig'},'trainlm','learngdm')

SCnet_PR.trainParam.show = 50;
SCnet_PR.trainParam.lr = 0.05;
SCnet_PR.trainParam.mc = 0.7;
SCnet_PR.trainParam.mu = 0.001
SCnet_PR.trainParam.epochs = 1000;
SCnet_PR.trainParam.goal = 1e-5;
SCnet_PR.trainParam.max_fail = 10;

SCnet_PR = train(SCnet_PR,input,output)
y = sim(SCnet_PR,input)

disp('y AL input ')
mreza_vs_AL=[y' AL' input' ]
mrezatocak5=SCnet_PR
save mrezatocak5



figure(1),
plot(input,AL,'or'),hold on
plot(input,y,'b*')
% SCnet_PR2 = SCnet_PR
% save('SCnet_PR2','SCnet_PR2')
%
% subplot(2,1,1),plot(input,TR,'or'),hold on
% plot(input,y2,'b*')
% subplot(2,1,2),plot(input,AL,'or'),hold on
% plot(input,y1,'b*')
```

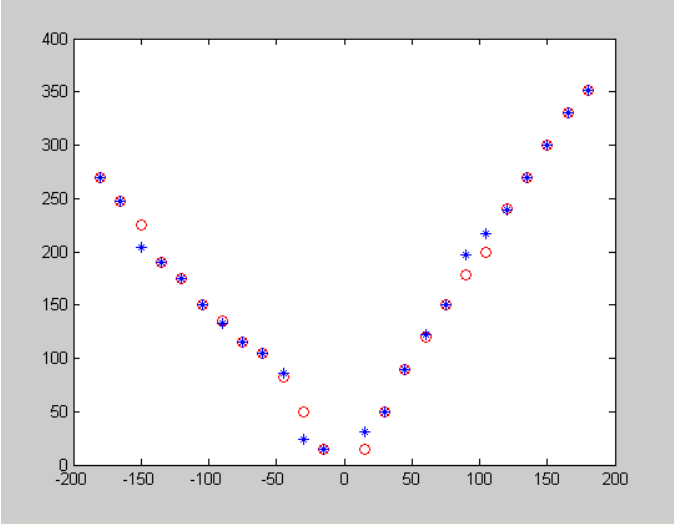
Табела 4: План експеримената обучавања ВНМ за ротацију робота током кретања у технолошком окружењу					
Број скривених слојева	Број неурона у првом скривеном слоју	Број неурона у другом скривеном слоју	Број неурона у трећем скривеном слоју	Параметар учења	Редни број експеримента
1	10	-	-	0.05	1
				0.3	2
				0.7	3
1	14	-	-	0.05	4
				0.3	5
				0.7	6
2	12	16	-	0.05	7
				0.3	8
				0.7	9
2	16	10	-	0.05	10
				0.3	11
				0.7	12
3	3	2	3	0.05	13

Поред величина које се у овом случају варирају, у оквиру *Neural Network Toolbox*-а *MATLAB*-а, дефинишу се још неки параметри који се овом приликом нису мењали. Тако се приликом дефинисања величине ВНМ, дефинишу још активационе функције за сваки од скривених слојева. У овом случају то је сигмоидна активациона функција (*tansig*). Затим се још дефинише примена Левенберг-Маркеоовог алгоритма учења (*trainlm*), као и поступак минималног градијента (*learngdm*).

Такође, *MATLAB*-ов *Neural Network Toolbox* омогућује дефинисање одређеног скупа параметара који се односе на сам процес обучавања и овом приликом нису варирани. То су следећи параметри:

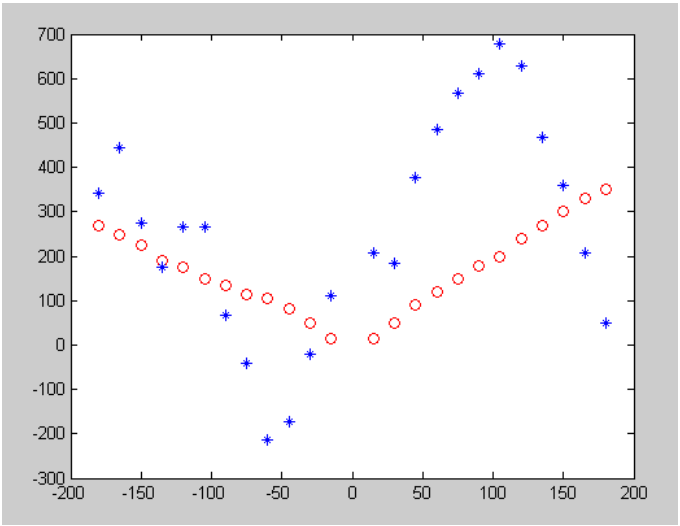
***show* = 50** - број итерација након ког се приказује промена.
***lr* = 0.1 и *mc* = 0.9** - константе момента ВНМ. Ови параметри утичу на инертност учења
***epochs* = 1000** - број итерација након кога се процес зауставља без обзира на испуњеност услова обучености мреже
goal=10^(-5) - захтевана вредност грешке између излаза који је остварила мрежа и захтеваног излаза
***max_fail* = 10** - број дозвољених падова на тесту приликом валидације
trainlm - све мреже користе Левенберг – Маркеов алгоритам обучавања

Као резултат процеса обучавања вештачких неуронских мрежа за дефинисан скуп обучавајућих парова, тест улазе и дефинисане параметре обучавања, пратећи установљени план експеримента, поред резултата који карактеришу успешност обучавања и које на крају тог процеса даје *Neural Network Toolbox*, даје се и упоредни графички приказ. Тај графички приказ се формира исцртавањем скупа обучавајућих парова (црвени кругови), резултата симулације ВНМ за улаз из обучавајућег скупа (плаве звезде) и резултата симулације ВНМ за скуп тест улаза.



Слика 6.2.3: Пример графичког приказа симулације ВНМ која даје добре резултате (ВНМ под редним бројем 11)

Са слике се уочава, за мрежу која даје добре резултате, да је преклапање вредности из обучавајућег скупа (црвени кругови) са вредностима које даје ВНМ након симулације (плаве звезде) у великој мери остварена. За разлику од добрих резултата, за мрежу која даје лоше резултате уочава се да је преклапање поменутих вредности мање.



Слика 6.2.4 : Пример графичког приказа симулације ВНМ која даје лоше резултате (ВНМ под редним бројем 6)

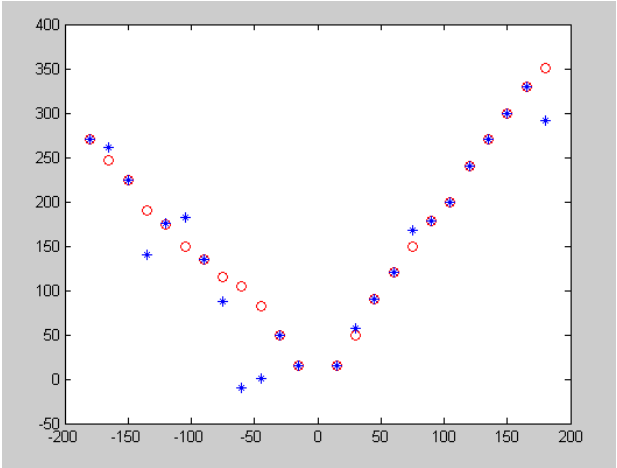
Приликом упоређивања графичког приказа резултата симулације за сваку од обучаваних неуронских мрежа, визуелно је вршено утврђивање степена поклапања резултата симулације ВНМ за улаз из обучавајућег скупа (плаве звезде) са вредностима из обучавајућег

скупа (црвени кругови). Визуелни увид, у оно што мрежа може да оствари након обучавања, употпуњују резултати симулације ВНМ за улаз из тест скупа. На овај начин су одабране пет ВНМ у ужи избор за имплементацију, а то су ВНМ под редним бројевима експеримента: 4, 7, 8, 9 и 12.

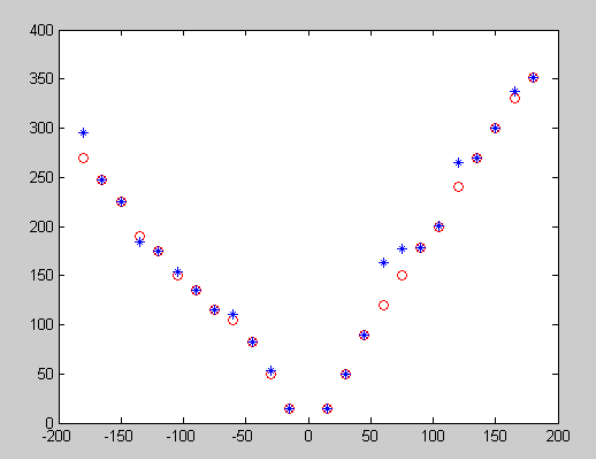
Приказ резултата које даје *Neural Network Toolbox* дат је у табели 5, за све обучаване ВНМ према плану експеримента.

Табела 5: Резултати обучавања ВНМ-а за ротацију робота током кретања у технолошком окружењу					
Редни број експеримента	Број итерација	Завршен процес минимизације грешке	Остварена минимална грешка	Остварена вредност градијента	Број негативних валидација мреже
1	7	Не	70.6	0.000208	6
2	7	Не	6.4	0.00355	4
3	8	Не	510	0.000755	7
4	**5**	**Да**	$\mathbf{8.75 \cdot 10^{-5}}$	**0.000208**	**1**
5	4	Не	80.5	0.213	2
6	4	Не	$5.12 \cdot 10^{4}$	0.0019	4
7	**5**	**Да**	$\mathbf{3.53 \cdot 10^{-13}}$	**0.000252**	**0**
8	**7**	**Да**	$\mathbf{1.27 \cdot 10^{-13}}$	**0.000278**	**0**
9	**4**	**Да**	$\mathbf{3.83 \cdot 10^{-10}}$	**0.00475**	**0**
10	5	Не	3.1	0.000654	2
11	6	Да	0.000134	0.00253	1
12	**5**	**Да**	$\mathbf{6.38 \cdot 10^{-12}}$	**0.00115**	**0**
13	110	Да	4.54	18.1	10

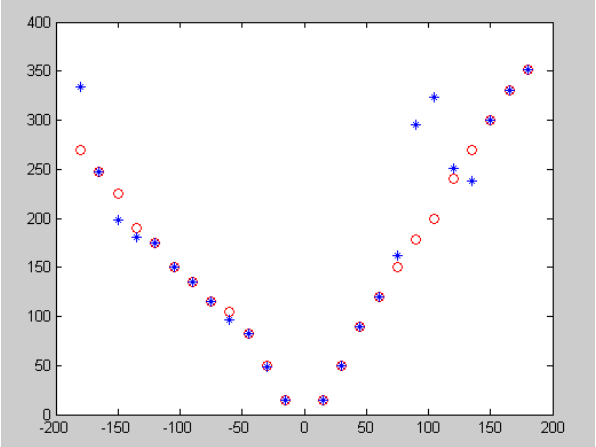
Коначна одлука о избору ВНМ за уградњу у управљачки систем донесена је на основу анализе резлутата датих у табели, на основу визуелне анализе графичких приказа резултат на сликама:



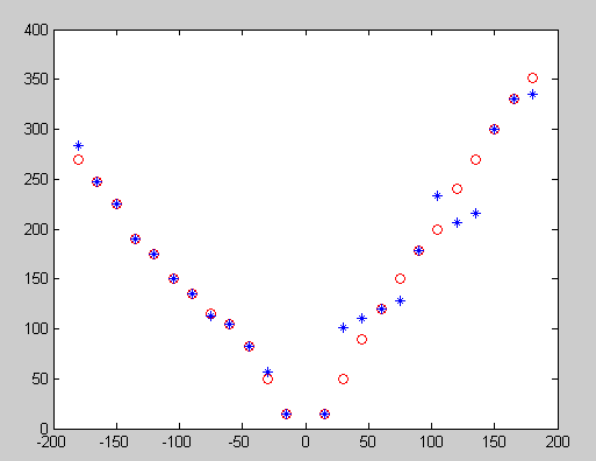
Слика 6.2.5 : Редни број експеримента 4.



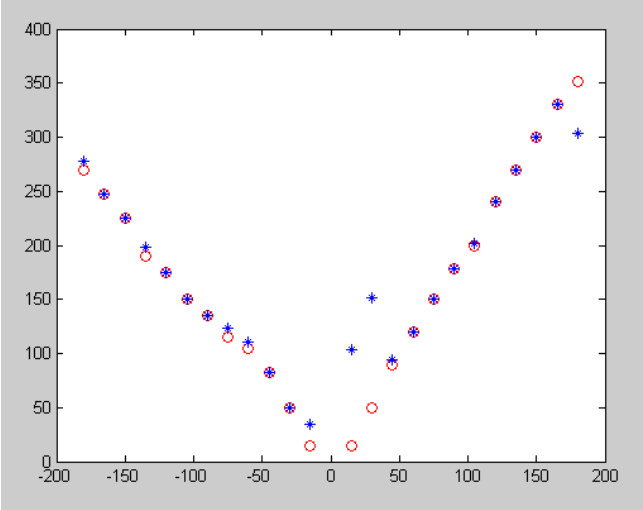
Слика 6.2.6: Редни број експеримента 7.



Слика 6.2.7: Редни број експеримента 8.

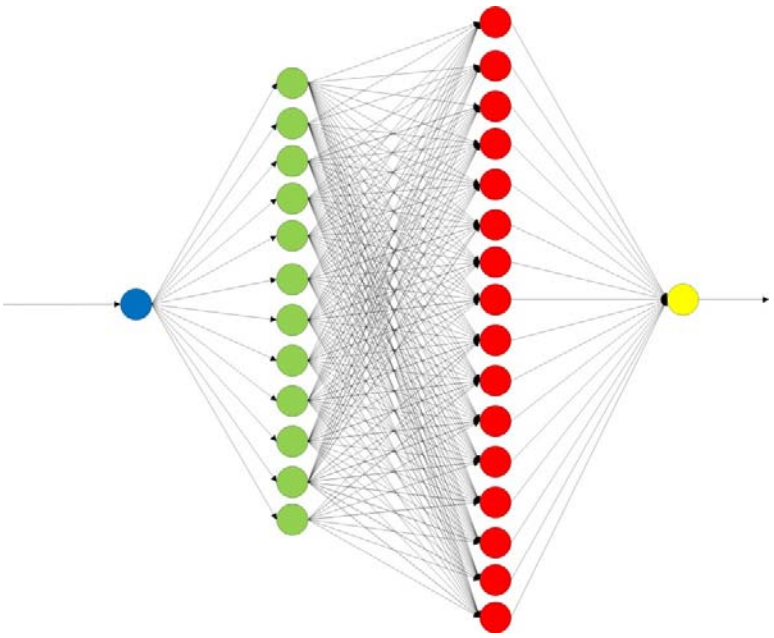


Слика 6.2.8: Редни број експеримента 9.

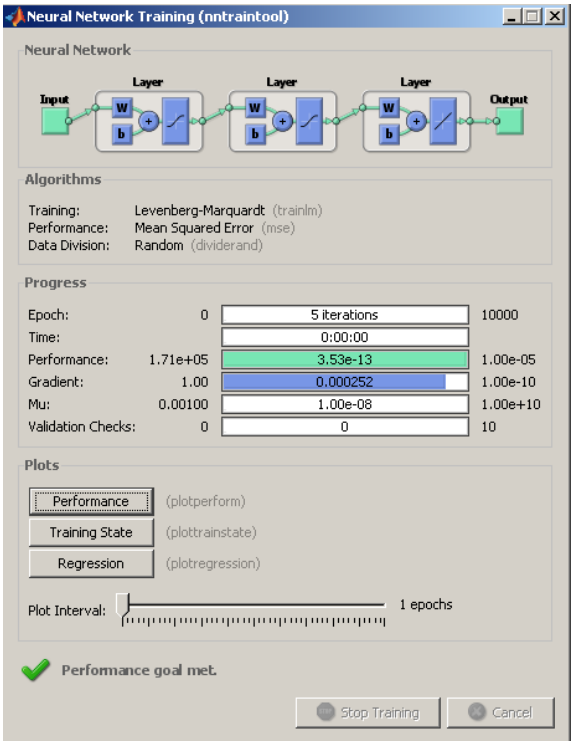


Слика 6.2.9: Редни број експеримента 13.

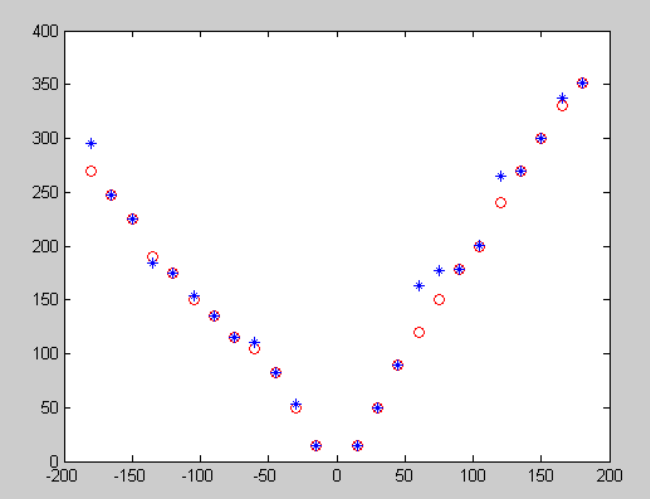
Изабрана ВНМ је под редним бројем 7. Из табеле се може уочити да је то вештачка неуронска мрежа са два скривена слоја, са 12 у првом и 16 неурона у другом скривеном слоју, као и да је параметар учења η=0.05. Како у општем случају број неурона у улазном и излазном слоју неуронске мреже зависи од димензија улазног, односно излазног вектора, тако да се у овом случају налази по један неурон у улазном и излазном слоју. Разлог је у томе што је неуронска мрежа развијена тако да, без обзира на то колики је улазни скуп, врши симулацију за по један улаз и као резултат даје по један излаз.



Слика 6.2.10: *Из*абрана мрежа за одређивање команде за ротацију вратила мотора у односу на потребну ротацију робота у глобалном координатном систему



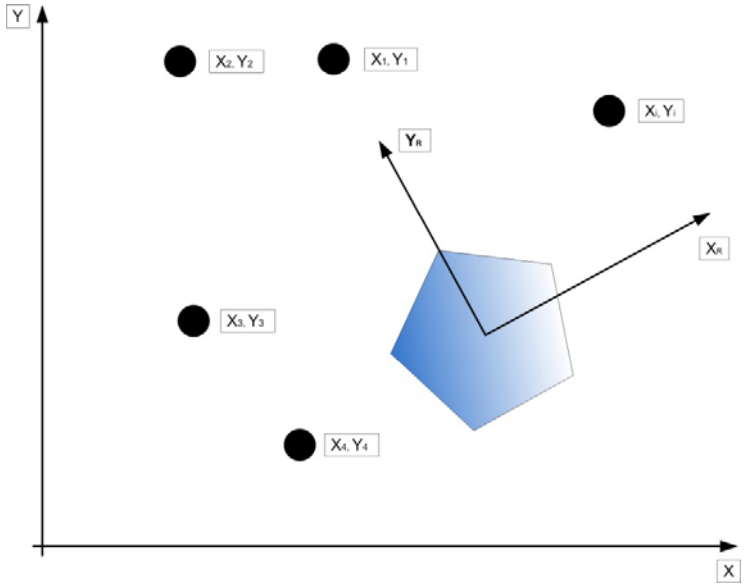
Слика 6.2.11: Приказ резултата обучавања из *Neural Network Toolbox*-а изабране ВНМ за одређивање команде за ротацију вратила мотора у односу на потребну ротацију робота у глобалном координатном систему



Слика 6.2.12: Пример графичког приказа резултата симулације ВНМ која изабрана (ВНМ под редним бројем 7)

6.3. Вештачка неуронска мрежа за препознавање контролних тачака у радном окружењу мобилног робота

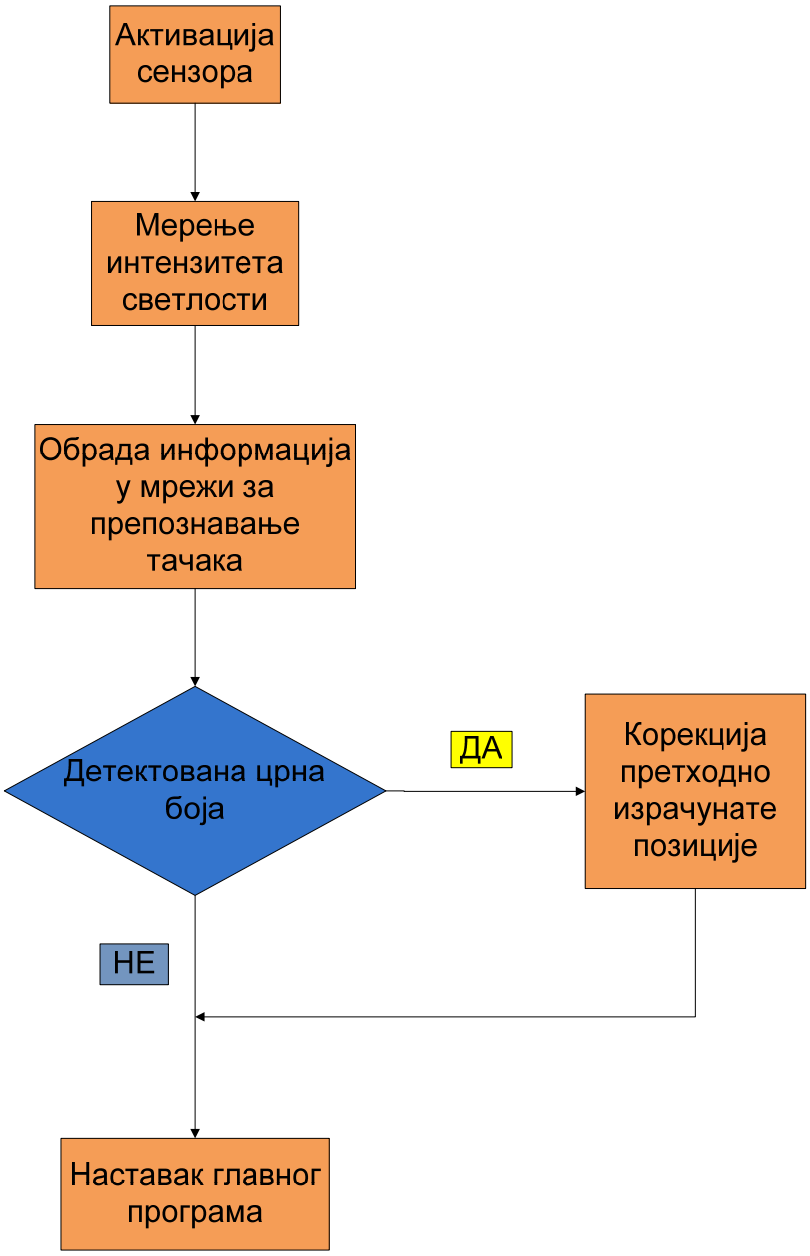
Конкретан задатак односи се на препознавање контролних тачака у радном окружењу које су црне, за разлику од окружења које је беле боје. За сваку од контролних тачака познате су њене координате у глобалном координатном систему .



Слика 6.3.1: Пример произвољне диспозиције контролних тачака у радном окружењу

Када се на основу мерења које врши сензор, а затим ВНМ препозна, установи да се робот налази изнад контролне тачке, потребно је ту информацију проследити управљачком систему.

На слици 6.3.2, приказан је ток информација кроз модел за препознавање контролних тачака у окружењу робота.



Слика 6.3.2 : Ток информација кроз модел за препознавање контролних тачака у окружењу робота

Проблем који ова ВНМ треба да реши, након уградње у управљачки систем, јавља се при узорковању јачине светлости одбијене од подлогу, односно при одређивању боје подлоге применом светлосног сензора. Услед нагле промене осветљења и појаве сенки, мерење које сензор може да оствари не даје јасно разграничење између резултата који карактеришу подлогу беле и црне боје. Може се закључити да сензор не поседује потребан ниво робусности, тако да се применом ВНМ жели постићи мултиплицирање његових

могућности, тј. да се и при граничним (критичним) мерењима одреди поуздана информација за даље акције.

Интензитет светлости одбијене од подлогу, као резултат мерења светлосног сензора, се добија у виду 1024 *bit*-ног податка. Овај податак узима вредност 0 уколико нема светлости одбијене од подлогу, односно вредност 1023 уколико је интензитет светлости одбијене од подлогу максималан који сензор може да измери.

Потребно је формирати такав обучавајући скуп да га карактеришу све сметње које су претходно наведене. Такав обучавајући скуп формиран је експериментално, уз разне промене осветљења у околини сензора, односно места мерења, и то у току кретања мобилног робота, тј. сензора. Укупно је извршено по пет мерења изнад беле подлоге окружења и црних контролних тачака, при чему свако од мерења садржи по 1000 узорака.

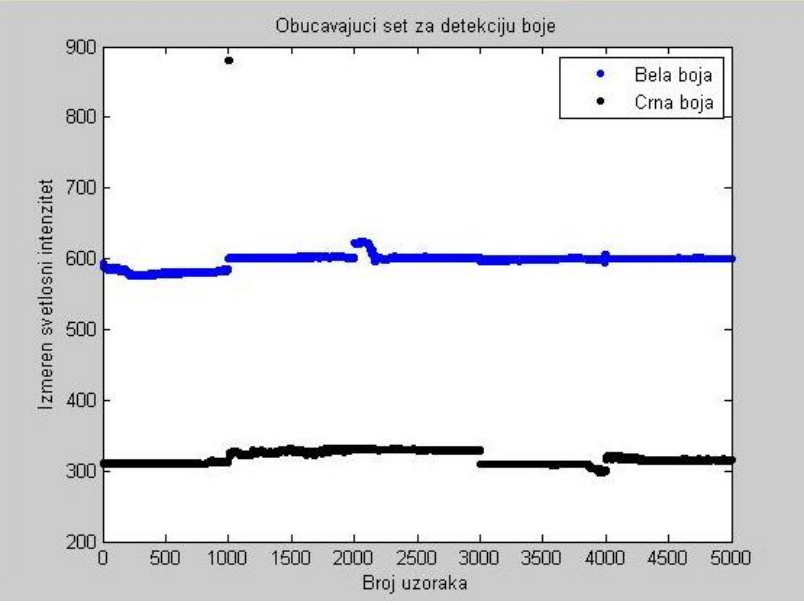
Експериментално се утврђује оптимална топологија, као и парамети учења, а константни остају:

Param.show = 50 - број итерација након ког се приказује промена.
lr = 0.1 и mc = 0.9 - константе момента ВНМ.
epochs = 1000 - број итерација након кога се процес зауставља без обзира на испуњеност услова обучености мреже
max_fail = 10 - Број дозвољених падова на тесту приликом валидације
trainlm - мреже користе Ливенберг - Марков алгоритам обучавања.

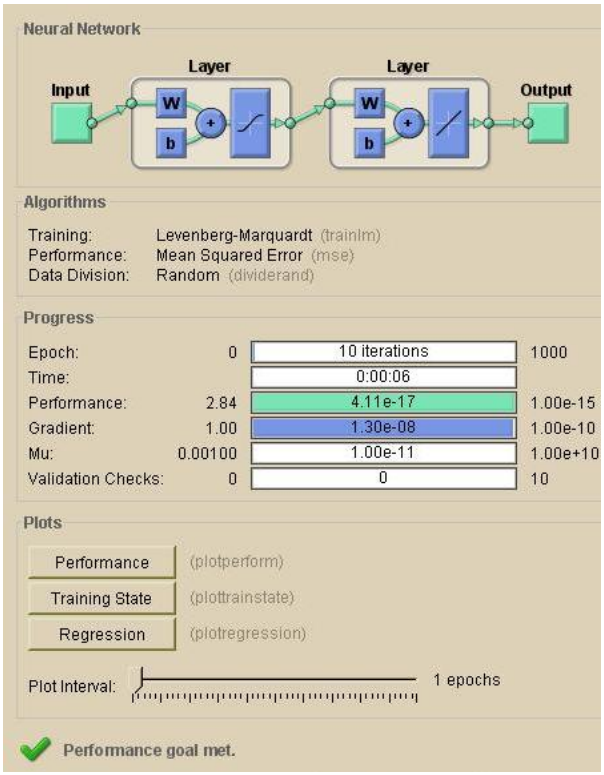
***Табела 6**:* План експеримента за обучавање мреже		
Број неурона у скривеном слоју	Параметар учења	Ред. бр. експеримента
4	0.05	1
	0.25	2
	0.75	3
6	0.05	4
	0.25	5
	0.75	6
9	0.05	7
	0.25	8
	0.75	9

Табела 7: Резултати обучавања ВНМ-а за детекцију контролних тачака у радном окружењу мобилног робота					
Ред. бр експеримента	Број понављања	Завршен процес минимизације грешке	Остварена минимална грешка	Остварена вредност градијента	Број негативних валидација мреже
1	522	Да	$5.58 \cdot 10^{-14}$	$9.96 \cdot 10^{-11}$	0
2	868	Да	$3.60 \cdot 10^{-14}$	$9.98 \cdot 10^{-11}$	0
3	396	Да	$1.84 \cdot 10^{-14}$	$9.88 \cdot 10^{-11}$	0
4	399	Да	$9.83 \cdot 10^{-16}$	$6.63 \cdot 10^{-11}$	0
5	743	Да	$9.99 \cdot 10^{-16}$	$1.48 \cdot 10^{-10}$	0
6	632	Да	$1.32 \cdot 10^{-15}$	$1.00 \cdot 10^{-10}$	0
7	11	Да	$8.53 \cdot 10^{-17}$	$3.78 \cdot 10^{-9}$	0
8	10	Да	$4.11 \cdot 10^{-17}$	$1.30 \cdot 10^{-8}$	0
9	20	Да	$9.55 \cdot 10^{-16}$	$1.24 \cdot 10^{-8}$	0

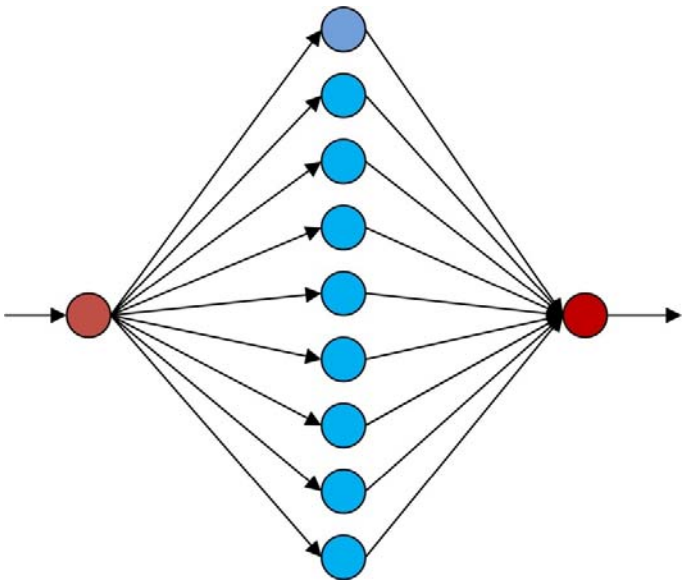
Одабир мреже ће се извршити на основу добијених вредности приказаних у табели. Може се уочити да експерименти 7 и 8 имају најбоље резултате, вредности минималне грешке и вредности градијента. Усвајамо осми експеримент јер има најмању остварену грешку.



Слика 6.3.3 *:* Графички приказ обучавајућих парова за вештачку неуронску мрежу за осми експеримент



Слика 6.3.4 : Приказ резултата обучавања из *Neural Network Toolbox*-а изабране ВНМ за обучавање препознавања контролних тачака у окружењу мобилног робота за осми експеримент.



Слика 6.3.5 : Изабрана мрежа за одређивање команде за препознавање контролних тачака

7. Калманов филтер и његова примена за потребе управљања мобилним роботом

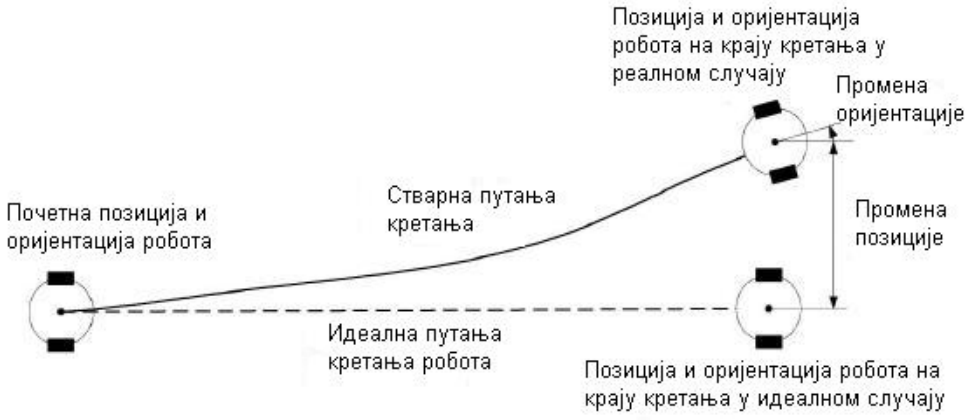
7.1. Особине Калмановог филтера

Често је потребно проценити право стање неког процеса на темељу несигурних и непотпуних мерења која садрже разне шумове. Уз измерени шум постоји и процесни шум због којег процес не заврши тачно у оном стању које је предвиђено самим моделом процеса. Појаву шумова чине несавршености компонената у току израде, као и утицај средине у којој се врши мерење. Од пресудног значаја за филтрирање реалних сигнала је познавање природе шума који се жели одстранити. Класификација шума услед његове разноврсности представља велики изазов. Услед непознавања нежељених компонената прибегава се моделовању грешке која настаје при реалним условима мерења.

Један од алата за естимацију стања процеса је Калманов филтер, темељен на познатом моделу процеса и мерења. Рекурзивна оптимална оцена, односно филтрирање, представља процес естимације стања и/или параметара система на основу вредности добијених мерењем, које пристижу током времена.

Калманов филтер је изумео *Rudolf E. Kalman* (19.05.1930). Овај филтер је оптималан у случају када је расподела вероватноће стања у којем се процес налази у неком дискретном временском кораку једнака Гаусовој расподели. То значи да је расподела униформна, односно да има само једну изражену хипотезу о могућем стању процеса. Оптималност такође зависи од претпоставки да је тренутно стање линеарно зависно о претходног стања, као и да је мерење линеарно зависно од тренутног стања. Уколико су наведени услови задовољени, Калманов филтер за такве процесе даје најбоље могуће резултате. Ако су модели процеса нелинеарни онда се разним техникама попут ЛКФ-а покушавају оптимално линеаризовати уз присутност неизоставне апроксимацијске грешке. Линеаризација је онолико исправна колико је стварно понашање процеса могуће описати Гаусовом расподелом јер у случају апроксимације неке друге расподеле естимације су најчешће нетачне. Предност Калмановог филтера је рекурзивност јер није потребно памтити сва претходна мерења, већ се у тренутној итерацији користи само најбоља естимација претходног стања процеса која је рекурзивно одређена на темељу свих претходних мерења. Иако наведена ограничења Калмановог филтера сужавају могући простор примене, области примене Калмановог филтера су многобројне, и крећу се од обраде сигнала до теорије управљања, а и добијени су задовољавајући резултати за већину занимљивих и корисних процеса попут управљања објектима, у нашем случају мобилиним роботом, као и у авионској и аутомобилској индустрији.

На следећој слици је приказан пример разлике између програмираног кретања и кретања које је робот извршио. Као што се примећује долази до промена и позиције и оријентације робота. У нашем случају је то добрим делом условљено компонентама које су уграђене у мобилни робот, али је ово одступање између реалне и програмиране путање кретања произведено и одређеним шумовима који се јављају у току процеса.



Слика 7.1.1: Поређење жељене и остварене позиције и оријентације програмираног робота

7.2. Принцип рада Калмановог филтера и његово коришћење

Калманов филтер - *КФ* (*Kalman filter*) је ефикасно рекурзивно решење за проблем дискретног линеарног филтрирања. Представља скуп математичких једначина за имплементацију естиматора типа предиктор-коректор, за постизање оптималности у смислу минимизације оцене коваријансе грешке. Због тога се овај филтер назива још и оптимални или једноставни КФ.

Принцип рада Калмановог филтера се заснива на томе да се при читању неке сензорске информације $y_{(t)}$, коју чине збир две величине од којих је једна величина сигнал који се жели очитати, а друга величина је шум који се јавља при жељеном очитавању. Услед овога је неопходну одредити колики део сензорске информације припада сигналу који се мери ($x_{1(t)}$), а колики део припада шуму ($x_{2(t)}$).

$$y_{(t)} = x_{1(t)} + x_{2(t)} \tag{7.1}$$

Услед свега овога, при уптреби Калмановог филтера неопходно је дефинисати следеће појмове:

- *Филтрација* - представља процес при којем се врши раздвајање сигнала и шума , тј одређивање удела сигнала $x_{1(t)}$ и шума $x_{2(t)}$ у сензорској информацији $y_{(t)}$.
- *Предикција* - процес при којем се прикупљају информације до жељеног тренутка да би се са што више прикупљених информација могло извршити што боље предвиђање стања система или само једног елемента система у неком наредном тренутку.
- *Интерполација* - процес интерполације карактерише прикупљање валидних информација о неком процесу или систему који се посматра

Калманов филтер претпоставља да се линеарни динамички систем напише у форми једначина. Коришћењем тих једначина врши се оцењивање стања система и генерисање информација о стању посматраног система.

Сходно овоме, могу се извести једначине стања система:

$$X_t = A_t X_{t-1} + B_t u_t + \varepsilon_t \tag{7.2}$$

$$p(x|u_t, X_{t-1}) = N(X_t; A_t X_{t-1} + B_t u_t, R_t) \tag{7.3}$$

Једначина мерења система (опсервациони модел) :

$$Z_t = C_t X_t + \delta_t \tag{7.4}$$

$$p(Z_t|X_t) = N(z_t; C_t X_t, Q_t) \tag{7.5}$$

При чему наведене ознаке представљају :

X_t - вектор стања система у тренутку t;

X_{t-1} - вектор стања у тренутку t;

Z_t – вектор мерења у тренутку t;

A_t - Матрица типа (nxn) која одређује како систем од тренутка t долази до тренутка t-1 без утицаја управљања или шумова;

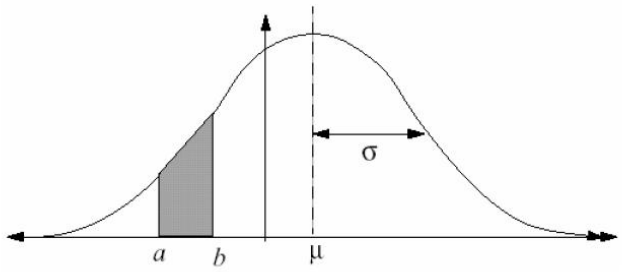
B_t - Матрица (nxl) која одређује како управљања u_t мењају стање система од t до t-1;

C_t - Матрица (kxn) дефинише пресликавање x_t у мерење z_t;

u_t - вектор управљања који дефинише прелазак стања система из стања t-1 у стање t;

ε_t, δ_t - Случајне променљиве које представљају шумове (поремећаје) у једначини стања и једначини мерења. Оне су независне и подлежу Гаусовој расподели са познатим матрицама коваријанси R_t и Q_t респективно.

На следећој слици је приказан пример Гаусове расподеле, на којој почива примена Калмановог филтера.



Слика 7.2.1: Приказ Гаусове расподеле

7.3. Алгоритми Калмановог филтера и Калмановог линеаризованог филтера

7.3.1. Калманов филтер

bel $(x_0) = N(x_0; \mu_0, \sum_0)$ - почетно стање система ;

bel$(x_t) = N(x_t; \mu_t, \sum_t)$ - стање система у тренутку t.

$$\dots bel = belief \dots$$

Табела 8: Алгоритам Калмановог филтера	
1. АЛГОРИТАМ Калманов филтер $(\mu_{t-1}, \sum_{t-1}, u_t, z_t)$	***ПРЕДИКЦИЈА*** *(prediction, time update)*
2. $\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$	
3. $\bar{\Sigma}_t = A_t \Sigma_{t-1} A^T{}_t + R_t$	
4. $K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1}$	***КОРЕКЦИЈА*** *(update correction, measurement update)*
5. $\mu_t = \bar{\mu}_t + K_t(z_t - C_t \bar{\mu}_t)$	
6. $\mu_t = \bar{\mu}_t + K_t(z_t - C_t \bar{\mu}_t)$	
7. $return\ \mu_t, \Sigma_t$	

Из претходне табеле се уочавају нове прменљиве чије ознаке гласе :

K_t – Калманово појачање;
μ_t - очекивана вредност расподеле;
Σ_t - матрица коваријанси.

7.3.2. Линеаризовани Калманов филтер

Уколико имамо случај да је функција нелинеарна, онда је неопходно користити линеаризован Калманов филтер. Да би се Калманов филтер применио на нелинеарним моделима, неопходно је модел линеаризовати у околини тренутне тачке, тј. нелинеарну функцију представити линеарном у једној тачки. Једначину стања нелинеарне функције управљања и претходног стања приказујемо следећом једначином :

$$x_t = g(u_t, x_{t-1}) \tag{7.6}$$

Опсервациони модел нелинеарне функције стања система се описује следећом једначином:

$$z_t = h(x_t) \tag{7.7}$$

Да би се применио Линеаризовани Калманов филтер, неопходно је најпре извршити развој функције у Тејлоров ред. Изводи функције $g(u_t, x_{t-1})$ за изабрану тачку μ_{t-1} и функције $h(x_t)$ у изабраној тачки μ_t могу се записати као :

$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + \frac{\partial g(u_t, \mu_{t-1})}{\partial x_{t-1}}(x_{t-1} - \mu_{t-1}) \tag{7.8}$$

$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + G_t(x_{t-1} - \mu_{t-1}) \tag{7.9}$$

$$h(x_t) \approx h(\bar{\mu}_t) + \frac{\partial h(\bar{\mu}_t)}{\partial x_t}(x_t - \bar{\mu}_t) \tag{7.10}$$

$$h(x_t) \approx h(\bar{\mu}_t) + H_t(x_t - \bar{\mu}_t) \tag{7.11}$$

Опис наведених променљивих гласи :

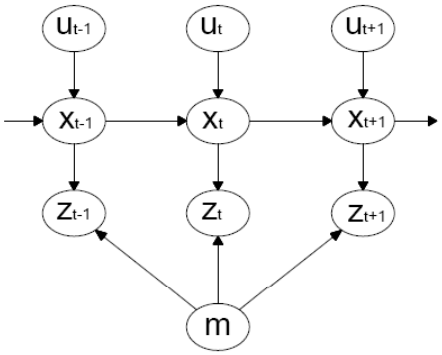
G_t – промена функције модела кретања у односу на положај робота
H_t – промена мерења са променом координата мобилног робота

Табела 9. Алгоритам Линеаризованог Калмановог филтера ЛКФ	
1. АЛГОРИТАМ линеаризован калманов филтер $(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$	***ПРЕДИКЦИЈА*** *(prediction, time update)*
2. $\bar{\mu}_t = g(u_t, \mu_{t-1})$	
3. $\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$	
4. $K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1}$	***КОРЕКЦИЈА*** *(update correction, measurement update)*
5. $\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$	
6. $\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t$	
7. $return\ \mu_t, \Sigma_t$	

7.4. Примена Калмановог филтера у процесу одређивања положаја мобилног робота

Мобилни робот мора у сваком тренутку да зна свој положај, да би могао обављати постављене задатке. Проблем проналажења и праћења положаја мобилног робота назива се локализацијом, која може бити глобална или локална. У овоме раду се објашњава локална локализација, која подразумева праћење положаја мобилног робота уз претпоставку да су познати његов почетни положај, кинематички модел и модел радног простора. Радни простор може бити статички или динамички, али се за потребе овог пројекта користи статички радни простор.

Претпоставке које се уводе у вези са мапом радног окружења су да је мапа окружења апсолутно тачна и да се на њој налази положај свих непомичних објеката који се налазе у окружењу. На следећој слици је дат шаблонски приказ мапе окружења.



Слика 7.4.1: Локализација мобилног робота

Локализација се може поделити на :

- *Глобална локализација* – почетни положај мобилног робота није познат, а вектор стања подлеже униформној расподели.
- *Локална локализација* – почетни положај мобилног робота је познат, а вектор стања подлеже нормалној (Гаусовој) расподели.

Посматрано са становишта управљања мобилним роботом, локализацију можемо поделити на:

- *Активну* – алгоритми локализације имају потпуну контролу над основним алгоритмом управљања мобилним роботом.
- *Пасивну* – подсистем за локализацију мобилног робота представља само један подсистем управљања и има само једну намену: одређивање позиције и оријентације мобилног робота, док главни систем надгледа функционисање и извршавање постављеног задатка.

Окружење у којем се налази робот може бити подељено као:

- Статичко окружење – окружење у којем се једино мобилни робот креће
- Динамичко окружење – окружење у којем и остали објекти поред мобилног робота могу мењати положај током времена

Табела 10: Алгоритам Линеаризованог Калмановог филтера , корак предикције		
	Формула	Назив
1.	$G_t = \frac{\partial g(u_t, \mu_{t-1})}{\partial x_{t-1}} = \begin{pmatrix} \frac{\partial x}{\partial \mu_{t-1,x}} & \frac{\partial x}{\partial \mu_{t-1,y}} & \frac{\partial x}{\partial \mu_{t-1,\theta}} \\ \frac{\partial y}{\partial \mu_{t-1,x}} & \frac{\partial y}{\partial \mu_{t-1,y}} & \frac{\partial y}{\partial \mu_{t-1,\theta}} \\ \frac{\partial \theta}{\partial \mu_{t-1,x}} & \frac{\partial \theta}{\partial \mu_{t-1,y}} & \frac{\partial \theta}{\partial \mu_{t-1,\theta}} \end{pmatrix}$	Јакобијан функције *g* у односу на полозај
2.	$V_t = \frac{\partial g(u_t, \mu_{t-1})}{\partial u_t} = \begin{pmatrix} \frac{\partial x}{\partial v_t} & \frac{\partial x}{\partial \omega_t} \\ \frac{\partial y}{\partial v_t} & \frac{\partial y}{\partial \omega_t} \\ \frac{\partial \theta}{\partial v_t} & \frac{\partial \theta}{\partial \omega_t} \end{pmatrix}$	Јакобијан функције *g* у односу на управљање
3.	$M_t = \begin{pmatrix} (\alpha_1 \lvert v_t \rvert + \alpha_2 \lvert v_t \rvert) & 0 \\ 0 & (\alpha_3 \lvert \omega_t \rvert + \alpha_4 \lvert \omega_t \rvert) \end{pmatrix}$	Шум управљања
4.	$\bar{\mu_t} = g(u_t, \mu_{t-1})$	Предвиђена очекивана вредност
5.	$\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + V_t M_t V_t^T$	Предвиђена матрица коваријанси

Табела 11: Алгоритам Линеаризованог Калмановог филтера , корак корекције		
Р. Бр.	Формула	Назив
1.	$\hat{z}_t = \begin{pmatrix} \sqrt{(m_x - \bar{\mu}_{t,x})^2 + (m_y - \bar{\mu}_{t,y})^2} \\ atan2(m_y - \bar{\mu}_{t,y}, m_x - \bar{\mu}_{t,x}) - \bar{\mu}_{t,\theta} \end{pmatrix}$	Предвиђена очекивана вредност мерења
2.	$H_t = \frac{\partial h(\mu_t, m)}{\partial x_t} = \begin{pmatrix} \frac{\partial r_t}{\partial \mu_{t,x}} & \frac{\partial r_t}{\partial \mu_{t,y}} & \frac{\partial r_t}{\partial \mu_{t,\theta}} \\ \frac{\partial \varphi_t}{\partial \mu_{t,x}} & \frac{\partial \varphi_t}{\partial \mu_{t,y}} & \frac{\partial \varphi_t}{\partial \mu_{t,\theta}} \end{pmatrix}$	Јакобијан функције x у односу на положај
3.	$Q_t = \begin{pmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_r^2 \end{pmatrix}$	Шум мерења
4.	$S_t = H_t \bar{\Sigma}_t {H_t}^T + Q_t$	Предвиђена матрица коваријанси мерења
5.	$K_t = \bar{\Sigma}_t {H_t}^T {S_t}^{-1}$	Калманово појачање
6.	$\mu_t = \bar{\mu}_t + K_t(z_t - \hat{z}_t)$	Предвиђена очекивана вредност
7.	$\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t$	Нова матрица коваријанси

8. Алгоритми за претраживање

Алгоритми претраживања служе за проналажење најкраће путање од почетне до крајње тачке, а при томе избегавају препреке, минимизирају трошкове (гориво, време, новац, итд...). Алгоритми претраживања се баве проблемом одабира путање. Могуће је да се потроши већи напор на само један од њих. У првом екстремном случају софистицирани алгоритми претраживања заједно са тривијалним алгоритмима за кретање би пронашли путању када објекат почиње да се креће и објекат ће следити тај пут, без обзира на све остало. Друга екстремна варијанта је један покрет, алгоритам не би рачунао целу путању (иницијална путања би била права линија), уместо тога узеће један корак у једном временском тренутку, узимајући у обзир локално окружење у свакој тачки.

8.1. Графови (решетке, мреже)

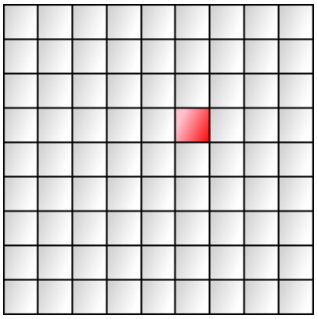
Алгоритми за претраживање и налажење путање, заснивају се на графовима. Графови у математичком смислу представљају скуп чворова са ивицама које их повезују. Постоје неколико врста графова, тј. решетки (мрежа), које су састављење од понављања основних облика.

Графови обухватају:

- ✓ четвороуглове, шетоуглове или троуглове;
- ✓ темена, ивице, лица (плочице);
- ✓ координатни систем и
- ✓ алгоритам за рад са графовима (решеткама).

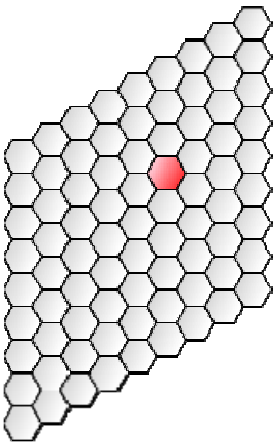
8.1.1. Облици графова

Најчешће коришћен граф је четвороугаона решетка (мрежа), која ће бити коришћена и у нашем пројекту из разлога што је локација одређена правоуглим (картезијанским) координатама (x, y), тј. осе су ортогоналне.



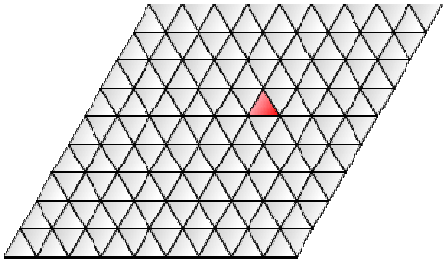
Слика 8.1.1: Четвороугаона решетка (мрежа)

У великом броју случајева су коришћене и шестоугаоне решетке јер оне нуде мање расипања удаљености од квадрата мреже. То је делом зато што шестоугао има више не дијагоналних суседних квадрата. (Дијагонале изискују веће раздаљине у мрежи). Шестоугаоне мреже постоје у природи, као на пример саће у кошницама пчела.



Слика 8.1.2: Шестоугаона решетка (мрежа)

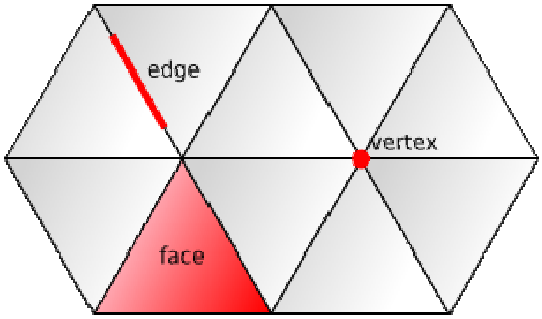
Велики недостатак троугаоних решетка, поред непознавања је велики обим и мала површина, што је супротно шестоугаоној мапи, и ређе се користе.



Слика 8.1.3: Троугаона решетка (мрежа)

8.1.2. Делови графова

Решетке имају три врсте области: лица (плочице), ивице и темена. Свако лице је дводимензионална површина окружена ивицама. Свака ивица је једнодимензионални сегмент, који се завршава са два темена. Свака тачка је нултих димезија. Уобичајено је да се фокусира само на један од ових типова делова. У нашем случају ће то бити лица (плочице).



Слика 8.1.4: Делови решетке (мрежа)

8.1.3. Бројање делова графова

Може се израчунати колико је лица, ивица и темена потребно да се формира мрежа. Приступ је такав да се гледа суседство и дељење. Сваки троугао има 3 ивице. Очекује се 3 пута онолико ивица колико има лица. Међутим свака ивица се дели са 2 лица тако да имамо 3 ивице са два лица. Сваки троугао има 3 темена. Свако теме деле 6 лица. Због тога постоје 3 темена на сваких 6 лица или 1 теме на 2 лица. Ти односи су важни за пројектовање координатних система.

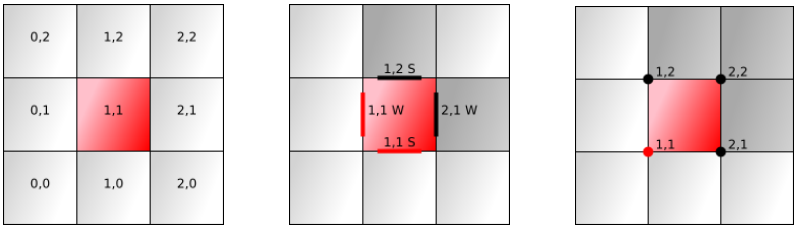
Табела 12: Бројање делова

Облик	Лица, Faces (F)	Линије, Edges (E)	Тачке, Vertices (V)
Четвороугао	1	2	1
Шестоугао	1	3	2
Ттроугао	2	3	1

F,E,V од квадрата је 1,2,1; од шестоугла је, 1,3,2; а 2,3,1 код троугла. Шестоугаоне и троугаоне решетке имају сличне тачке, изузев темена и лица чији је броји обрнут. Ако поставимо тачку у центру сваког квадрата мреже добићемо још квадрата мреже.

8.1.4. Координатни системи

Постоје три дела мреже, где се налази начин да се адресира сваки од њих. Почиње се са једноставним нумеричким координатама које одговарају осама мреже. F, E, V бројеви нам говоре колико делова мреже деле исту координату.

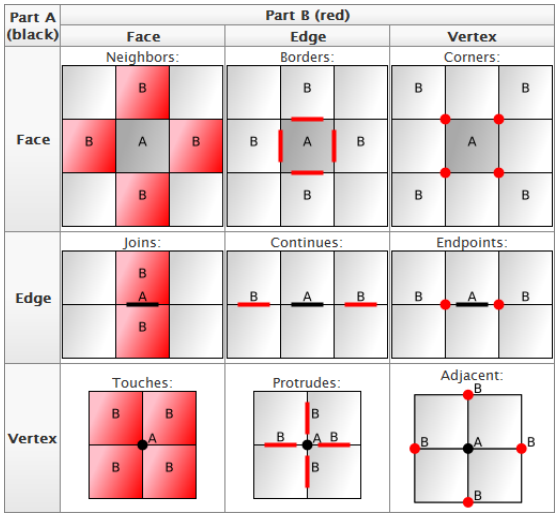


Слика 8.1.5: Четвороугаона решетка, координате лица (face), ивица и тачака

Први дијаграм на слици 8.1.5. за стандардни координатни систем, за лица *"F, E, V"* броје се са 1,2,1. То значи да ће само ивицама требати ознака за разврставање. За свако лице (произвољно), додељујемо једно теме да дели своје координате. Изабран је доњи леви угао квадрата. За сваки квадрат смо доделили две странице да деле своје координате. Изабране су лева и доња ивица, и додељене координатама са словима C и B. Упоређујући дијаграме види се како су координате повезане.

8.1.5. Односи између делова у решетки

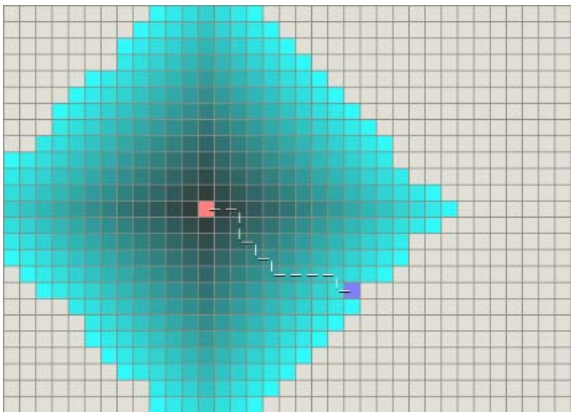
Имајући у виду сваки од 3 дела мреже, могу се дефинисати односи између та 3 дела из мреже (решетке), дајући нам 9 односа.



Слика 8.1.6: Односи између делова у решетки [11]

8.2. Dijkstra's Algorithm

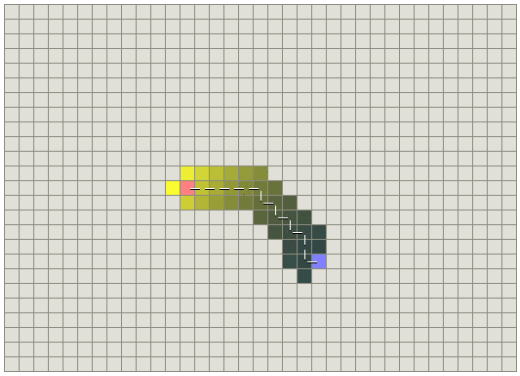
Дајкстра алгоритам функционише тако што посећује чворове у графу, (2D дискретизованом окружењу), почевши од полазне тачке објекта. Онда, више пута, испитује најближе теме, додајући своје чворове на скуп чворова које треба испитати. Он се шири ка споља од почетне тачке па све док се не достигне циљ. Дајкстра алгоритам гарантује да ће пронаћи најкраћи пут од почетне тачке до циља, све док ниједна од ивица нема негативну цену. У следећем дијаграму, розе квадрат је полазна тачка, плави квадрат је циљ, и светлоплаве области приказују шта Дајкстра алгоритам скенира. Најсветлија светлоплава поља су она која су најудаљенија од почетне тачке, и тиме чине границу за истраживање:



Слика 8.2.1: Пример претраживања Дајкстра алгоритмом

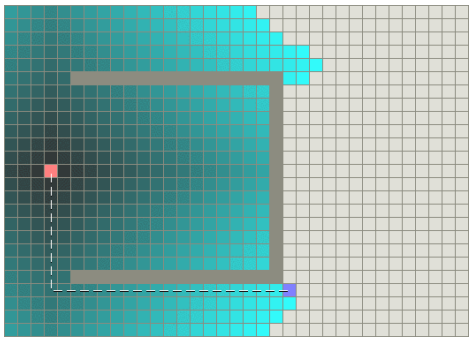
8.3. Greedy Best-First-Search

Greedy Best-First-Search алгоритам ради на сличан начин осим што користи хеуристику да би проценио колико далеко од циља је неки пиксел или ивица. Уместо да изабере ивицу која је најближа почетној тачки, он бира ивицу најближу циљу. Овај алгоритам не гарантује да ће пронаћи најближи пут. Међутим, он ради много брже у односу на Дајкстра алгоритам због тога што користи хеуристичке функције да нађе свој пут до циља веома брзо. На пример, ако је циљ јужно од почетне тачке, он ће се фокусирати на путању која води јужно, тј. према југу. На следећој слици жута боја претставља чворове са високим хеуристичким вредностима (висока цена за долажење до циља).



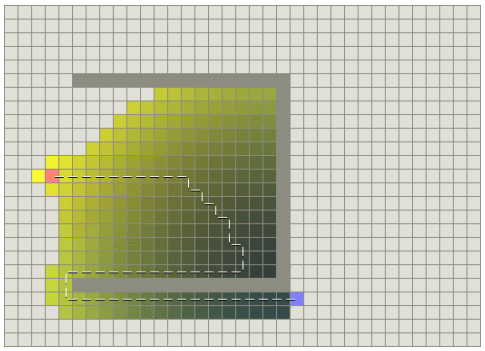
Слика 8.3.1: Пример претраживања “*Greedy Best-First-Search”* алгоритам

Међутим, сви ови примери показују случај, када на мапи (графу) нема препрека. Ако узмемо у обзир конвексну препреку приказану на слици испод, Дајкстра алгоритам ради спорије и теже, али гарантује да ће наћи најкраћу путању.



Слика 8.3.2: Пример претраживања са укљученом препреком Дајкстра алгоритам

Greedy Best-First-Search алгоритам, са друге стране, ради мање, али његова путања није тако добра, може се видети на слици испод.



Слика 8.3.3: Пример претраживања са укљученом препреком *“Greedy Best-First-Search”* алгоритам

Проблем са *Greedy Best-First-Search* алгоритмом је његова “похлепа – greedy”, што покушава да стигне до циља, по сваку цену. Пошто он узима само цену долажења до циља, а игнорише трошкове пређеног пута до сада, он чак иде и ако је путања постала исувише дугачка.

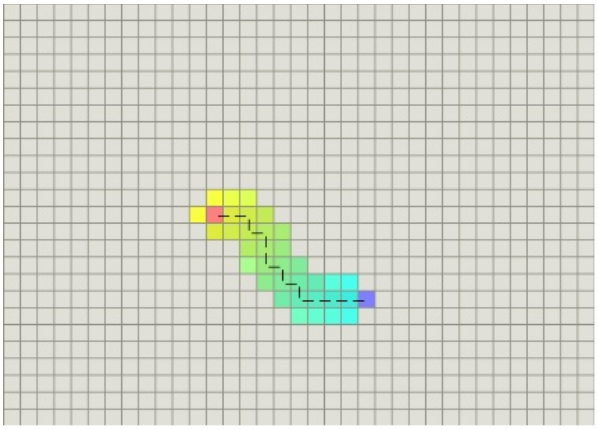
8.4. A алгоритам*

A*, (A звезда), је најпопуларнији избор за креирање путање, јер је веома флексибилан и може се користити у разним околностима.

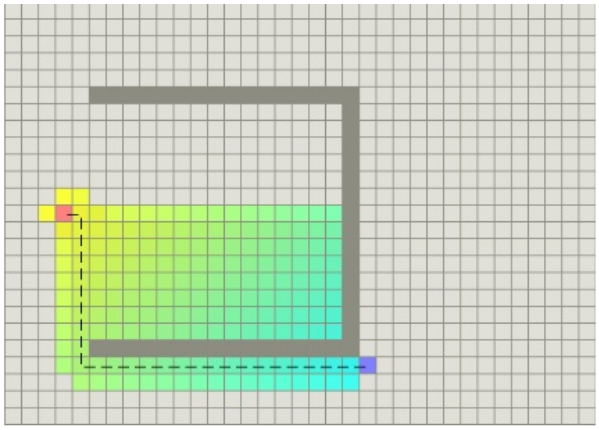
A* је развијена 1968. године и комбинује хеуристички приступ као горе објашњени Greedy Best-First-Search алгоритам и формалин приступ као Дајкстра алгоритам.

A* је као и остали алгоритми за граф-претраживање, с тим што потенцијално може да тражи огромну површину мапе (графа). Она је као алгоритам Дајкстра с тим што може да се користи за проналажење најкраћег пута. Она је као Greedy Best-First-Search алгоритам с тим што може да користи хеуристички начин да води сама себе. У најједноставнијем случају, то је тако брзо као *Greedy Best-First-Search*.

У примеру са конкавним препрека, A* проналази пут исто тако добро као и Дајкстра алгоритам.



Слика 8.4.1: Пример претраживања A*



Слика 8.4.2: Пример претраживања A* са укљученом препреком

A* комбинује информације које Дајкстра алгоритам користи (фаворизовање темена која су близу почетне тачке) и информација које први Greedy Best-First-Search користи (фаворизовање темена која су близу циља). У стандардној терминологији која се користи када се говори о A*, $g(n)$ представља трошкове (цена) пута од почетне тачке до било ког темена n, и $h(n)$ представља хеуристички процењене трошкове од темена n до циља. У горњем дијаграму,

жута (x) представља темена далеко од циља а (г) представља даљину темена од почетне тачке.

Сваки пут кроз главне петље, испитује се теме које има најниже $f\,(n) = g\,(n) + h\,(n)$.

8.4.1. Коришћење хеуристике

Хеуристичка фукција $h(n)$ говори A* и процењује минималну цену кретања из било ког темена n до циља.

Хеуристика се користи за контролу понашања A* у неколико случајева:

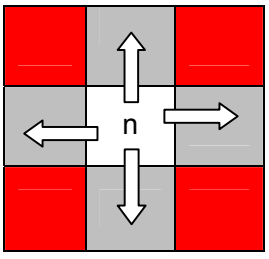
- Када је $h(n) = 0$, тада главну улогу има $g(n)$ и A* се претвара у Дајкстра алгоритам који гарантује да ће наћи најкраћу путању.
- Ако је $h(n)$ увек мање од (или једнако) цени померања из n до циља, онда A* гаранује да ће наћи најкраћу путању. Што је нижа вредност $h(n)$ више се шире чворови A*, што је чине споријом.

8.4.2. Хеуристика за мреже, односно графове

На графовима се примењују добро познате хеуристичке функције, тј. норме, $h\,(n)$ од којих су:

Менхетн норма – Растојање између две тачке је збир апсолутне разлике у његовим координатама. Примењује се на квадратној мрежи или графу која дозвољава померање у 4 правца. Ова норма је добила назив из разлога што је трајекторија алгоритма таква да подсећа на распоред већине улица на острву Менхетн, САД.

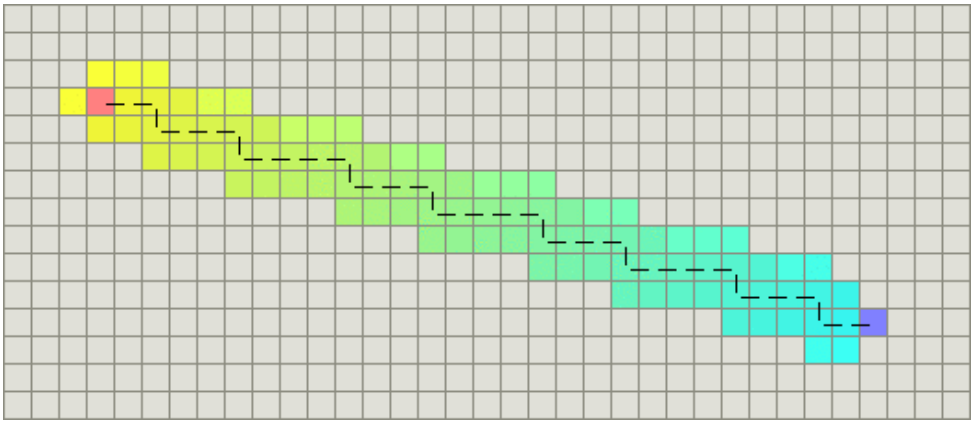
Гледа фукцију цене и тражи минималну цену за пребацивање из једног простора у други.



Слика 8.4.3: Део квадратног графа, померање пиксела у 4 правца

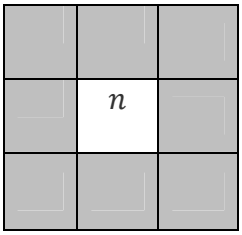
Једначина Менхетн (Manhattan) норме [8.1]

$$h(n) = D * (abs(n.x - goal.x) + abs(n.y - goal.y) \qquad [8.1]$$



Слика 8.4.4: Пример трајекторије добијене Менхетн нормом

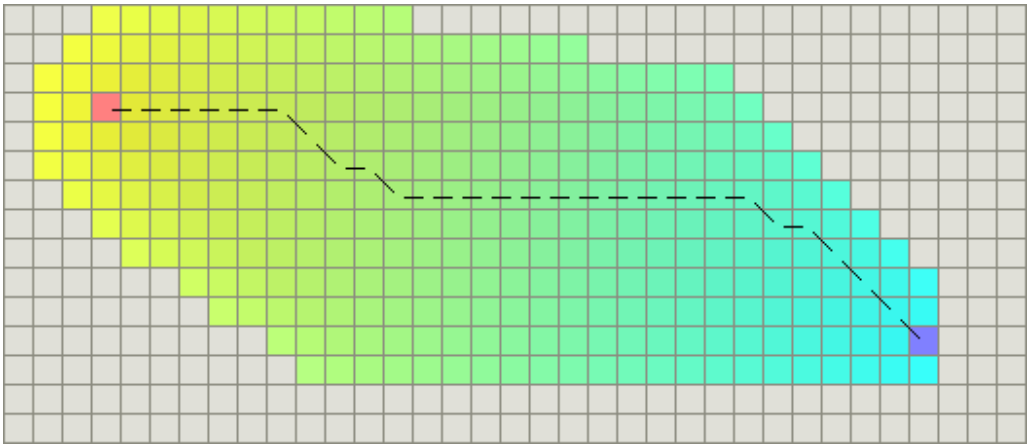
Еуклидска норма – Примењује се на квадратној мрежи или графу, која дозвољава померање у 8 праваца. Дијагонално растојање између два поља изражено је преко тигонометије и приказано у једначини испод.



***Слика 8.4.5*:** Део квадратног графа, померање пиксела у 8 правца

Једначина Еуклидске (Euclidean) норме [8.2]

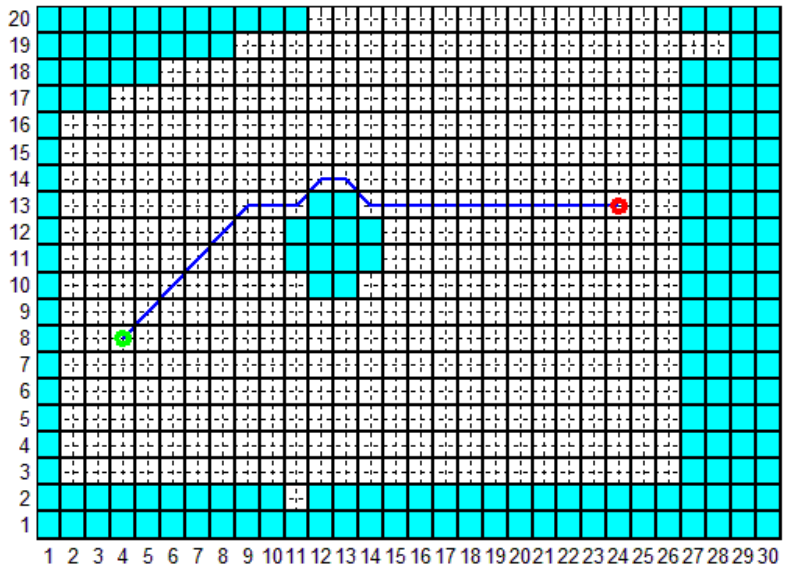
$$h(n) = D * sqrt((n.x - goal.x)\text{^}2 + (n.y - goal.y)\text{^}2) \qquad [8.2]$$



Слика 8.4.6: Пример трајекторије добијене Еуклидском нормом

8.4.3. Имплементација A алгоритма*

На почетку, у тачки 8.1 било је речи о графовима и његовим врстама. На основу тога окружење мобилног робота је представљено у облику квадратног графа, тј. дискретизовано је у k пиксела. Од величине пиксела зависи време и прецизност претраживања. Ако је пиксел већи, време претраживања ће бити веће, а прецизност мања и обратно. У нашем случају један пиксел одговара растојању од 5cm. Пошто су димензије нашег радног окружења за вршење експеримента, $150x100mm$, то ће бити $30x20$ пиксела. Тако смо добили мапу окружења у облику матрице димензија 30x20.



Слика 8.4.7: Дискртетизовано радно окружење

Алгоритам А* састоји се из три под-алгоритма, односно три под-функције, а то су “crtez”, “heuristika”, и “predjeni_put_i_nova_pozicija”.

Алгоритам “crtez” можемо схватити као кориснички интерфејс (User friendly) за графички приказ и контролу експеримента, слика 8.4.7. Могу се јасно видети исцртани пиксели, као и њихова средишта, испрекиданом линијом. Пиксели плаве боје су препреке, тј. технолошко окружење са диспозиционим планом машина. Наравно, њих је при кретању робота потребно опслуживати, без колизије робота, тј. физичког контакта. Алгоритам је исписан испод, ту се налази матрица препрека, на основу које овај алгоритам једноставним кодом исцртава радно окружење. Потребно је нагласити да је овај алгоритам врло флексибилан, јер је потребно променити само матрицу препрека, да би се променила цела скица окружења.

```
function [s] = graficki_prikaz(rezultat_i,rezultat_j,pot,krt)

plot(rezultat_i,rezultat_j,'LineWidth',2)    % iscrtavanje grafika predjenog puta

matrica_prepreka=[10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000
,10000,10000,10000,10000,10000,10000,10000,10000,10000;...
    10000,10000,0,0,0,0,0,0,0,0,0,0,0,0,0,0,10000,10000,10000,10000;...
    10000,10000,0,0,0,0,0,0,0,0,0,0,0,0,0,0,10000,10000,10000,10000;...
    10000,10000,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,10000,10000,10000;...
    10000,10000,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,10000,10000,10000;...
    10000,10000,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,10000,10000;...
    10000,10000,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,10000,10000;...
    10000,10000,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,10000,10000;...
    10000,10000,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,10000;...
    10000,10000,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,10000;...
    10000,0,0,0,0,0,0,0,0,0,10000,10000,0,0,0,0,0,0,0,10000;...
    10000,10000,0,0,0,0,0,0,0,10000,10000,10000,10000,0,0,0,0,0,0,0;...
    10000,10000,0,0,0,0,0,0,0,10000,10000,10000,10000,0,0,0,0,0,0,0;...
    10000,10000,0,0,0,0,0,0,0,0,10000,10000,0,0,0,0,0,0,0,0;...
    10000,10000,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;...
    10000,10000,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;...
    10000,10000,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;...
    10000,10000,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;...
    10000,10000,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;...
    10000,10000,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;...
    10000,10000,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;...
    10000,10000,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;...
    10000,10000,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;...
    10000,10000,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;...
    10000,10000,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;...
    10000,10000,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;...

10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000
,10000,10000,10000,10000,0,10000;...

10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000
,10000,10000,10000,10000,0,10000;...

10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000
,10000,10000,10000,10000,10000,10000;...

10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000
,10000,10000,10000,10000,10000,10000;]';
```

```
size_matrica_prepreka_x = size(matrica_prepreka,2);
size_matrica_prepreka_y = size(matrica_prepreka,1);


for i=0:size_matrica_prepreka_x+1
for j=0:size_matrica_prepreka_y+1
rectangle('Position',[0.5+i, 0.5+j, 1, 1],'LineWidth',2)
hold on
end
end

xlim([0.5 30.5])
ylim([0.5 20.5])
grid on
 s=1:31;
  set(gca,'XTick',s);
  set(gca,'YTick',s);
  hold on

  j11=matrica_prepreka(:,1);
  j111=j11';

  for i2=1:size_matrica_prepreka_x
      for j2=1:size_matrica_prepreka_y
      if matrica_prepreka(j2,i2)>0
%           if matrica_prepreka(j2,:)>0
          rectangle('Position',[0.5+i2-1,                0.5+j2-1,                1,
1],'LineWidth',2,'FaceColor','c')
          hold on
%           end
      end
      end
  end

plot(pot(1),pot(2),'o','LineWidth',3,'color','green')
hold on
plot(krt(1),krt(2),'o','LineWidth',3,'color','r')
hold on
%
```

Алгоритам “heuristika”, израчунава растојање између почетне и крајње тачке. То рачунање почиње од циљне, тако што је у циљној тачки вредност нула, вредности у нормалним правцима се постепено увећавају за инкремент један, а по дијагонали тригонометриском једначином.

Такво рачунање представља Еуклидску норму, па наше окружење шематски изгледа као на слици 8.4.8.

h=20	20.41	20.83	21.24	21.66	.	36.06
.	.	.	.	.	.	
h=4	4.41	4.83	5.24	5.66	.	31.66
h=3	3.41	3.83	4.24	5.24	.	31.24
h=2	2.41	2.83	3.83	4.83	.	30.83
h=1	1.41	2.41	3.41	4.41	.	30.41
h=0	h=1	h=2	h=3	h=4	.	h=30

Слика 8.4.8: Дискретизовани део радног окружења са вредностима хеуристике

Значи, свако поље у матрици препрека садржи вредност оцене тог поља. То је вредност хеуристике за свако поље *h*, $h\ (n)$, по Еуклидској норми.

Овим вредностима додељују се вредности параметра $g(h)$, то су трошкови пута, као што речено у некој од претходних тачака. Зеленом бојом је означен тренутни пиксел, сивом су означени могући помераји са њиховим трошковима.

g=1.41	*g=1*	*g=1.41*
g=1	*n*	*g=1*
g=1.41	*g=1*	*g=1.41*

Слика 8.4.9: Цена помераја од пиксела "*n"* до суседних пиксела

Ако у окружењу постоји препрека онда ће она бити дефинисана са већом ценом, односно трошком, на тај начин смо обезбедили да нам путања не пролази кроз препреку, јер алгоритам претраживања тражи најнижу цену кретања.

G=100	g=100	g=100
g=1	n	g=1
G=1.41	g=1	g=1.41

Слика 8.4.10: Цена померања од пиксела "*n*" до суседних пиксела са препрекама

На основу наведених параметара, $f(n)$ израчунава цену најкраћег пута од старног до циљног пиксела. Избором минималне вредности овог параметра за све суседне пикселе у односу на пиксел у коме се тренутно налази A* бира следећи пиксел и све тако док не достигне циљни пиксел, што је и приказано на слици 8.4.11.

h=20	20.00	20.12	20.20	20.22	.	36.06
.	.	.	.	.	.	
h=4	4.12	4.47	5.00	5.66	.	30.27
h=3	3.16	3.61	4.24	5.00	.	30.15
h=2	2.24	2.83	3.61	4.47	.	30.07
h=1	1.41	2.24	3.16	4.12	.	30.02
h=0 g=1 f=1	H=1	h=2	h=3	h=4	.	h=30

Слика 8.4.11: Графички приказ кретања по Еуклидској норми

Алгоритам ове фукције је приказан испод.

```
function [h] = heuristika(i_ciljno, j_ciljno)

for i = 1:30   %dodati end na kraju i zameniti u krajnjatacka i=x
for j = 1:20   %dodati end na kraju i zameniti u krajnjatacka j=y

krajnjatacka = [j_ciljno i_ciljno];        %krajnja tacka

% clc, clear all, close all
% krajnjatacka = [19 19];

inkh=1 ;                          %inkrement pomeraja horizontalno
inkv=1 ;                          %inkrement pomeraja vertikalno
inkd=sqrt((inkh^2)+(inkv^2));     %inkrement pomeraja dijagonalno

h = ones(30,20)*0;  %generisanje mape, prazna matrica, nula dole levo

x = krajnjatacka(1,1);    %Krajnje tacka koord. x
y = krajnjatacka(1,2);    %Krajnje tacka koord. y

h(x,y) = 0; %za krajnju tacku u matrici dodeljuje vrednost 0
%POGLEDATI GDE CEMO STAVITI KOORDINATNI SISTEM JER MATRICA RACUNA ZA TACKU

ndes = size(h,2)-krajnjatacka(1,1);    %broj tacaka desno od krajnje tacke do kraja
mape
nlev = size(h,2)-ndes-1;    %broj tacaka levo  od krajnje tacke do kraja mape
ndol  = size(h,1)-krajnjatacka(1,2);     %broj tacaka na dole od krajnje tacke do
kraja mape
ngor = size(h,1)-ndol-1;     %broj tacaka na gore od krajnje tacke do kraja mape

%=============== DODELA VREDNOSTI HORIZONTALNO I VERTIKALNO ===============

for k = 0 : ndes
    h(y,x+k) = k*inkh;  %dodeljivanje inkremenata na desno od krajnje tacke, k*inkh
se za svako sledece polje uvecava za inkh
end
for k = 0 : nlev
    h(y,x-k) = k*inkh;  %dodeljivanje inkremenata na levo  od krajnje tacke
end
for k = 0 : ndol
    h(y+k,x) = k*inkv;  %dodeljivanje inkremenata na dole od krajnje tacke
end
for k = 0 : ngor
    h(y-k,x) = k*inkv;  %dodeljivanje inkremenata na gore od krajnje tacke
end

%================== DODELA VREDNOSTI OSTATKU GORE DESNO ==================

if ndes <= ngor
    for k = 0 : ndes-1          %menja tacke od krajnjatacka horizontalno nadesno
        for e = 1 : ndes-k        %menja tacke od krajnjatacka dijagonalno udesno
navise do kraja mape
            h(y-e,x+(e+k))    =    e*inkd+h(krajnjatacka(1,2),krajnjatacka(1,1)+k);
%pocevsi od krajnjatacka pa horizontalno nadesno za svaku tacku penje se po
dijagonali
```

```
                                                                %i dodeljuje vrednosti
%1,2,3... po redu, mnozi ih sa inkd i dodaje osnovnu
                                                                %vrednost       pocetne
horizontalne tacke
        end
    end
else
    for k = 0 : ndes-ngor        %menja od krajnjatacka horiontalno tacke do tacke
cija dijagonala udesno navise zavrsava u desnom gornjem cosku mape
        for e = 1 : ngor         %menja tacke po dijagonali navise udesno do kraja
mape
            h(krajnjatacka(1,2)-e,krajnjatacka(1,1)+(e+k))                        =
e*inkd+h(krajnjatacka(1,2),krajnjatacka(1,1)+k);     %dodeljuje vrednosti kao i u
predhodnom slucaju
        end
    end
    for k = size(h,2)-ngor-nlev : ndes-1     %menja od poslednje tacke u prosloj
petlji do predposlednje tacke na mapi na horiz. potegu od krajnjatacka udesno
        for e = 1 : ndes-k                     %menja tacke po dijagonali navise
udesno
            h(krajnjatacka(1,2)-e,krajnjatacka(1,1)+(e+k))                        =
e*inkd+h(krajnjatacka(1,2),krajnjatacka(1,1)+k);   %dodeljuje vrednosti tackama
        end
    end
end

for k = 1 : ndes            %menja tacke od krajnjatacka nadesno po horizontali
    for e = k+1 : ngor       %menja tacke iznad dijagonale od krajnjatacka na gore
desno
        h(y-e,x+k)  =  h(y-k,x+k)+e-k;       %za svaku tacku iznad dijagonale od
krajnjatacka na gore desno, dodaje 1 novoj tacki u odnosu na tacku vertikalno ispod
nje
    end
end

%=================== DODELA VREDNOSTI OSTATKU GORE LEVO ===================

if nlev <= ngor
    for k = 0 : nlev-1          %menja tacke od krajnjatacka horizontalno nalevo
        for e = 1 : nlev-k        %menja tacke od krajnjatacka dijagonalno ulevo
navise do kraja mape
            h(y-e,x-(e+k))    =    e*inkd+h(krajnjatacka(1,2),krajnjatacka(1,1)-k);
%pocevsi od krajnjatacka pa horizontalno nalevo za svaku tacku penje se po
dijagonali
                                                                %i dodeljuje vrednosti
%1,2,3... po redu, mnozi ih sa inkd i dodaje osnovnu
                                                                %vrednost       pocetne
horizontalne tacke
        end
    end
else
    for k = 0 : nlev-ngor        %menja od krajnjatacka horiontalno tacke do tacke
cija dijagonala ulevo navise zavrsava u levom gornjem cosku mape
        for e = 1 : ngor         %menja tacke po dijagonali navise ulevo do kraja
mape
            h(krajnjatacka(1,2)-e,krajnjatacka(1,1)-(e+k))                        =
e*inkd+h(krajnjatacka(1,2),krajnjatacka(1,1)-k);     %dodeljuje vrednosti kao i u
predhodnom slucaju
```

```
        end
    end
    for k = size(h,2)-ngor-ndes : nlev-1     %menja od poslednje tacke u prosloj
petlji do predposlednje tacke na mapi na horiz. potegu od krajnjatacka udesno
        for e = 1 : nlev-k                      %menja tacke po dijagonali navise
udesno
            h(krajnjatacka(1,2)-e,krajnjatacka(1,1)-(e+k))                        =
e*inkd+h(krajnjatacka(1,2),krajnjatacka(1,1)-k);   %dodeljuje vrednosti tackama
        end
    end
end

for k = 1 : nlev            %menja tacke od krajnjatacka nalevo po horizontali
    for e = k+1 : ngor       %menja tacke iznad dijagonale od krajnjatacka na gore
levo
        h(y-e,x-k)  =  h(y-k,x-k)+e-k;       %za svaku tacku iznad dijagonale od
krajnjatacka na gore desno, dodaje 1 novoj tacki u odnosu na tacku vertikalno ispod
nje
    end
end

%================== DODELA VREDNOSTI OSTATKU DOLE DESNO ===================

if ndes <= ndol
    for k = 0 : ndes-1          %menja tacke od krajnjatacka horizontalno nalevo
        for e = 1 : ndes-k        %menja tacke od krajnjatacka dijagonalno ulevo
navise do kraja mape
            h(y+e,x+(e+k))    =    e*inkd+h(krajnjatacka(1,2),krajnjatacka(1,1)+k);
%pocevsi od krajnjatacka pa horizontalno nalevo za svaku tacku penje se po
dijagonali
                                                                %i dodeljuje vrednosti
%1,2,3... po redu, mnozi ih sa inkd i dodaje osnovnu
                                                                %vrednost       pocetne
horizontalne tacke
        end
    end
else
    for k = 0 : ndes-ndol        %menja od krajnjatacka horiontalno tacke do tacke
cija dijagonala ulevo navise zavrsava u levom gornjem cosku mape
        for e = 1 : ndol         %menja tacke po dijagonali navise ulevo do kraja
mape
            h(krajnjatacka(1,2)+e,krajnjatacka(1,1)+(e+k))                        =
e*inkd+h(krajnjatacka(1,2),krajnjatacka(1,1)+k);     %dodeljuje vrednosti kao i u
predhodnom slucaju
        end
    end
    for k = size(h,2)-ndol-nlev : ndes-1     %menja od poslednje tacke u prosloj
petlji do predposlednje tacke na mapi na horiz. potegu od krajnjatacka udesno
        for e = 1 : ndes-k                      %menja tacke po dijagonali navise
udesno
            h(krajnjatacka(1,2)+e,krajnjatacka(1,1)+(e+k))                        =
e*inkd+h(krajnjatacka(1,2),krajnjatacka(1,1)+k);   %dodeljuje vrednosti tackama
        end
    end
end

for k = 1 : ndes            %menja tacke od krajnjatacka nalevo po horizontali
```

```
    for e = k+1 : ndol       %menja tacke iznad dijagonale od krajnjatacka na gore
levo
        h(y+e,x+k)  =  h(y+k,x+k)+e-k;       %za svaku tacku iznad dijagonale od
krajnjatacka na gore desno, dodaje 1 novoj tacki u odnosu na tacku vertikalno ispod
nje
    end
end

%================== DODELA VREDNOSTI OSTATKU DOLE LEVO ====================

if nlev <= ndol
    for k = 0 : nlev-1          %menja tacke od krajnjatacka horizontalno nalevo
        for e = 1 : nlev-k        %menja tacke od krajnjatacka dijagonalno ulevo
navise do kraja mape
            h(y+e,x-(e+k))    =    e*inkd+h(krajnjatacka(1,2),krajnjatacka(1,1)-k);
%pocevsi od krajnjatacka pa horizontalno nalevo za svaku tacku penje se po
dijagonali
                                                                %i dodeljuje vrednosti
%1,2,3... po redu, mnozi ih sa inkd i dodaje osnovnu
                                                                %vrednost       pocetne
horizontalne tacke
        end
    end
else
    for k = 0 : nlev-ndol        %menja od krajnjatacka horiontalno tacke do tacke
cija dijagonala ulevo navise zavrsava u levom gornjem cosku mape
        for e = 1 : ndol         %menja tacke po dijagonali navise ulevo do kraja
mape
            h(krajnjatacka(1,2)+e,krajnjatacka(1,1)-(e+k))                          =
e*inkd+h(krajnjatacka(1,2),krajnjatacka(1,1)-k);     %dodeljuje vrednosti kao i u
predhodnom slucaju
        end
    end
    for k = size(h,2)-ndol-ndes : nlev-1     %menja od poslednje tacke u prosloj
petlji do predposlednje tacke na mapi na horiz. potegu od krajnjatacka udesno
        for e = 1 : nlev-k                      %menja tacke po dijagonali navise
udesno
            h(krajnjatacka(1,2)+e,krajnjatacka(1,1)-(e+k))                          =
e*inkd+h(krajnjatacka(1,2),krajnjatacka(1,1)-k);   %dodeljuje vrednosti tackama
        end
    end
end

for k = 1 : nlev            %menja tacke od krajnjatacka nalevo po horizontali
    for e = k+1 : ndol       %menja tacke iznad dijagonale od krajnjatacka na gore
levo
        h(y+e,x-k)  =  h(y+k,x-k)+e-k;       %za svaku tacku iznad dijagonale od
krajnjatacka na gore desno, dodaje 1 novoj tacki u odnosu na tacku vertikalno ispod
nje
    end
end

end
end
```

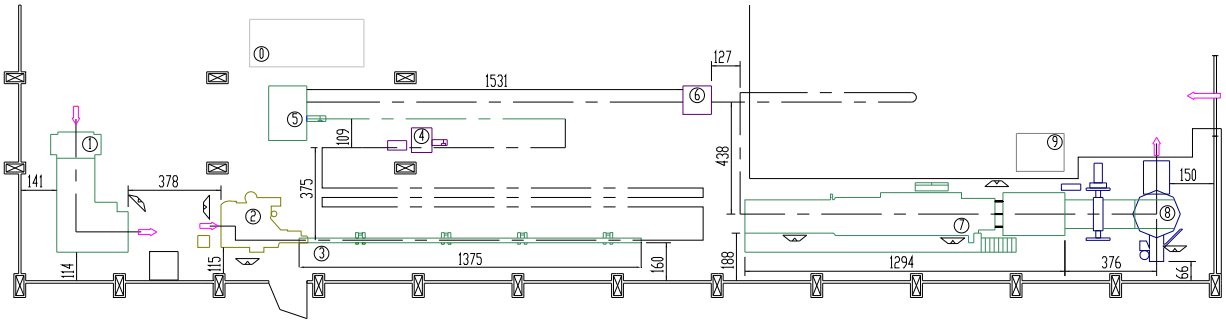
Алгоритам “predjeni_put_i_nova_pozicija“ додељује инкремент пређеног пута између два суседна пиксела и проглашава координате у које ће робот отићи у зависности од почетних координата i и j. Као улазни податак у овом алгоритму је врста и колона у којој се налази минимална вредност за f. На пример, ако је добијена колона 1 и врста 1, нови пиксел који ће бити посећен је дијагонално од тренутног пиксела 2,2 матрице 3x3. Инкремент пређеног пута се добија по еуклидској норми и износи 1,41. “I_novo” и “j_novo” је потребно умањити за 1 пиксел.

```
function                    [predjeni_put_inkrement,i_novo,j_novo]                    =
predjeni_put_i_nova_pozicija(vrsta,kolona,i_pocetno,j_pocetno)

 if kolona == 1 && vrsta == 1
     predjeni_put_inkrement = 1.41
     i_novo = i_pocetno-1
     j_novo = j_pocetno-1
 elseif kolona==1 && vrsta==2
     predjeni_put_inkrement= 1
     i_novo=i_pocetno
     j_novo=j_pocetno-1
 elseif kolona==1 && vrsta==3
     predjeni_put_inkrement= 1.41
     i_novo=i_pocetno+1
     j_novo=j_pocetno-1

 elseif kolona==2 && vrsta==1
     predjeni_put_inkrement= 1
     i_novo=i_pocetno-1
     j_novo=j_pocetno
 elseif kolona==2 && vrsta==2
     predjeni_put_inkrement= 0
     i_novo=i_pocetno
     j_novo=j_pocetno
 elseif kolona==2 && vrsta==3
     predjeni_put_inkrement= 1
     i_novo=i_pocetno+1
     j_novo=j_pocetno

 elseif kolona==3 && vrsta==1
     predjeni_put_inkrement= 1.41
     i_novo=i_pocetno-1
     j_novo=j_pocetno+1
 elseif kolona==3 && vrsta==2
     predjeni_put_inkrement=1
     i_novo=i_pocetno
     j_novo=j_pocetno+1
 elseif kolona==3 && vrsta==3
     predjeni_put_inkrement= 1.41
     i_novo=i_pocetno+1
     j_novo=j_pocetno+1
 end
end
```

9. Развој симулационог модела у AnyLogic софтверском окружењу

Anylogic је алат за моделирање динамичке симулације које обухвата динамику система, централно процесне (дискретни догађаји), и агентно базиране приступе у оквиру једног језика за моделовање и једног развојног окружења модела. Језик *Anylogic* нам омогућава да проучимо комплексност и хетерогеност пословања, економије и социјалних система у било ком жељеном нивоу детаља. Anylogic-ов скуп примитивних и "library" објеката нам омогућава моделирање производње и логистике, пословних процеса, људских ресурса, потрошача и понашања пацијената, као и животну средину.

У овој фази пројектног задатка потребно је развити симулациони модел постојећег диспозиционог плана погона за производњу и палетизацију конзерви у предузећу “XY“, где се врши производња лименки ø73 и ø83.



Слика 9.1: Диспозициони план линије 3

У табели 13 се налази списак машина према ситуационом плану предузећа.

Табела 13: ***Списак и опис машина у производној линији Л3***	
Машина	**Опис**
M1	Маказе за сечење
M2	Аутомат за утискивање полуреза и обликовање
M3	Аутомат за заваривање и пећ
M4	Аутомат за повијање и расецање
M5	Аутомат за ребрење и сужавање
M6	Аутомат за монтирање дна
M7	Палетизатор
M8	Машина за увезивање и улепљивање

У табели 14 су приказана времена обраде делова за сваки део понаособ (у секундама) и растојање између машина или складишта (у метрима).

Табела 14: Приказ времена обраде по технолошким поступцима			
Редни број	**Технолошки поступак**	**Машина**	**Време трајања поступка (s)**
1	Транспорт палете са лимовима од улаза у халу до маказа и спуштање	Виљушкар-бензинац	25
2	Постављање палете на машину	Виљушкар-електрични	179
3	Отпакивање палете	Ручна манипулација	25
4	Подизање палете до вак. сисљки	Аутоматски	25
5	Сечење припремака из табле лима	Маказе	9
6	Скидање трака са маказа и постављање на сто	Ручна манипулација	20
7	Преношење трака и пуњење шаржера	Ручна манипулација	3
8	Утискивање полуреза и обликовање (савијање)	Аутомат	2
9	Заваривање и печење споја	Аутомат и пећ	20
10	Транспорт и хлађење	Магнетни транспортер	111
11	Повијање и расецање	Аутомат	2
12	Транспорт	Магнетни транспортер	10
13	Ребрење и сужавање	Аутомат	4
14	Транспорт	Магнетни транспортер	21
15	Монтирање дна	Аутомат	2.5
16	Транспорт	Магнетни транспортер	17
17	Палетизација	Палетизатор	64
18	Увезивање и облепљивање	Машина	269
19	Одвожење палете са лименкама до врата погона	Виљушкар-бензинац	27

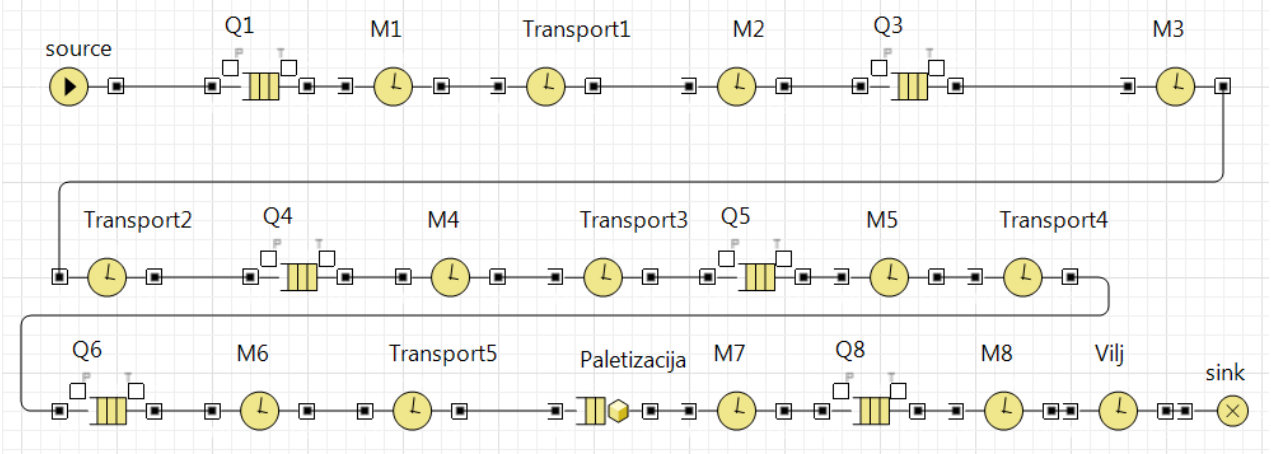
На основу података из табеле 14 извршена је симулација. Од ентитета у симулацији су коришћени:

- ***Source*** - преставља улаз, тј. у нашем случају делове који се достављају у одређеном временском интервалу
- ***Delay*** - овај ентитет је коришћен као машина са дефинисаним временима припреме и обраде материјала, као и за поступке транспорта материјала и припремака са њиховим временима
- ***Queue*** - ентитет који представља ред делова испред машине који чекају на обраду
- ***Sink*** – представља излаз, тј. отпремање готових делова до складишта

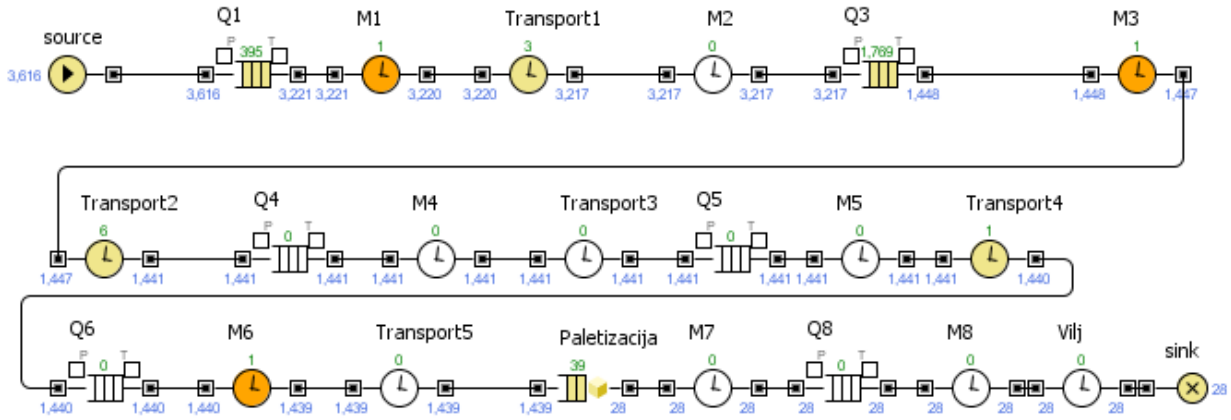
9.1. Симулациони модел без анализе транспортних токова

Симулациони модел транспортног тока слика 9.1.1 моделиран је тако да приближно изгледа диспозиционом плану, ради лакшег сналажења.

Симулација је тако подешена да симулира рад једне смене која траје 8 часова (28.800 секунди). Као што се са слике 9.1.1 види, испред сваке машине налази се ред чекања за ту машину. Узима се у обзир да је број припремака једнак броју готових делова.



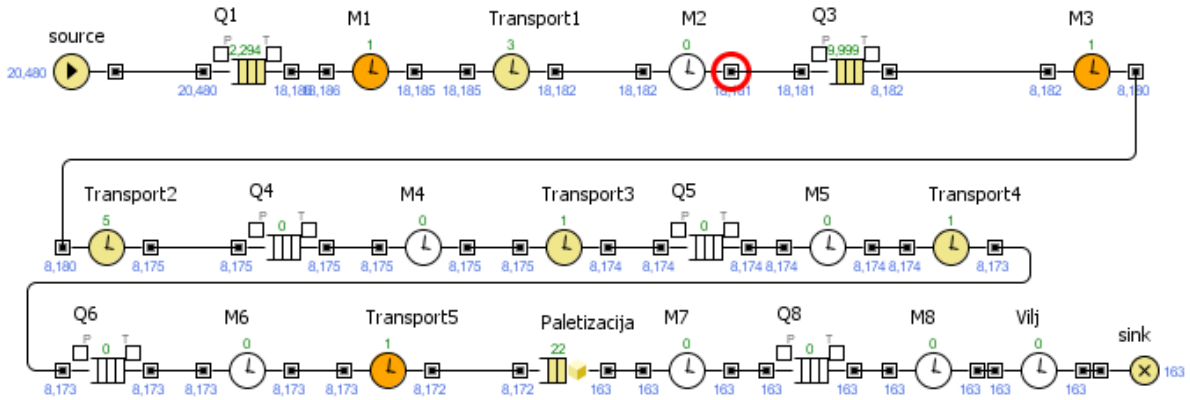
Слика 9.1.1: Симулациони модел производног тока



Слика 9.1.2: Симулациони модел у току симулације

На слици 9.1.2 се види да ће након 8 часовне смене бити израђено 1439 лименки, која ће бити складиштена на 28 палета са по 50 комада. Пошто је ова симулација реалне производне линије са реалним растојањима између машина и временима обрада није дошло до никаквих загушења у току рада.

После одређеног периода долази до загушења слика 9.1.3, тако да смо покушали унапредити пројектовањем новог диспозиционог плана технолошког окружења.

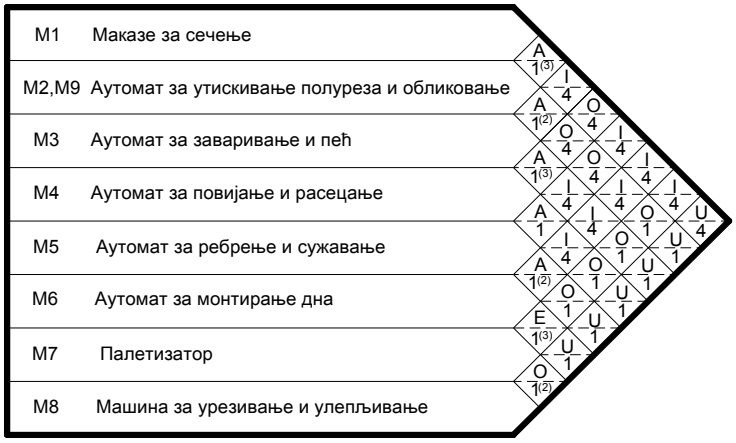


Слика 9.1.3: Симулациони модел на крају тока симулације

9.2. Пројектовање диспозиционог плана технолошког окружења

Пројектовање диспозиционог плана предузећа врши се помоћу троугаоне матрице која показује зависности између машина.

Троугаона матрица слика успоставља зависност машина по критеријумима као што се може видети на слици 9.2.1.



ozn.	Степен зависности
A	апсолутно неопходно
E	веома важно
I	важно
O	потребно
U	неважно

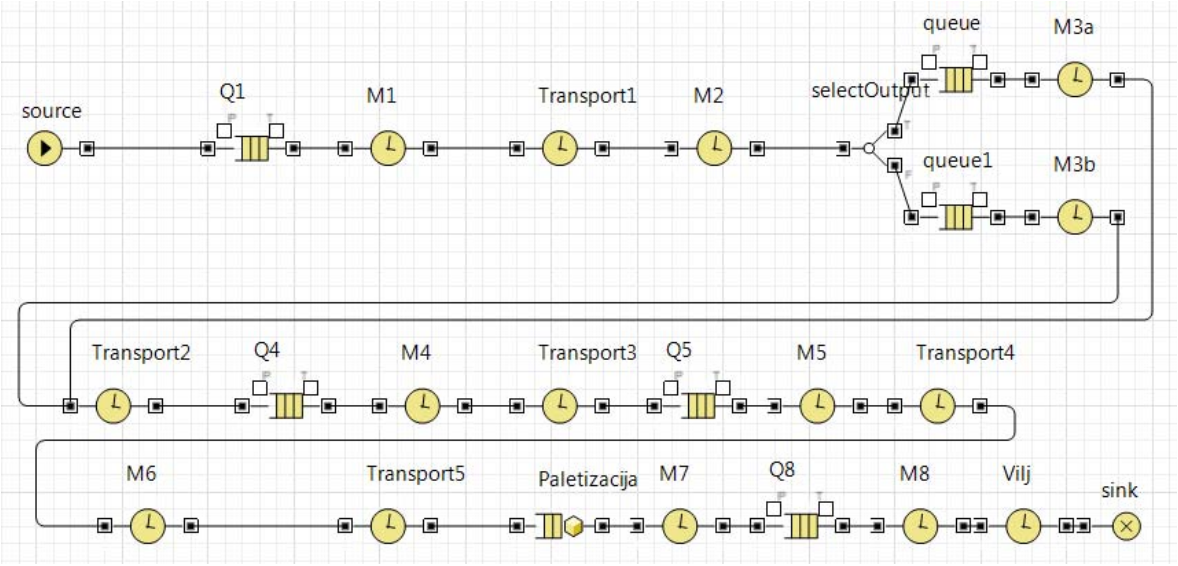
озн.	Разлог
1	ток материјала
2	одржавање машине
3	промена производног асортимана
4	међуфазна контрола

Слика 9.2.1: Троугаона квалитативна матрица међузависности активности на основу степена зависности и разлога веза између машина

9.3. Ново предложено решење технолошког окружења

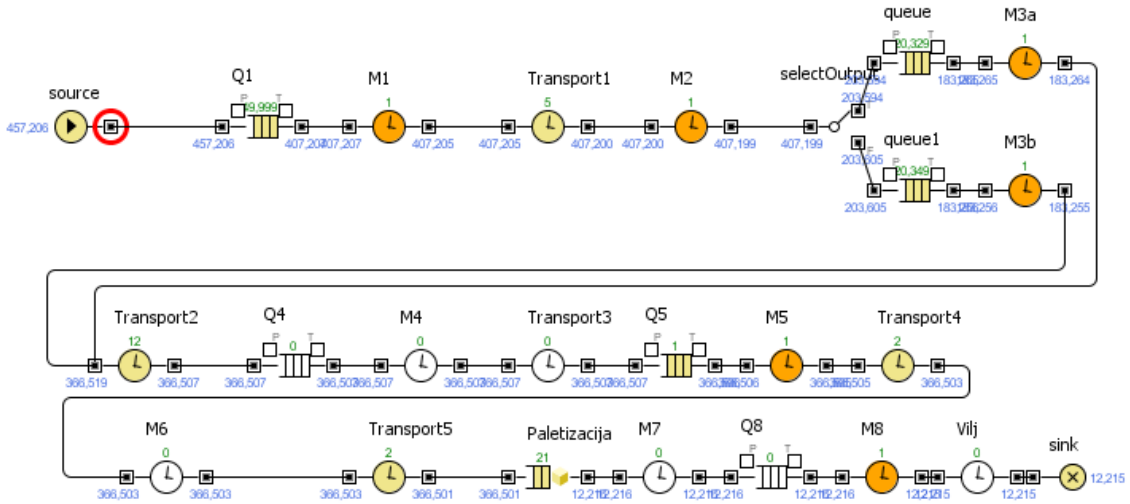
Загушење у производном процесу се могу избећи увођењем додатаних машина на местима где долази до застоја и чекања на ред. На слици 9.3.1 је дат симулациони модел оваквог решења. Симулиран је рад једне смене у трајању од 8 сати.

У конкретном случају, машина M3 замењена је са машином M3a и машином M3b са редом чекања.



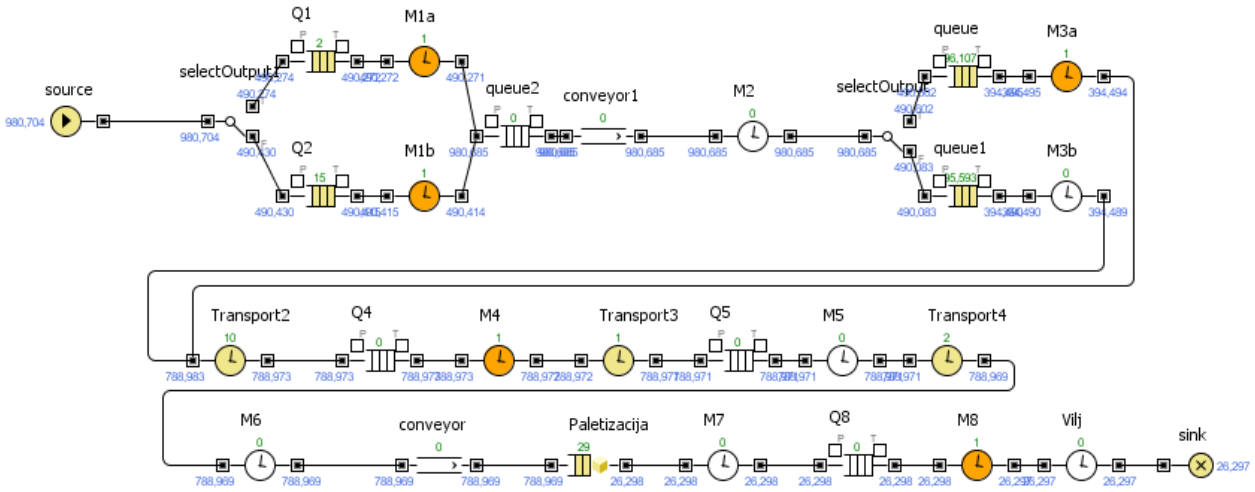
Слика 9.3.1: Симулациони модел производног тока након измене

Такође и овде се види да се испред сваке машине налази ред чекања. Испред машине M7 се налази палетизација која врши скупљање лименки и прослеђује их машини M7, која врши њихово паковање у одговарајуће палете.



Слика 9.3.2: Симулациони модел на крају симулације након прве измене

На слици 9.3.2 приказан је симулациони модел на крају симулације, где се виде застоји након дужег периода. Зато су извршене додатне измене, уместо машине M1 уведене су машине M1a и M2b које ће спречити застоје слика 9.3.3.



Слика 9.3.3: Симулациони модел производног тока на крају симулације након коначних измена

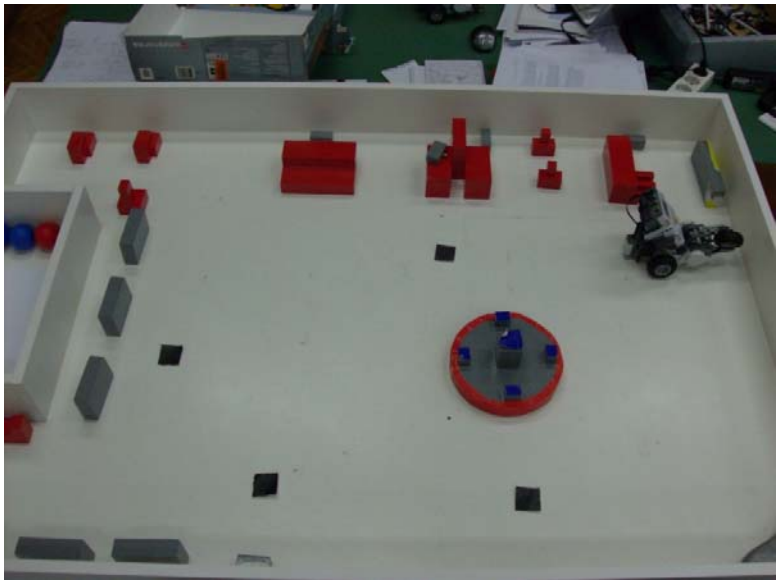
Након унетих измена на слици 9.3.3 се види да ће након 8 часовне смене бити израђено 788.909 лименки, која ће бити складиштена на 26.297 палета са по 30 комада. У овом симулационом моделу је смањен број произведених лименки, али су зато спречени застоји приликом производње.

10. Експериментална верификација пројектног решења

Експериментална верификација се састоји у томе да се робот *LEGO MINDSTORMS NXT* тестира у моделу технолошког окружења предузећа „XY”, опслужујући моделе машина и радних станица.

Циљ је да се на основу управљачког алгоритма покрене робот у реалном окружењу, при том крећући се по задатим тачкама и заобилазећи препреке, да се утврди његово понашање и да се упореде добијени резултати у реалном окружењу, са резултатима добијених из управљачког алгоритма MATLAB, графици пређеног пута, елипсе несигурности, као и провера прихватања контролних тачака и кориговања позиције.

Окружење за вршење експеримента је модел технолошког окружења приказана на слици 10.1, димензија 150x100cm. Унутар ње су расподељене макете машина и радних станица, као према диспозиционом плану машина. У некој од претходних тачака је детаљно описано технолошко окружење са диспозиционим планом машина и радних станица.



Слика 10.1: *Модел технолошког окружења*

У управљачком алгоритму кретања робота дефинисани су следећи параметри:

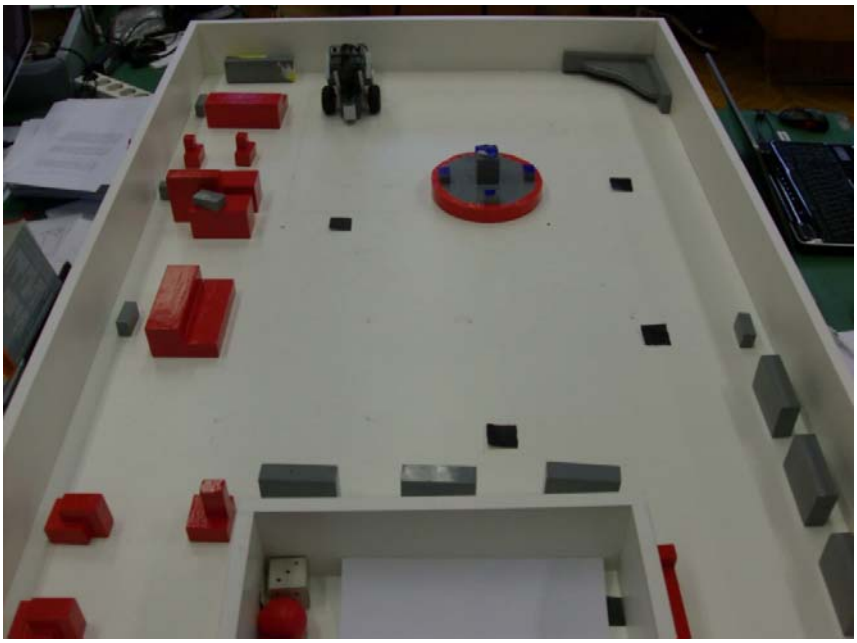
- конструкционе карактеристике робота:
  - осно растојање између точкова, $b = 10.5mm$,
  - полупречник точкова, $r = 2.9mm$
- координате контролних тачака, $map = [30\ 30;\ 70\ 30]$;
- координате почетних тачака, међуположаја и крајњих тачака,
- $stoptacke = [4\ 6;\ 10\ 6;\ 16\ 6;\ 16\ 15]$
- дозвољена грешка при позиционирању мобилног робота, $d\_wp = 8cm,$
- угао између тренутног правца кретања и правца ка наредној задатој тачки за који је неопходно извршти ротацију пре кретања у напред, $fi\_steer = 15°$.

На основу горе наведених параметара може се уочити да су контролне тачке, почетне тачке, међутачке, као и крајње тачке задате у оквиру матрица димензија $2xn$, где је n број тачака које робот треба да достигне, а број 2 представља две координате те тачке (x,y).

Координате контролних тачака имају веће вредности јер су изражене у центиметрима, док су координатне почетних, међу и крајњих тачака изражене у пикселима. Подсећање: $1\,pixel\, = 5\,cm$.

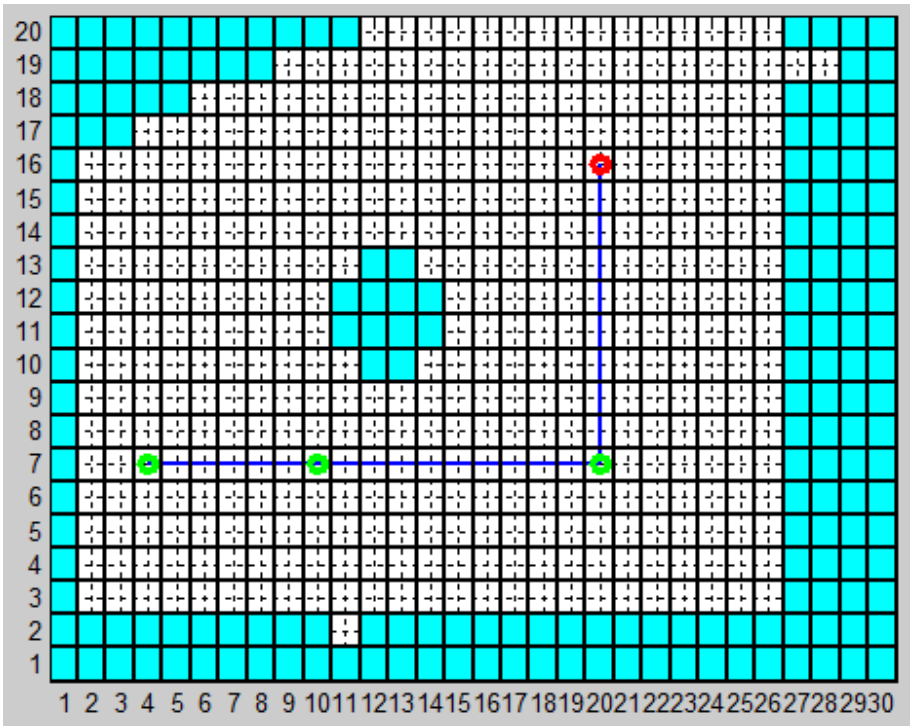
На почетку кретања робота, односно непосредно пре покретања управљачког алгоритма, робот се налази у положају као на слици 10.2. На слици запажамо и црне квадрате који нам представљају контролне тачке, као и место вршења монтаже.

Све позиције машина и сва радна места су узета у обзир када је формирана матрица препрека у A* алгоритму.



Слика 10.2: Почетни положај робота и контролне тачке

На основу дефинисаних улазаних података, а покретањем 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 под називом *“path”.*



Слика 10.3: Оптимална путања од почетне па до циљне међутачке

За сваку од тих међутачака из наведене матрице, упоређујући тренутну позицију и орјентацију са наредном тачком, у наставку алгоритма, израчунава се растојање и потребна орјентација робота. Испод је приказан део управљачког алгоритма који извршава ову операцију.

```
for k = 1 : sizpath

    StopMotor('all', 'off');
    ResetMotorAngle(MOTOR_B); ResetMotorAngle(MOTOR_C);
    d_wp = ITS_compute_distance(x, path(k,:))
    fi_steer = ITS_compute_direction(x,path(k,:))*57.3
```

Подфункција за израчунвање растојања од тренутне до наредне тачке приказана је испод.

```
function d = ITS_compute_distance(pose, cp)
%
% robot's distance to goal (checkpoint)
% pose ==> [x y theta]'
% checkpoint ==> [x y]'
%=========================================================================
dx = cp(1)-pose(1);
dy = cp(2)-pose(2);
d = sqrt((dx)^2 + (dy)^2);
```

Такође, приказана је и подфункција за израчунавање оријентације.

```
function [phi] = ITS_compute_direction(pose, cp)

%[phi] = compute_direction(pose,checkpoint);
% pose ==> [x y theta]'
% checkpoint ==> [x y]'

alfa = atan2((cp(2) - pose(2)),(cp(1)-pose(1)));

phi = alfa - pose(3);

if phi >= pi
    phi = phi - 2*pi;
elseif phi <= -pi
    phi = phi + 2*pi;
end
```

Ако се пре почетка кретања тренутна орјентација робота разликује за више од "fi_steer_max", тада се прво изврши ротација робота пре кретања унапред. Тада ротацију вратила мотора одређује вештачка неуронска мрежа (ВНМ) која је обучена за ову намену. У овом случају је то *"mrezatocak4",* што се може видети из управљачког кода испод.

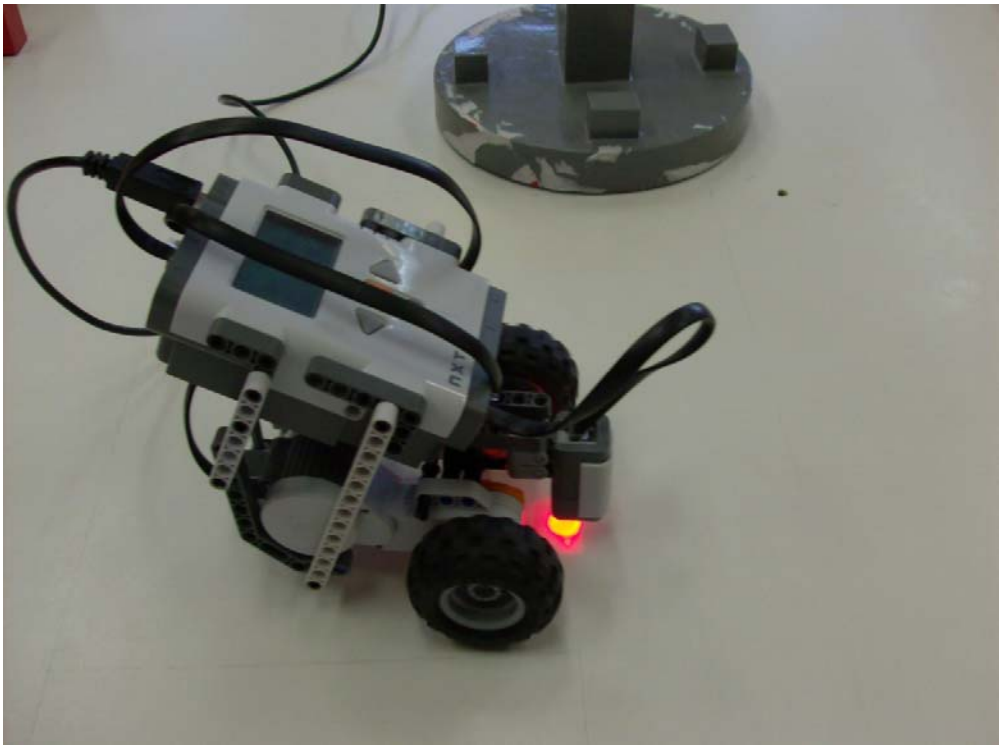
```
if abs(fi_steer) >= 15

                    if fi_steer <= 0
                trrot = -100;
            else trrot = 100;
                    end

alrot = round(sim( mrezatocak4,fi_steer));%round zaokruzuje na celu vrednost

        StopMotor('all', 'off');
        ResetMotorAngle(MOTOR_B); ResetMotorAngle(MOTOR_C);
        SetMotor(MOTOR_C); % accesses the motor b
        SyncToMotor(MOTOR_B); % provides same controls to motor c
        SetPower 15 %
        SetTurnRatio (trrot)  %
        SetAngleLimit (alrot) %
        SendMotorSettings
        %pause(1)
        WaitForMotor(MOTOR_B);
        %pause(1)
        MC = GetMotorSettings(MOTOR_C);
        MB = GetMotorSettings(MOTOR_B);
```

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.

```
while d_wp > 8
```

```
            MC = GetMotorSettings(MOTOR_C);
            MB = GetMotorSettings(MOTOR_B);

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

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

            [x,C] = modelkretanja(x, C, dsr, dsl, b, M)
            ResetMotorAngle(MOTOR_B); ResetMotorAngle(MOTOR_C);
            subplot(2,1,1); plot_simulation_data(x, C),hold on;

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

    if abs(fi_steer) >= 15

                if fi_steer <= 0
                    trrot = -100;
                else trrot = 100;
                end
                alrot = round(sim( mrezatocak4,fi_steer));%round zaokruzuje na
    celu vrednost

                StopMotor('all', 'off');
                ResetMotorAngle(MOTOR_B); ResetMotorAngle(MOTOR_C);
                SetMotor(MOTOR_C); % accesses the motor b
                SyncToMotor(MOTOR_B); % provides same controls to motor c
                SetPower 15 %
                SetTurnRatio (trrot)  %
                SetAngleLimit (alrot) %
                SendMotorSettings
                %pause(1)
                WaitForMotor(MOTOR_B);
                %pause(1)
                MC = GetMotorSettings(MOTOR_C); MB = GetMotorSettings(MOTOR_B);

                dsl = MC.Angle/57.3*r; dsr = MB.Angle/57.3*r;
                M = [10^-5*abs(dsl) 0;0 10^-5*abs(dsr)]; % control uncertainty
                [x,C] = modelkretanja(x, C, dsr, dsl, b, M);
```

Из кода се види да се ту још једном проверава да ли се оријентација разликује за више од “fi_steer_max”.



Слика 10.4: Робот у току кретања

У току кретања се стално врши израчунавање тренутне позиције (x, y) и оријентација (θ) у координатном систему модела технолошког окружења. Ово израчунавање се врши на основу мерења угаоног помераја вратила мотора које врше енкодери. То се јасно може видети у коду.

Пошто мотори не извршавају задате команде идеално, такође ни њихови енкодери не врше идеално тачна мерења, тада се израчунати положај разликује од оствареног. Из тог разлога, у току кретања је потребно вршити корекцију израчунатог положаја.

Корекција израчунатог положаја се остварује на основу резултата мерења светлосног сензора, а на основу одређивање боје подлоге помоћу вештачке неуронске мреже која је развијена за одређивање контолних тачака у радном окружењу мобилног робота. Напомињемо да се мерења помоћу светлосног сензора остварују искључиво када робот стоји у међуположају пре извршавања наредног кретања. Управљачки код за ову функцију је приказан испод.

```
s=Getlight(SENSOR_1)
        S=[S;s]
        xi = 2*(s-min(xx))/(max(xx)-min(xx))-1
        %xi = (s - (max(xx)+min(xx))./2)./((max(xx)-min(xx))./2)
        Yp=sim(netff11,xi)
        yp=[yp;Yp]

        if Yp<0
            [x, C]=ITS_data_association(x, C, Q, map)
        else
        end
```

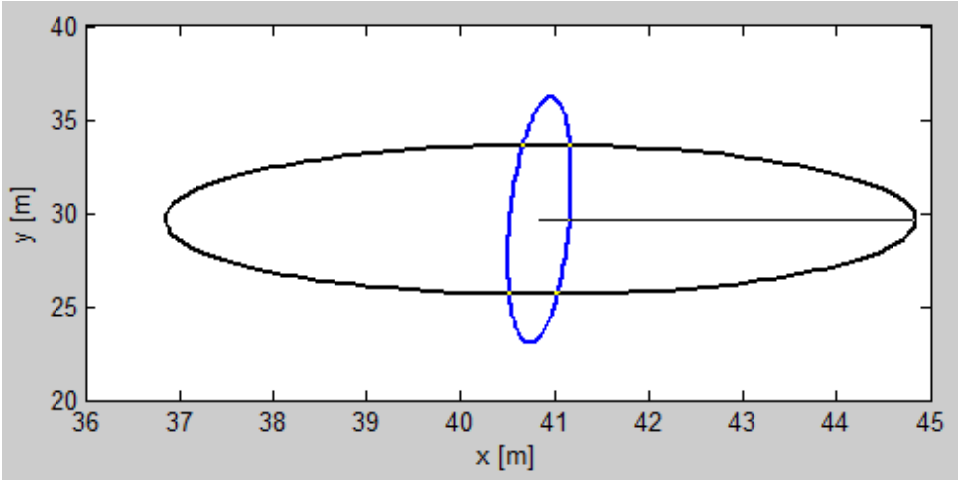
Наиме, када се подаци исчитају са сензора, слажу се у матрицу *''S''*. Подаци су вредности светлосног интензитета, који се разликују у зависности од боје подлоге. У нашем случају је бела подлога, па је светлосни интензитет износио приближно 600 (шест стотина). Реч, ''приближно'' је из тог разлога што услед промене светлости и сенке у просторији тај број се разликује, тј. није увек исти, из тог разлога на сцену ступају ВНМ, као што је речено у некој претходној тачки. Када сензор детектује контролну тачку црне боје, тада су вредности интензитета око 300 (три стотине).

Детектоване вредности се слажу у матрицу, скалирају на вредности између -1 и 1, где је -1 за црну боју, а 1 за белу боју, а све између су њихове нијансе. Тако скалиране вредности се провлаче кроз вештачку неуронску мрежу (ВНМ), која у овом случају има назив *"netff11",* види горе у коду, која је обучена и развијена за ову намену. У зависности од излаза мреже, тачније ако се детектује црна боја, излаз мањи од нуле, врши се корекција кретања преко функције чији је код приказан испод, тако што се за израчунати положај узима положај контролне тачке са највећом вероватноћом да буде детектована. Овај поступак је спроведен помоћу Калмановог филтера који је у претходним тачкама детаљно објашњен. Подфункција из управљачког кода је приказана испод.

```
function [x, C]=ITS_data_association(x, C, Q, map)

xls = 5.25
yls = 0
N = []
e = 1

for k = 1 : size(map,1)
%     na ovom mestu (umesto predict_measurements_with_lihgt_sensors() )
% stavite vasu funkciju koja predvidja merenja
    [zp,Hp] = predict_measurements_with_lihgt_sensors(x,map(k,:), xls, yls)
    z = map(k,:)'
    S= Hp*C*Hp' + Q;
    %     very very very simple association....
    nis= (z-zp)'*inv(S)*(z-zp);
    N = [N;nis]

   min(N)

for j = 1 : size(N,1)
    if N(j)==min(N)
        j
    end
end
end

% j = find(N<=e)
if isempty(j)
    return
else
    Z = map(j,:)'
    [x, C] = EKF_update_light_sensors(x, C, z, zp, Hp, Q)

    disp('korigovan sam')
end
```

Овај код за детектовање контролних тачака се налази на неколико места у управљачком коду и то у свим фазама управљања кретањем. Тако да се после сваког померања робота и израчунавања тренутне позиције, проверавају контролне тачке ради правовремене корекције израчунатог положаја.

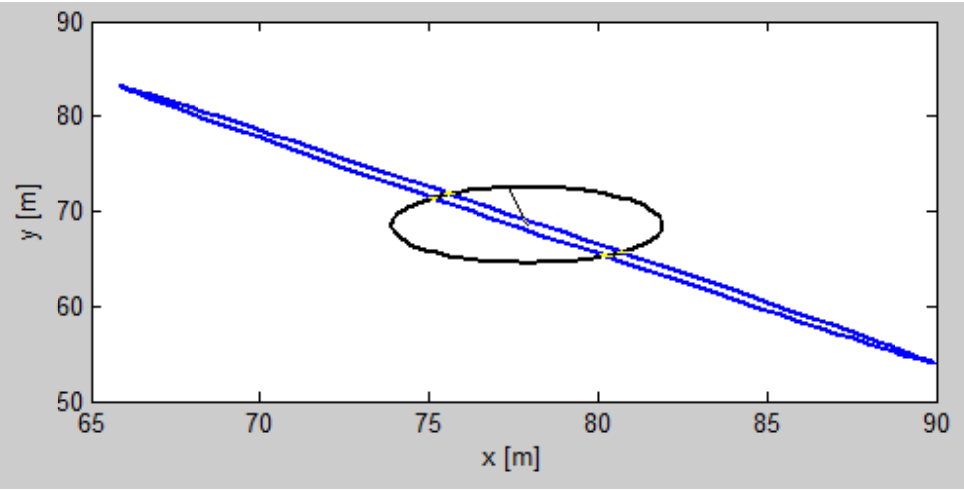
У току кретања робота управљачки алгоритам нам стално даје на увид графички приказ сигурности да се робот налази у израчунатом положају. Ова сигурност опада у току кретања све док се не изврши локализација, односно док се не детектује контролна тачка. На слици 10.5 се види графички приказ сигурности робота након извршене локализације, када је детектована црна тачка, тј. маркер.



Слика 10.5: Графички приказ сигурности мобилног робота после урађене локализације

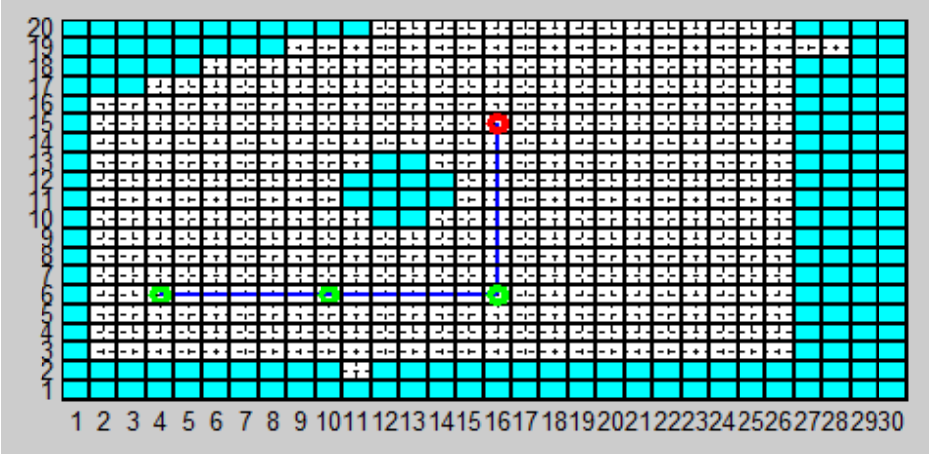
Са слике се види да је смањена плава елипса тј. повећана је сигурност робота да се налази баш у положају који је тренутно израчунат.

Након што је робот урадио сва потребна кретања према претходно дефинисаном поступку и нашао се у међутачки, која је дефинисана поставком, а која је за прву интерацију била циљна, тада управљачки алгоритам циљну тачку из претходне интерације проглашава за почетну, а нову тачку, која је дефинисана на почетку у матрици *“stoptacke”* за циљну, под условом да постоји. Тада се цео поступак налажења оптималне путање и покретања робота понавља, све док робот не дође у крајњу тачку дефинисану на почетку.



Слика 10.6: Графички приказ сигурности мобилног робота, када се налази у израчунатом положају након завршетка кретања

Мобилни робот наставља кретање ка крајњој тачки и успешно извршава постављени задатак. На слици 10.6. је приказан график сигурности на крају кретања, а на слици 10.7. приказана је целокупна путања робота према горе наведеном примеру.



Слика 10.7: Целокупна путања робота према датом примеру

Сигурност мобилног робота на крају кретања је смањена, слика 10.6., јер није вршио локализацију до краја кретања, видимо његову позицију, x, y координате и оријентацију. Када се те координате упореде са координатама крајње тачке, добија се резултат након кога се може констатовати да је мобилни робот успешно прошао постављени задатак у конкретном примеру.

11. Закључак

У овом пројектном раду је анализирана и експериментално доказана могућност примене интелигентног мобилног робота за послове унутрашњег транспорта материјала и делова у анализираном предузећу, тј. технолошком систему. За ове потребе је развијен алгоритам у софтверском пакету *"MATLAB",* којим се управља кретањем робота. Управљачки алгоритам или код се заснива на примени модела кретања, Калмановог филтера, развијених модела вештачких неуронских мрежа и A* алгоритма.

На основу модела кретања и прикупљања информација са сензора и њиховом класификацијом помоћу вештачких неуронских мрежа обезбеђено је кретање робота.

Калманов филтер имао је главну улогу у отклањању неусаглашености између остварене и израчунате позиције, где се врши корекција положаја.

Применом A* алгоритма генерисана је најкраћа путања између задатих почетних и циљних тачака, као и функција избегавања препрека за дато окружење.

Да би се омогућило управљање овим роботом односно очитавање сигнала са сензора на роботу и слање контролних акција, коришћен је RWTH Toolbox [8] у софтверском пакету MATLAB, где је развијен код за управљање одговарајућег кретања мобилног робота.

Резултат рада представља, робота који је у стању да се оријентише и обави одговарајуће кретање на основу ознака на подлози.

Може се закључити да програмирани робот са задовољавајућом тачношћу изводи задато кретање. Тачност која је постигнута се налази у толерисаним границама. Разлог појаве размимоилажења рачунске вредности која је добијене коришћеним програмом и позиције и оријентације робота је пре свега заснована на тачности израде самог робота. Међутим, уколико се користи други робот састављен од квалитетнијих компонената, тада би се остварена позиција више поклапала са рачунским вредностима које су добијене у *MATLAB*-у.

Симулација у софтверском пакету *AnyLogic* омогућује сагледавање и анализу призводних токова унутар датог технолошког окружења, где за одређене параметре (времена транспортног материјала и обраде припремка, капацитети бафера и конвејера, брзина кретања конвејера) било је потребно оптимизовати како би се повећала продуктивност и производност производа.

12. Литература

[1] Милутиновић, Д., Предавања на предмету Индустријски роботи, Универзитет у Београду – Машински факултет, Београд, 2009.

[2] Миљковић,З., Бабић, Б, Изводи са предавања и вежби на предмету Интелигентни технолошки системи (ПРО220-0131), Универзитет у Београду – Машински факултет, Београд, 2010.

[3] Миљковић,З., *Системи вештачких неуронских мрежа у производним технологијама*, Серија монографских дела Интелигентни технолошки системи (Уредник серије: Проф. др Владимир Милачић), Књига 8, Универзитет у Београду - Машински факултет, Београд, 2003.

[4] Миљковић, З., Предавања на предмету Интелигентни технолошки системи (ПРО220-0131), Универзитет у Београду – Машински факултет, Београд, 2009.

[5] Петровић П., Предавања на предмету Мехатронски системи (ПРО), Универзитет у Београду Машински факултет, Београд, 2009.

[6] http://shop.lego.com/, датум последњег приступа 23.03.2011.

[7] http://mfkg.kg.ac.rs/weblab/vezbe/19/Primena%20LEGO%20robota%20u%20inzenjerskoj%20edukaciji.pdf/, датум последњег приступа 23.03.2011.

[8] http://www.mindstorms.rwth-aachen.de/, датум последњег приступа 25.03.2011.

[9] http://www.mathworks.com/products/matlab/, датум последњег приступа 08.04.2011.

[10] http://www.fmp.rs, датум последњег приступа 05.04.2011.