Универзитет у Београду
Машински факултет

Дипломске 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: др Божица Бојовић, Најдан Вуковић, дипл. маш.инж. Марко Митић, дипл. маш.инж. Милица Петровић, дипл. маш.инж.
Потпис наставника:	Група:

РБ	Презиме и име:	Бр.инд.	Потпис:
1.	Аранђеловић Синиша	1032/09	
2.	Бундало Мирослав	1186/09	
3.	Најдески Борис	1187/09	
4.	Петровић Златко	1191/09	

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, mmitic@mas.bg.ac.rs I mmpetrovic@mas.bg.ac.rs.

Датум издавања задатка:

Рок завршетка задатка:

Задатак издао: _____

(Најдан Вуковић)

Синиша Аранђеловић[1], Мирослав Бундало[2], Борис Најдески[3], Златко Петровић[4]

Резиме

У овом елаборату приказана је примена интелигентних технолошких система у домену производних технологија. Детаљно је пројектован и разрађен систем мобилног робота који за циљ свог рада има опслуживање машина у производном погону, омогућавање одвијања технолошког процеса и обезбеђивање веће продуктивности и поузданости самог производног система. На тај начин је створена могућност замене досадашњих транспортних система као што су виљушкар и слични ручни транспортни системи. За решење проблема транспорта предложен је едукациони мобилни робот, а верификација концепцијског решења извршена у лабораторијском моделу технолошког окружења. Управљачки код је написан у MatLab софтверском пакету. Софтверским алатом AnyLogic приказана је могућност пројектовања диспозиционог плана производног система помоћу прикупљених података о технолошком процесу производног система, врстама и наменама машина као и времена појединачних операција у току производње. То је рађено у циљу побољшања квалитета пројектовања производног предузећа и скраћења времена пројектовања.

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

[1] **Синиша Аранђеловић 1032/09**, Универзитет у Београду – Машински факултет, студент друге године Дипломских академских студија.
Е-пошта: sinisa.arandjelovic@gmail.com
[2] **Мирослав Бундало 1186/09**, Универзитет у Београду – Машински факултет, студент друге године Дипломских академских студија.
Е-пошта: miroslavbundalo@gmail.com
[3] **Борис Најдески 1187/09**, Универзитет у Београду – Машински факултет, студент друге године Дипломских академских студија.
Е-пошта: najdeski.boris@open.telekom.rs
[4] **Златко Петровић 1191/09**, Универзитет у Београду – Машински факултет, студент друге године Дипломских академских студија.
Е-пошта: zlaya1987@yahoo.com

Списак слика

Слика 1.1.1: Пример AGV колица..... 10

Слика 2.1.1: Делови пакета LEGO Mindstorms NXT..... 13

Слика 2.1.2: Контролна јединица пакета LEGO Mindstorms NXT.....13

Слика 2.1.3: Архитектура контролне јединице пакета LEGO Mindstorms NXT..... 14

Слика 2.1.1.1: Изглед конфигурисаног мобилног робота..... 14

Слика 2.1.1.2: Приказ спајања мотора..... 15

Слика 2.1.1.3: Елементи за спајање контролне јединице.....15

Слика 2.1.1.4: Приказ задњег точка.....16

Слика 2.1.1.5: Приказ подсклопа и везе светлосног сензора.....16

Слика 2.1.1.6: Маркер за приказ осе робота.....16

Слика 2.1.2.1: Архитектура контролне јединице пакета LEGO Mindstorms NXT.....17

Слика 2.1.3.1: Принцип рада светлосног сензора.....18

Слика 2.2.1: Позиција и оријентација у Декартовом координатном систему.....19

Слика 2.2.2: Брзински модел кретања..... 19

Слика 2.2.3: Пређени пут сваког од точкова.....20

Слика 2.2.4: Позиција и оријентација у предходном и садашњем тренутку посматрања..... 21

Слика 2.3.1: Примери нормалних распореда у зависности од скаларних параметара..... 22

Слика 2.3.2: Пример нормалне расподеле за вектор x=[X,Y] и параметре μ=[0 0], Σ=[1 0.4; 0.4 1].....23

Слика 2.3.3: Нормална расподела за вектор x=[X,Y] у равни X,Y..... 23

Слика 2.4.1: Неурон са својим окружењем..... 28

Слика 2.4.2: Основни елементи неурона.....29

Слика 2.4.3: Архитектура BP вештачке неуронске мреже…….. 30

Слика 2.5.1: Четвороспојиви и осмоспојиви пиксели…… 31

Слика 2.5.2: Еуклидска и Хеуристичка норма…… 31

Слика 2.5.1.1: Примери матрица 3x3: а) препрека, б) h, в) g, г) пређеног пута и д) посећених тачака……..32

Слика 2.5.1.2: Први корак при налажењу најкраће путање……32

Слика 2.5.1.3: Други корак при налажењу најкраће путање……33

Слика 2.5.1.4: Трећи корак при налажењу најкраће путање……...33

Слика 2.5.1.5: Последњи корак при налажењу најкраће путање……...34

Слика 2.5.1.1.1: Пинцип програмирања h матрице……36

Слика 2.5.1.3.1: Графички приказ излаза из алгоритма А звезда…… 41

Слика 2.5.2.1: Разлика између алгоритма А звезда и Дајкстра…… 42

Слика 2.5.2.2: Прва врста проблема уколико се не дефинише матрица посећених тачака - први корак……43

Слика 2.5.2.3: Прва врста проблема уколико се ne дефинише матрица посећених тачака - други корак…… 43

Слика 2.5.2.4: Прва врста проблема уколико се не дефинише матрица посећених тачака – графички приказ..... 44

Слика 2.5.2.5: Друга врста проблема уколико се не дефинише матрица посећених тачака - први корак..... 45

Слика 2.5.2.5: Друга врста проблема уколико се не дефинише матрица посећених тачака - други корак..... 45

Слика 2.5.2.6: Друга врста проблема уколико се не дефинише матрица посећених тачака - трећи корак..... 46

Слика 2.5.2.7: Друга врста проблема уколико се не дефинише матрица посећених тачака – графички приказ.....46

Слика 2.5.2.8: Рачун по избацивању сувишних параметара – први корак.....47

Слика 2.5.2.9: Рачун по избацивању сувишних параметара – други корак..... 47

Слика 2.5.2.10: Рачун по избацивању сувишних параметара – трећи корак..... 48

Слика 2.5.2.11: Рачун по избацивању сувишних параметара – четврти корак.....48

Слика 2.5.2.12: Хеуристички приступ без дискретизације..... 49

Слика 2.5.2.12: Хеуристички приступ без дискретизације – уколико се не избаци матрица g..... 49

Слика 2.6.3.1: Концепт примене ВНМ за одређивање вредности угла рорације излазног вратила мотора.....52

Слика 2.6.3.2: Нелинеарност обучавајућег вектора.....52

Слика 2.6.3.3: а) добри и б) лоши резултати генерисаних ВНМ..... 55

Слика 2.6.3.4: Регресија одабране ВНМ..... 56

Слика 2.6.4.1: Контролне тачке за кориговање позиције робота..... 57

Слика 2.6.4.2: алгоритам препознавања контролних тачака..... 57

Слика 2.6.4.3: обучавајући парови за вештачку неуронску мрежу..... 58

Слика 2.6.4.4: Конвергенција ВНМ ка решењу..... 60

Слика 2.6.6.1: Програмски дијаграм..... 63

Слика 2.6.6.2: Алгоритам главног програма управљања..... 64

Слика 2.7.1: Диспозициони план симулираног окружења..... 65

Слика 2.7.2: Симулирано радно окружење производног погона.....66

Слика 2.7.3: Изведена путања мобилног робота..... 66

Слика 2.7.4: Прикази мобилног робота током кретања..... 67

Слика 2.8.1: Остварене циљне позиције мобилног робота..... 69

Слика 3.1.1: Диспозициони план прдузећа задатог задатком..... 72

Слика 3.1.1: Симулациони модел производног тока..... 73

Слика 3.1.2: Симулациони модел на крају тока симулације..... 73

Слика 3.2.1: Троугласта матрица пројектовања.....74

Слика 3.2.2: Табеле критеријума и разлога веза између машина..... 74

Слика 3.2.3: Симулациони модел производног тока након измене..... 75

Слика 3.2.4: Симулациони модел на крају симулације након прве измене..... 75

Слика 3.2.5: Симулациони модел производног тока на крају симулације након коначних измена.....76

Списак табела

Табела 2.1.2.1: Преглед позиција мотора..... 17

Табели 2.3.1.1: *Алгоритам* линеаризованог Калмановог филтера локализације..... 27

Табела 2.5.2.1: Команде за контролу кретања мобилног робота.....51

Табела 2.5.2.2*:* Команде за комуникацију рачунара и мобилног робота..... 51

Табла 2.6.3.1: Обучавајући парови за ВНМ..... 53

Табела 2.6.3.2: Обучаване мреже за заокретање робота..... 54

Табела 2.6.3.3: Вредности параметара одабране мреже за обучаване вредности..... 55

Табела 2.6.4.1: Обучаване мреже за детекцију контролних тачака.....59

Тебела 2.6.4.2: Упоредне вреднсти очеиваних резултата и резултата генерисаних из ВНМ..... 60

Табела 3.1.1: Трајање технолошких поступака за изабреани део из производног асортимана..... 71

Табела 3.1.2: Списак и опис машина у проижводној линији.....72

Садржај

1. Увод..... **10**

1.1. Поставка проблема прве целине пројектног задатка..... **10**

1.2. Поставка проблема друге целине пројектног задатка..... **11**

2. Мобилни робот..... **12**

2.1. Конструкција пројектованог мобилног робота..... **13**

2.1.1. Механички систем моблиног робота..... **14**

2.1.2. Актуациони систем мобилног робота..... **17**

2.1.3. Сензорски систем мобилног робота..... **17**

2.2. Модел кретања пројектованог мобилног робота...... **18**

2.3. Калманов филтер..... **21**

2.3.1. Примена калмановог филтера..... **25**

2.4. Вештачке неуронске мреже..... **28**

2.5. Пројектовани *A алгоритам претраге**.....**30**

2.5.1. Пројектовање алгоритма у МАТЛАБ окружењу..... **32**

2.5.1.1. Подпрограм h матрица..... **36**

2.5.1.2. Пређени пут и нова позиција..... **39**

2.5.1.3. Подпрограм цртеж..... **40**

2.5.2. Дискусија и закључак..... **42**

2.6. Имплементација пројектованих подсистема у пројектном решењу мобилног робота..... **50**

2.6.1. Управљачки систем мобилног робота..... **50**

2.6.2. Управљање кретањем мобилног робота..... **50**

2.6.3. Обучавање окретања мобилног робота помоћу вештачке неуронске мреже.. **52**

2.6.4. Обучавање светлосног сензора помоћу вечтачке неуронске мреже..... **56**

2.6.5. Обрада података са светлосног сензора и корекција позиције робота..... **61**

2.6.6. Потпуни програмски код за управљање мобилним роботом..... **63**

2.7. Верификација резултата пројектног решења мобилног робота **65**

2.8. Дискусија и закључак **68**

3. Технолошко окружење производног предузећа **70**

3.1. Симулациони модел у *Anylogic* окружењу **71**

3.2. Пројектовање новог диспозиционог плана технолошког окружења **74**

3.3. Нови предложени диспозициони план технолошког окружења **75**

3.4. Дискусија и закључак **77**

4. Литература **78**

1. Увод

Овај елаборат је израђен у оквиру студентског пројекта на предмету Интелигентни технолошки системи, на другој години дипломских академских студија. Пројекат и сам елаборат се састоји из две целине.

Прва целина пројекта и елабората обухвата конфигурисање мобилног робота са свим својим подсистемима у које се убрајају механички, актуациони и сензорски подсистем. Такође, овај део пројекта садржи разраду и писање програмског кода у MatLAB окружењу. Овај програм управља кретањем мобилног робота са елементима интелигентног понашања на основном нивоу, што и јесте циљ студентског пројекта на предмету Интелигентни технолошки системи. Програмски код укључује обраду сензорских сигнала помоћу којих се извршавају одређене функције мобилног робота. У програм су имплементирани и одређени математички модели као што је Калманов филтер.

Друга целина пројекта и елабората обухвата пројектовање диспозиционог плана производног предузећа на основу прикупљених података о технолошком процесу производње, временима технолошких операција и врста машина које се за то користе. Предложен је нови диспозициони план на основу резултата симулације у циљу усавршавања датог, већ постојећег диспозиционог плана предузећа. За то је коришћен савремени софтвер AnyLogic за симулацију технолошког процеса израде производа.

1.1. Поставка проблема прве целине пројектног задатка

Као што је већ речено, у првој целини пројектног задатка потребно је пројектовати систем мобилног робота чија је намена транспорт материјала, делова, алата и специјалних уређаја у оквиру производног предузећа. Мобилни роботи се данас користе у индустрији за опслуживање машина и магацина највише у случају флексибилних технолошких система. Они су често конципирани као системи који се крећу по шинама или вођицама, а пошто су флексибилни технолошки системи најчешће линијске структуре, таква концепција мобилних робота задовољава потребе. Такође постоје мобилни роботи који су познати као AGV – Automated Guided Vehicle. Овакви роботи поседују већи степен аутономности и представљају предмет интензивног истраживања. На слици 1.1 дат је пример AGV колица.

Слика 1.1.1: Пример AGV колица

Мобилни робот о коме ће бити речи у овом елаборату је едукационог карактера. Из ток разлога за конструисање мобилног робота узет је пакет LEGO Mindstorms NXT. Помоћу тог пакета потребно је конфигурисати робот са механичким, актуационим и сензорским системом који морају бити спрегнути са програмом за управљање са елементима интелигентног понашања како би се разматрали нови аспекти мобилних робота који се могу користити у индустрији.

У оквиру програма за управљање мобилнијм роботом потребно је развити алгоритам претраживања на бази тзв. алгоритма „А звезда“. Поред тога потребно је имплементирати математички модел Калмановог филтера којим се коригује тренутна позиција робота и модел кретања мобилног робота.

Систем мобилног робота треба да се креће у симулираном лабораторијском моделу технолошког окружења са распоређеним машинама према постојећем диспозиционом плану.

1.2. Поставка проблема друге целине пројектног задатка

У другој целини пројекта изложено је постојеће решење пројектованог диспозиционог плана предузећа. На основу података о технолошком процесу производње, временима технолошких операција и типовима и распореду машина, потребно је предложити и пројектовати ново решење диспозиционог плана предузећа.

Пројектовање диспозиционог плана представља битан процес при пројектовању новог производног погона. То је процес пројектовања везан је за врсту машина које ће бити инсталиране у погону, врсту технолошких процеса који се појављују у погону, врсту производа, обим производње, врсту делова , саму величину расположивог простора као и за многе друге параметре производног предузећа.

Врло битну ставку пројектовања диспозиционог плана производног предузећа представља симулација технолошког процеса. За то је потребно познавати времена појединих операција као и њихову расподелу узорака. На основу ових података може се симулирати процес у циљу преиспитивања како постојећих процеса тако и процеса који се пројектују. При томе се обично користе савремени рачунарски алати што и јесте задатак овог дела пројекта и елебората.

Помоћу симулационог софтвера AnyLogic направљена је симулација технолошког процеса према познатим подацима технолошких операција. На основу те симулације могуће је установити где је потребно извршити корекције процеса и што је најбитније може се установити где долази до губитка у времену производње што је од великог значаја у масовном типу производње.

Као основ за пројектовање диспозиционог плана производног предузећа показано је коришћење троугаоне матрице пројектовања која приказује квалитативну међузависност машина у погону.

2. Мобилни робот

Пре разраде овог поглавља, потребно је дефинисати интелигентне технолошке системе и мобилне роботе са интелигентним понашањем.

Интелигентни технолошки системи су највиша класа данас познатих технолошких система. Свој највећи развој доживели су у последње две деценије првенствено због великог развоја рачунарских технологија и компјутера посебно. Тиме је могуће имплементирати у интелигентни систем његову основу рада, а то су комплексни алгоритми које систем користи за процесирање сензорских улазних информација уз подршку софистицираних компонената система које се називају агенти. Такође, поред обраде сензорских информација, интелигентни технолошки системи треба да их меморише како би успешност будућих одлука била што већа. У наставку је дата дефиниција интелигентних технолошких система према [1]:

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

Један од главних чиниоца интелигентних технолошких система је вештачка интелигенција (AI – Artificial Intelligence) што обухвата експертне системе и адаптивно процесирање. Компјутерски интегрисане технологије као што су CAD, CAM и CAPP исто имају кључну улогу у интелигентним технолошким системима.

Парадигме вештачке интелигенције налазе примену у разним софтверима као што су софтвери за израду технолошког поступка, софтвери за одлучивање, софтвери за управљање, али исто тако налазе примену у хардверским системима као што су роботи, на чију је тему и писан овај елаборат. Досадашњи индустријски роботи имају, поред своје одличне примене, недовољну флексибилност у раду и због тога се тежи ка вишем нивоу који представљају интелигентни технолошки системи. Дефиниција индустријског робота према RIA – Robotic Industries Association [12], дата је у наставку:

Индустријски робот је аутоматски управљан, репрограмабилни мултифункционални манипулатор, програмабилан са три или веше оса које могу бити или фиксиране у месту или мобилне, а користе се за потребе индустријске аутоматизације.

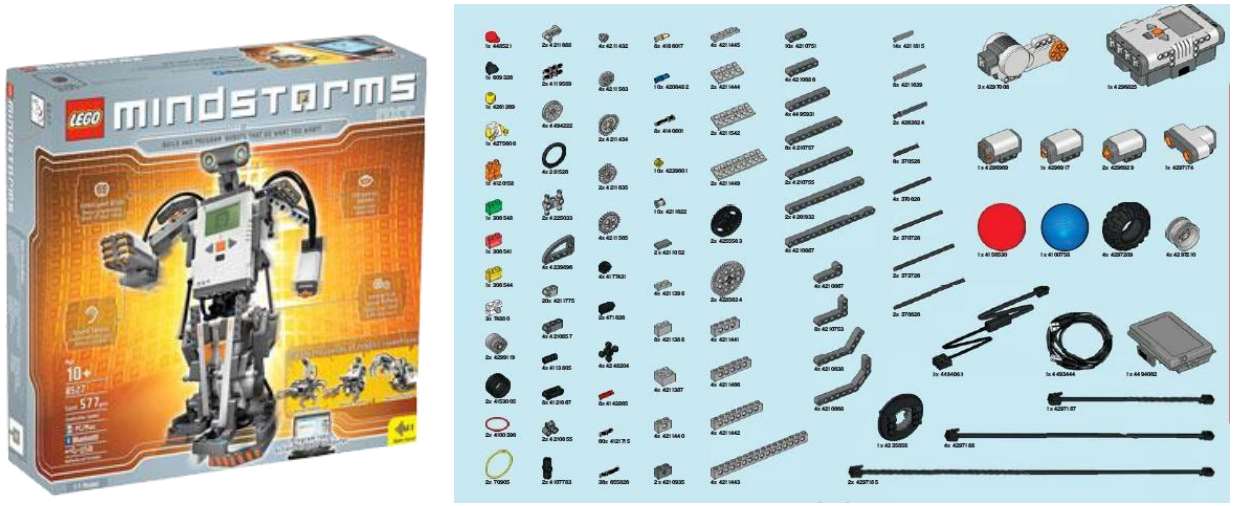
Да би се ова дефиниција могла уклопити у тему овог студентског пројекта и елабората недостају три кључна појма. Први појам представља појам мобилности. Мобилност је могућност кретања робота кроз простор према жељеном задатку. За тај појам уско су везани интелигенција и аутономност.

Аутономан робот је онај робот који може самостално да доноси закључке на основу свог окружења. Може се рећи да је такав робот на неки начин „свестан себе и окружења“ и ако је вештачки направљена машина.

Интелигенција је најбитнији појам међу недостајућим појмовима. Овај појам побољашњава потребну интеракцију робота са окружењем у циљу бољег извршавања задатка. Ово захтева сложене алгоритме којима се дефинише понашање.

2.1 Конструкција пројектованог мобилног робота

За конфигурисање мобилног робота изабран је пакет LEGO Mindstorms NXT. Овај пакет садржи велики број могућих комбинација конструкције робота и зато је врло погодан за ову прилику. Поред елемената који имају конструкциону улогу као што су елементи за везу, зупчаници, блокови и слично, у овом пакету се налазе и комплекснији елементи као што су електро мотори, сензори и контролна јединица. Поједини делови пакета дати су на слици 2.1.1.

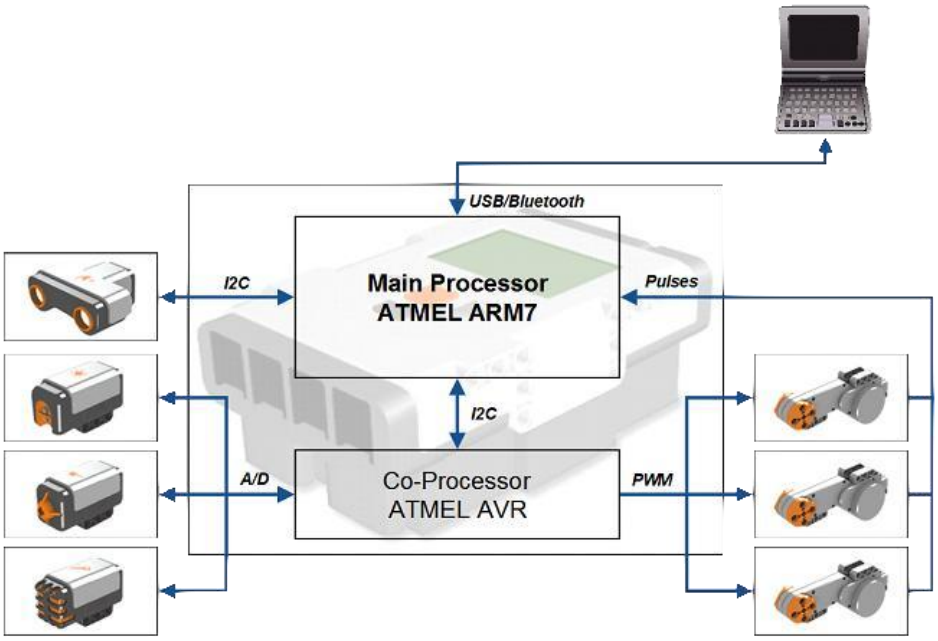


Слика 2.1.1: Делови пакета LEGO Mindstorms NXT

Контролна јединица овог пакета представља главни управљачки систем пакета. Помоћу ње се прикупљају подаци са мерних система мотора и подаци са сензора, да би се прослеђивали преко USB комуникационог протокола на PC рачунар који у овом случају представља главну управљачку јединицу целокупног система мобилног робота. Контролна јединица има на себи три прикључка за повезивање са моторима, четири прикључка за повезивање са сензорима и један прикључак за USB комуникацију са рачунаром. Поред тога контролна јединица садржи дисплеј и тастере за контролу рада система. Ова контролна јединица је дата на слици 2.1.2 где је дат и приказ свих елемената који се са њом повезују.

Слика 2.1.2: Контролна јединица пакета LEGO Mindstorms NXT

Архитектура контролне јединице је дата на слици 2.1.3 где се налазе и начини комуникације са сензорима, моторима и рачунаром. Контролна јединица се састоји од два процесора где је први главни а други помоћни процесор за обраду сензорских сигнала и управљање моторима.



Слика 2.1.3: Архитектура контролне јединице пакета LEGO Mindstorms NXT

2.1.1 Механички систем мобилног робота

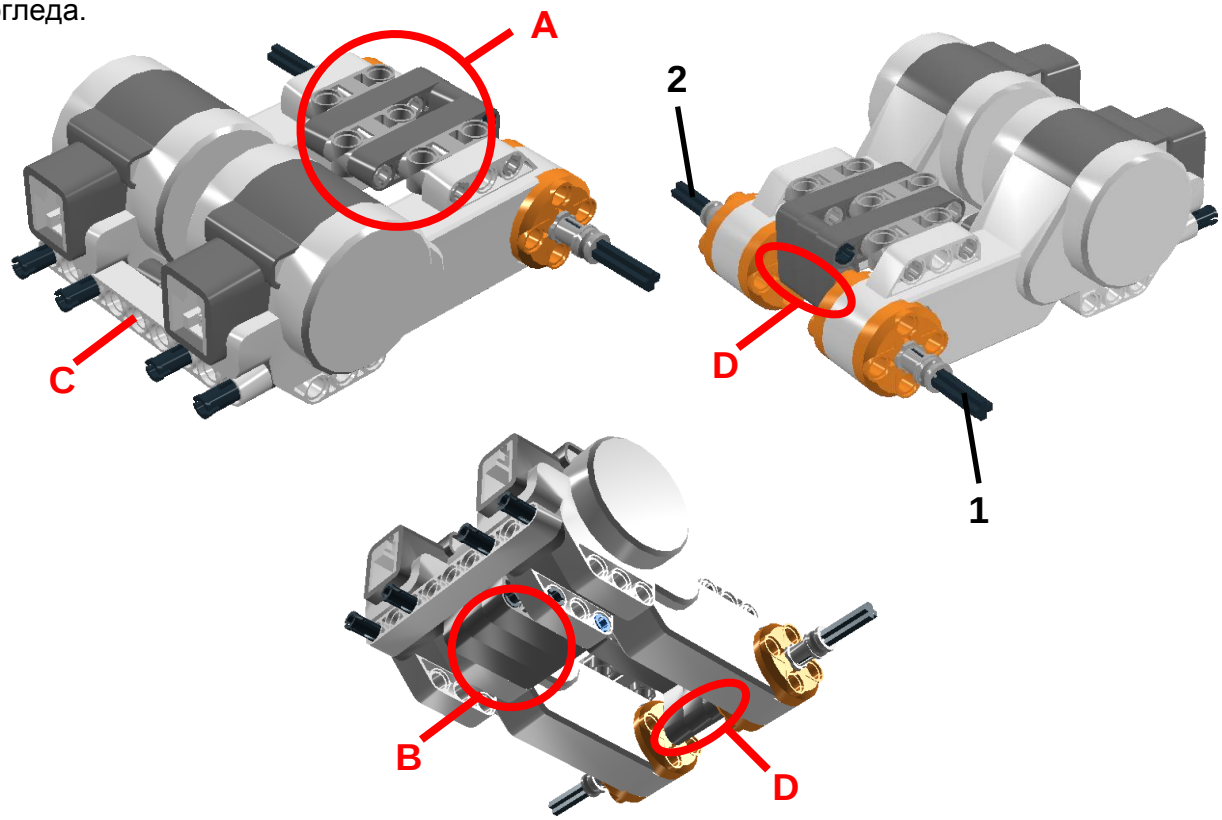
Мобилни робот је склопљен помоћу конструкционих елемената LEGO пакета. То су елементи за везу, блокови, точкови, гуме и сл. Такође, сам мотор је предвиђен као градивни елемент на који се могу везивати остали елементи. Основни функционални захтеви при конструисању робота били су:

1. постизање што веће крутости конструкције,
2. компактност конструкције и
3. што мање осно растојање точкова.

Изглед пројектованог мобилног робота дат је на слици 2.1.1.1

Слика 2.1.1.1: Изглед конфигурисаног мобилног робота

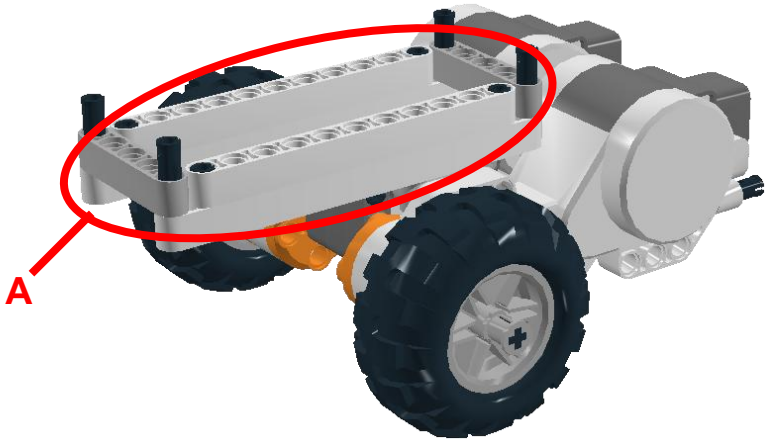
Два мотора, која независно покрећу леви и десни точак, спојени су међусобно помоћу свих везних места која на свом кућишту имају из разлога повећања крутости система. Такође спојени су на малом осном растојању како би точкови прелазили мањи пут приликом окретања робота. То је урађено првенствено због могућности акумулирања грешке при великим вредностима пређеног пута. Спојени мотори приказани су на слици 2.1.1.2. из више погледа.



Слика 2.1.1.2: Приказ спајања мотора

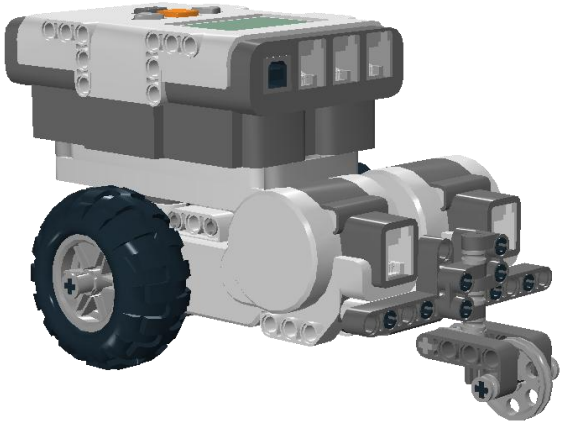
Мотори су у попречном правцу спојени са елементима у области A и B и укрућени су елементом C, што је показано на слици 2.1.1.2. Такође примећено је да се независне осовине точкова, које су означене са 1 и 2, угибају под дејством тежине робота. Разлог томе је недовољна крутост излазног елемента мотора који је наранџасте боје. У том циљу, осовине су провучене кроз отворе у деловима у области означеној са D, чиме се смањило угибање осовина. Ово нема пресудан утицај у конструкцији робота, али није занемарљиво.

Контролна јединица је додата на елементе који су означени са A на слици 2.1.1.3 чиме је омогућено лако скидање контролне јединице.



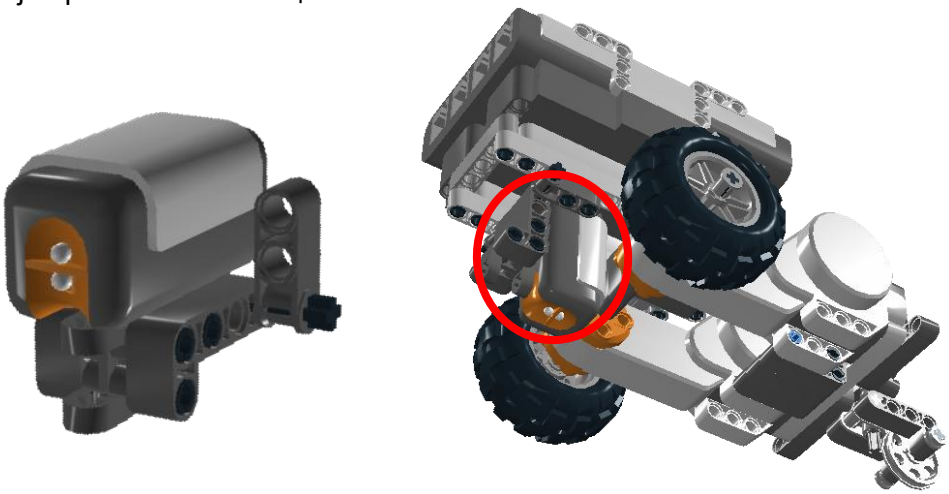
Слика 2.1.1.3: Елементи за спајање контролне јединице

Ради стабилности робота додат је трећи точак на задњем делу робота, који може да се окреће око вертикалне осе, чиме је омогућено несметано окретање робота у месту. Задњи точак је приказан на слици 2.1.1.4.



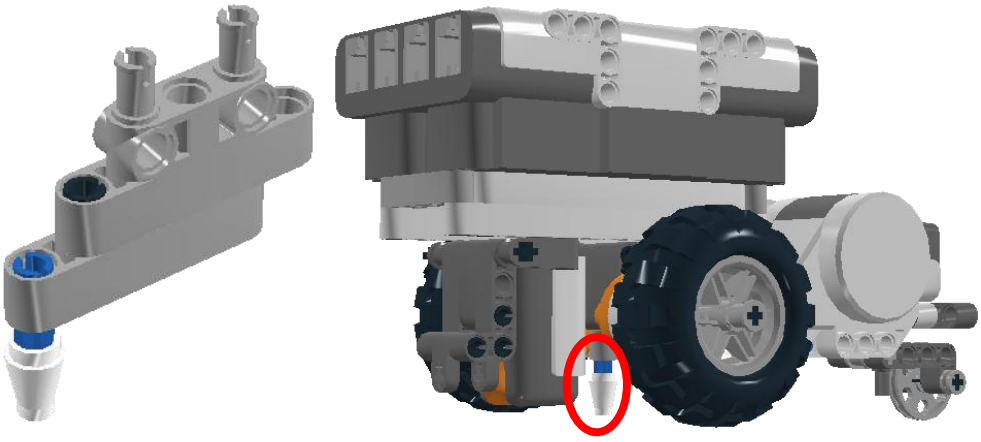
Слика 2.1.1.4: Приказ задњег точка

Светлосни сензор је помоћу везивних елемената стављен што ближе оси окретања робота као што је приказано на слици 2.1.1.5.



Слика 2.1.1.5: Приказ подсклопа и везе светлосног сензора

На крају је додат конусни део који показује осу око које се робот окреће. То је бели конусни део који је приказан на слици 2.1.1.6.

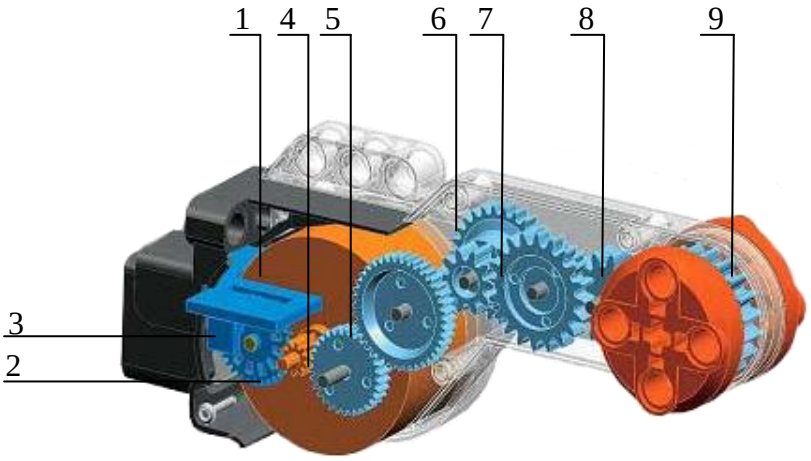


Слика 2.1.1.6: Маркер за приказ осе робота

2.1.2. Актуациони систем мобилног робота

Актуациони систем мобилног робота су електро мотори из LEGO пакета. Као што је већ речено за погон се користе два мотора као два независна погона левог и десног точка мобилног робота.

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 инкременатални 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. На слици 2.1.4 дат је пресек актуационог система са означеним позицијама. Обљашњења означених позиција дата су у табели 2.1.1.



Слика 2.1.2.1: Архитектура контролне јединице пакета LEGO Mindstorms NXT

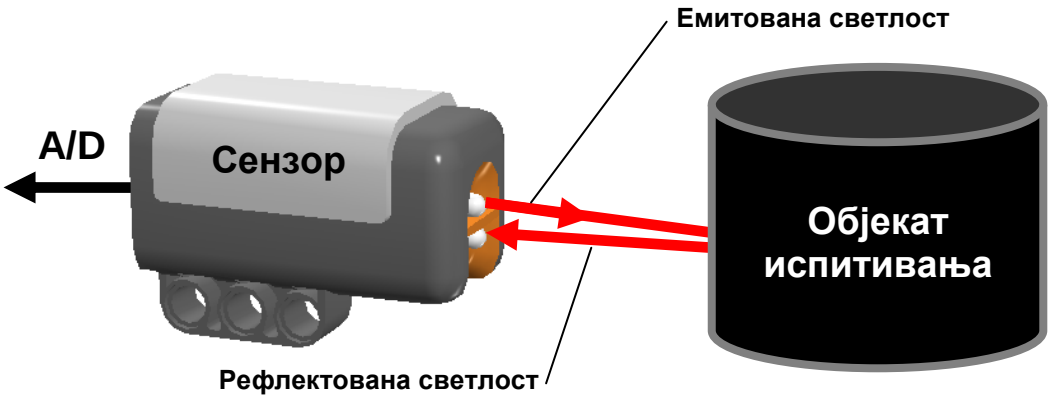
Табела 2.1.2.1: *Преглед позиција мотора*			
Позиција	Опис	Позиција	Опис
1	Електро мотор	6	Трећи зупчасти пар
2	Диск енкодера	7	Четврти зупчасти пар
3	Диода енкодера	8	Пети зупчасти пар
4	Први зупчасти пар	9	Шести зупчасти пар
5	Други зупчасти пар		

2.1.3. Сензорски систем мобилног робота

Поред енкодерског система који служи за давање информација повратној спрези контролера мотора, у овом пројектном решењу је коришћен и светлосни сензор из LEGO пакета. Основна идеја јесте да овај сензор детектује референтне тачке на појединим деловима радног окружења како би се кориговала тренутна позиција робота, али о томе ће речи бити касније.

Оваква врста сензора састоји се из кућишта у коме су смештене две светлосне диоде. Једна је одашиљач чија светлост је уперена ка објекту који се сензором испитује, док је друга диода фото осетљива диода. Друга диода се користи за примање рефлектоване светлости прве диоде о површину објекта који се испитује. Тиме друга диода на излазу даје

напон који се A/D конверзијом у контролној јединици претвара у одговарајућу дигиталну вредност. Шематски приказ рада овог сензора дат је дат на слици 2.4.1.



Слика 2.1.3.1: Принцип рада светлосног сензора

Ниске вредности A/D конверзије представљају црну боју, а високе вредности белу боју. Према томе све остале боје се налазе између интервала вредности црне и беле боје, сходно спектру боја.

Проблем у раду овог сензора понекад представља недовољна осветљеност површине испитивања, чиме се добија и ниста вредност очитавања, тј. вредност близу црне боје. То се може решити додавањем нове диоде која служи за осветљење што је и урађено у сензорима нових LEGO пакета.

2.2 Модел кретања пројектованог мобилног робота

Интелигентни робот у технолошком систему увек треба да може што тачније да одговори на питања “где сам?” и “куда идем?”. За одговоре на ова питања је неопходно математичко моделирање кретања робота у систему. Постоји много начина локализације, а овде ће детаљно бити описана два вида:

1. локализација помоћу брзинског модела кретања и
2. локализација помоћу модела кретања на основу пређеног пута.

Модел кретања се ослања на теоријске основе приказане у литератури [6]. Слободно тело (робот) у простору има шест степени слободе. Да би се знала његова тачна позиција и оријентација у простору, потребно је знати:

- X - позицију робота дуж x осе,
- Y - позицију робота дуж y осе,
- Z - позицију робота дуж z осе,
- θ - pitch, угао пропињања робота (обртање око X осе),
- φ – roll, угао ваљања робота (обртање око Y осе) и
- ψ – jaw, угао скретања робота (обртање око Z осе),

Имајући у виду претходно речено, тачан положај робота је функција шест координата, тј. увек важи:

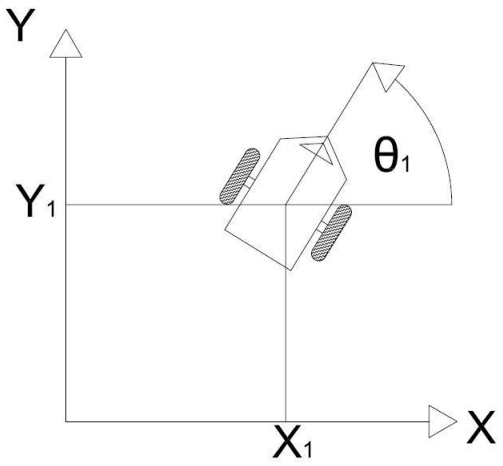
$$x_t = f(x, y, z, \theta, \psi, \varphi) \qquad (2.2.1)$$

Равански проблем доноси поједностављења. Тако, три координате постају нерелавантне, односно решења њихових једначина су тривијална или константе. Једначина (2.2.1) постаје (2.2.2). Овакав модел се може приказати у Декартовом раванском систему (слика 2.2.1), где:

- X1 представља позицију робота дуж x осе
- Y1 представља позицију робота дуж y осе
- θ1 представља угао који робот захвата са X осом,

и према томе математички модел постаје:

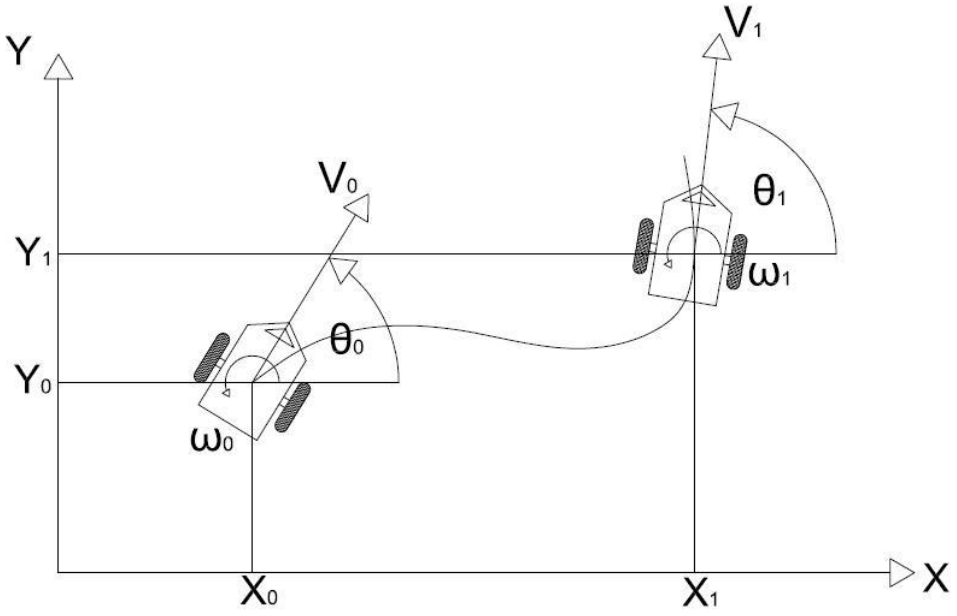
$$x_t = f(x_1, y_1, \theta_1). \qquad (2.2.2)$$



Слика 2.2.1: Позиција и оријентација у Декартовом координатном систему

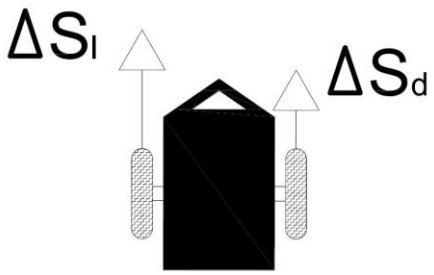
Робот ће увек знати да одговори на питање "где сам?" ако у сваком тренутку може да одреди вредности X_1 і Y_1, (представљају позицију робота) као и вредност θ_1 (представља оријентацију). Ако је у сваком тренутку могуће одредити први извод вектора X_t, могуће је користити брзински модел.

Брзински модел се базира на промени пројекција брзине центра маса робота на осе координатног система. Брзине сваког точка посебно се понашају као две силе које праве спрег, па се појављује вредност угаоне брзине заокретања робота (слика 2.2.2). Тако, вектор управљања у моделу брзине је скуп брзина сваког точка посебно.



Слика 2.2.2: Брзински модел кретања

У недостаку сезора брзине за брзине точкова (мотора који их погоне), имплементиран је модел кретања на основу пређеног пута – одометрија (енг. *odometry*). Одометрија се заснива на подацима о пређеном путу сваког точка посебно. Мотори који покрећу робота имају енкодере са којих је могуће у сваком тренутку очитати вредност за коју су се окренули. Када се узме вреднст заокретања мотора, могуће је одредити пређени пут точка који одговарајући мотор покреће. На тај начин добијамо вредности пређеног пута за сваки од точкова (слика 2.2.3).



Слика 2.2.3: Пређени пут сваког од точкова

Прираштај лучне координате се рачуна по формули:

$$\Delta s = \frac{\Delta s_l + \Delta s_d}{2}, \tag{2.2.3}$$

а прираштај угла по формули:

$$\Delta\theta = \frac{\Delta s_l - \Delta s_d}{2b} \tag{2.2.4}$$

где Δs_l и Δs_d представљају померање левог и десног точка респективно, а b представља растојање између точкова.

Тренутна позиција и оријентација робота представља збир претходно познате позиције и оријентације и прираштаја лучне координате и угла (2.2.5).

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \Delta s \cdot \cos\left(\theta + \frac{\Delta\theta}{2}\right) \\ \Delta s \cdot \sin\left(\theta + \frac{\Delta\theta}{2}\right) \\ \Delta\theta \end{bmatrix} \tag{2.2.5}$$

Када заменимо одговаруће вредности, добија се коначна једначина за модел кретања преко пређених путева за сваки од точкова.

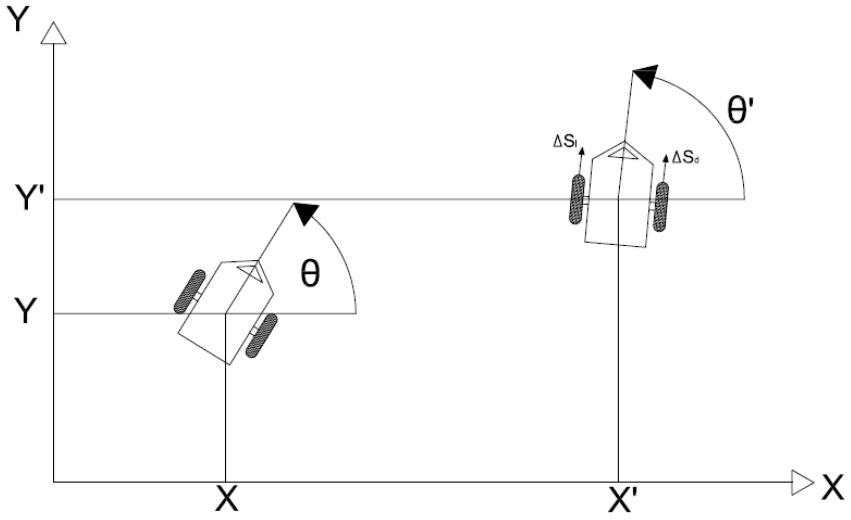
$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{\Delta s_l + \Delta s_d}{2} \cdot \cos\left(\theta + \frac{\Delta s_l - \Delta s_d}{2b}\right) \\ \frac{\Delta s_l + \Delta s_d}{2} \cdot \sin\left(\theta + \frac{\Delta s_l - \Delta s_d}{2b}\right) \\ \frac{\Delta s_l - \Delta s_d}{2b} \end{bmatrix} \tag{2.2.6}$$

Према (2.2.6), вредности координата вектора стања рачунају по једначини

$$x' = x + \frac{\Delta s_l + \Delta s_d}{2} \cdot \cos\left(\theta + \frac{\Delta s_l - \Delta s_d}{2b}\right)$$
$$y' = y + \frac{\Delta s_l + \Delta s_d}{2} \cdot \sin\left(\theta + \frac{\Delta s_l - \Delta s_d}{2b}\right) \qquad (2.2.7)$$
$$\theta' = \theta + \frac{\Delta s_l - \Delta s_d}{2b}$$

,

где Δs_l и Δs_d представљају померање левог и десног точка респективно, вредности x', y' представљају тренутну позицију робота у Декартовом координатном систему, θ' тренутну оријентацију (угао који робот заклапа са x осом), а вредности x, y и θ су одговарајуће вредности у претходном тренутку (слика 2.2.4).



Слика 2.2.4: Позиција и оријентација у предходном и садашњем тренутку посматрања

2.3. Калманов филтер

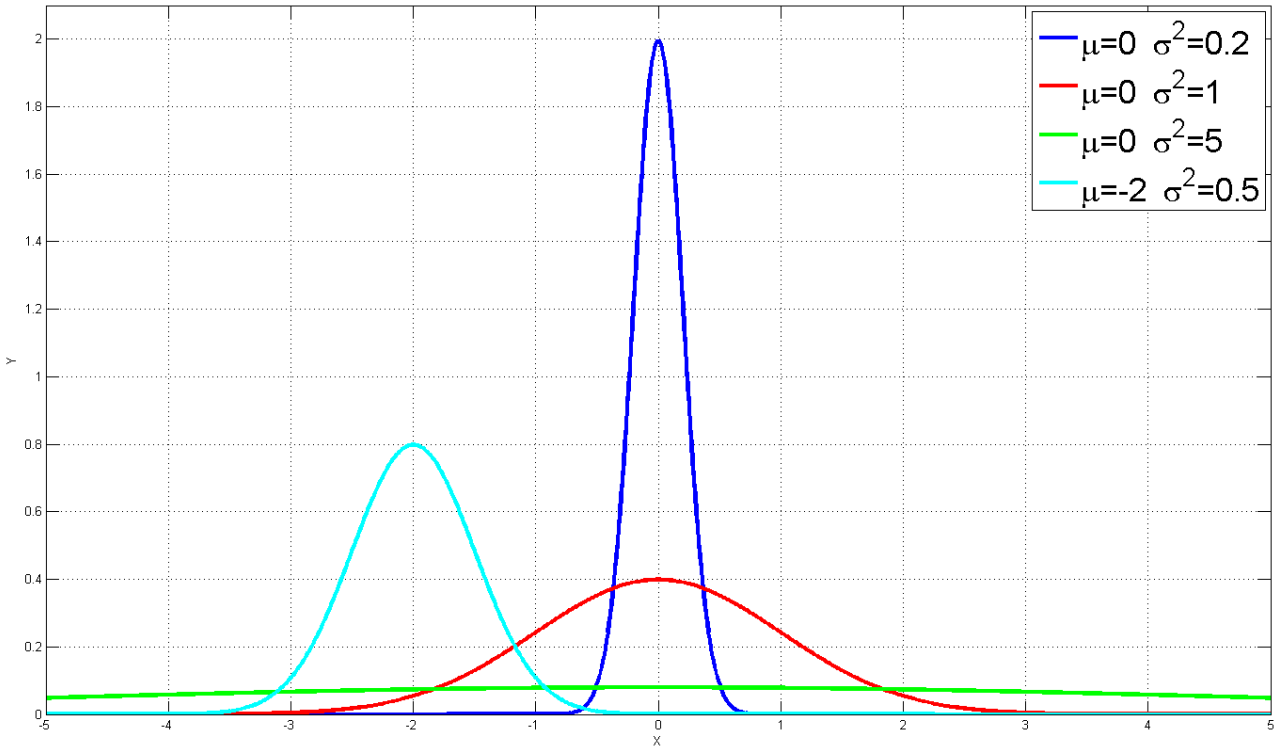
Калманов филтер је добио назив по свом творцу Rudolf E. Kalman. Реч је о математичком моделу који се позива на статистику. Његова намена је да се од измерених вредности које садрже шумове и разне друге узроке нетачности, добију вредности које теже стварној вредности мерења. Основна идеја рада овог математичког модела заснива се на неколико појмова:

- *Филтрација* – процес одређивања удела сигнала и удела шума у некој измереној информацији.
- *Интерполација* – процес прикупљања што више валидних информација о неком систему или процесу.
- *Предикција* – процес што бољег предвиђања стања неког система или процеса у наредном посматраном тренутку времена, а на темељу прикупљених података до тренутне вредности времена.
- Корекција – процес корекције стања неког система или процеса по преласку у наредно стање у времену, а на основу неких познатих измерених и рачунских вредности.

Калмановим филтером се предпоставља да је процес или систем који се посматра временски дискретан. Због тога се може говорити о предходним, садашњим и наредним тренутцима стања система или процеса.

Већ је речено да се у математички модел овог филтера ослања на законе статистике, па је стога потребно поменути и основе статистике. Калманов филтер предпоставља да вероватноћа неке променљиве подлеже нормалном распореду који се описује са две вредности и то μ и σ – стандардна девијација. На слици 2.3.1 дат је приказ различитих врста нормалне расподеле променљиве x, а у зависности од параметара расподеле. То је случај за скаларне променљиве:

$$p(x) \sim N(\mu, \sigma^2) \tag{2.3.1}$$

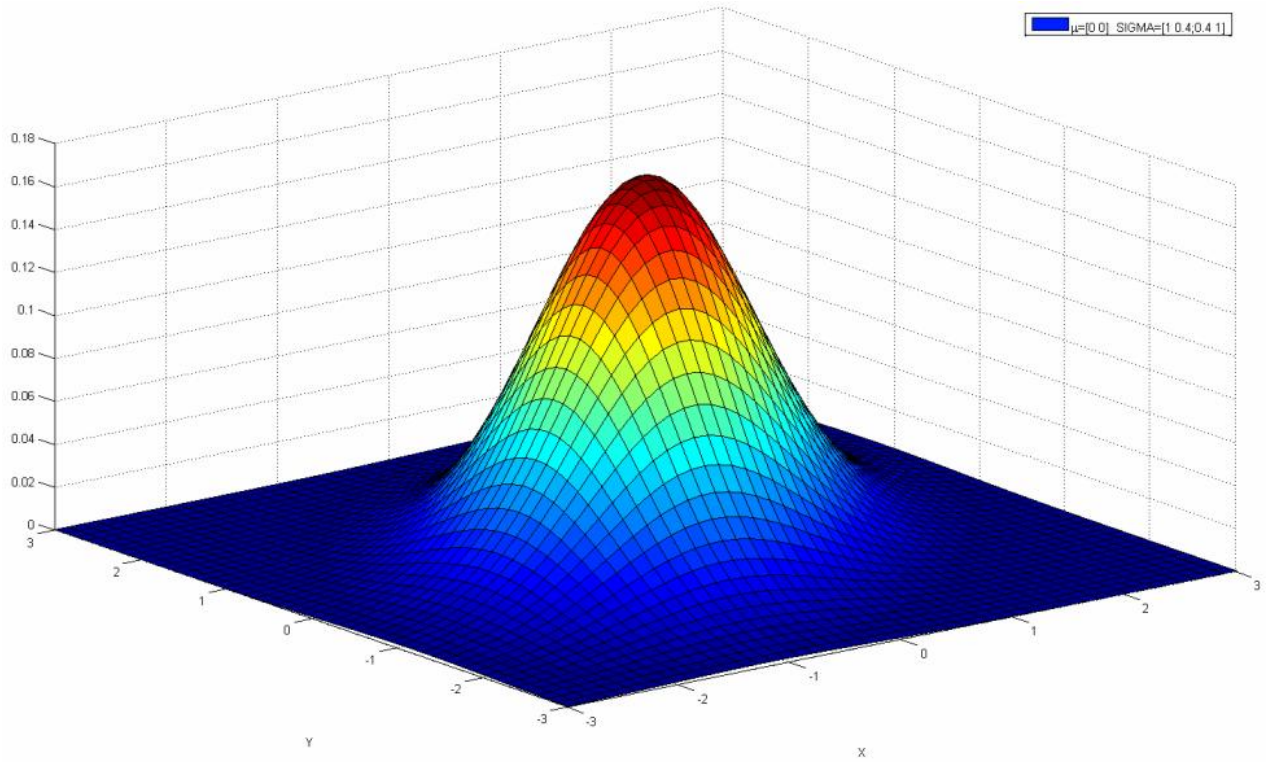


Слика 2.3.1: Примери нормалних распореда у зависности од скаларних параметара

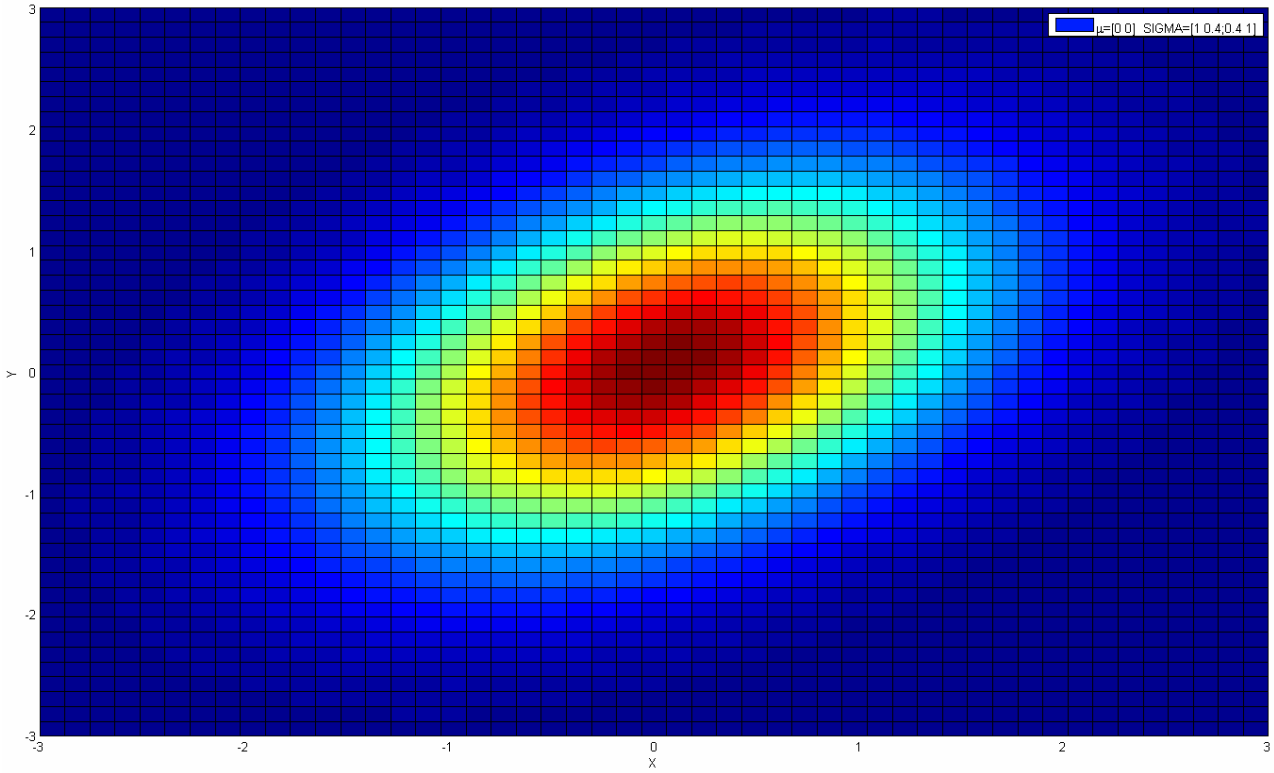
У случају да је променљива x вектор, тада расподела поприма просторни облик, а према одговарајућим параметрима расподеле:

$$p(x) \sim N(\mu, \Sigma) \tag{2.3.2}$$

То је приказано на слици 2.3.2 за вектор x=[X,Y], а на слици 2.3.3 приказана је расподела у X,Y са закривљеношћу због вредности стандардне девијације SIGMA= Σ.



Слика 2.3.2: Пример нормалне расподеле за вектор x=[X,Y] и параметре μ=[0 0], Σ=[1 0.4; 0.4 1]



Слика 2.3.3: Нормална расподела за вектор x=[X,Y] у равни X,Y

Променљиве неког система или процеса могу се поделити на две врсте и то дискретне и непрекидне променљиве. Оне могу бити бити међусобно зависне или међусобно независне. Када постоје две променљиве x и y које описују неки систем, и када су оне међусобно зависне, онда се расподела променљиве x за познато y представља као:

$$p(x \mid y). \qquad (2.3.3)$$

За случај неког система којим се управља могу се написати једначини стања система (2.3.4) и једначини мерења – опсервациони модел (2.3.5) на следећи начин:

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t \quad (2.3.4)$$

$$z_t = C_t x_t + \delta_t \quad (2.3.5)$$

где је:

x_t - вектор стања система у тренутку t

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

A_t - матрица nxn која одређује како систем из стања у тренутку $t-1$ прелази у стање у тренутку t без утицаја управљања или шумова

u_t - вектор управљање који дефинише прелазак стања система из $t-1$ у t

B_t - матрица nxl која одређује како управљање u_t мења стање система из стања у тренутку $t-1$ у стање у тренутку t

ε_t - вектор случајних променљивих које моделирају шум и остале поремећаје у систему

z_t - вектор мерења у тренутку t

C_t - матрица kxn која успоставља везу између стања у тренутку t и мерења у тренутку t

δ_t - вектор случајних променљивих које моделирају шум и остале поремећаје

Узима се у обзир да вектор стања система и вектор мерења подлежу нормалној расподели, што се може формулисати на следећи начин:

$$p(x_t \mid u_t, x_{t-1}) = N(x_t; A_t x_{t-1} + B_t u_t, R_t) \quad (2.3.6)$$

$$p(z_t \mid x_t) = N(z_t; C_t x_t, Q_t) \quad (2.3.7)$$

где су R_t и Q_t познате матрице коваријанси које припадају случајним променљивима ε_t и δ_t респективно.

Према свему предходно реченом, Rudolf E. Kalman је формулисао филтер према следећем алгоритму:

ALGORITAM kalmanov_filter$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$

1. $\overline{\mu}_t = A_t \mu_{t-1} + B_t u_t$
2. $\overline{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$

Корак предикције

3. $K_t = \overline{\Sigma}_t C_t^T \left(C_t \overline{\Sigma}_t C_t^T + Q_t \right)^{-1}$
4. $\mu_t = \overline{\mu}_t + K_t \left(z_t - C_t \overline{\mu}_t \right)$
5. $\Sigma_t = \left(I - K_t C_t \right) \overline{\Sigma}_t$

Корак корекције

return μ_t, Σ_t

где је K_t калманово појачање, μ_t очекивана вредност расподеле и Σ_t матрица коваријанси.

Алгоритам се лако може применити у случају линеарне зависности функције x_t од x_{t-1} и u_t, тј. ако је једначина стања линеарна. У случају да је једначина стања нелинеарна функција предходног стања x_{t-1} и управљања u_t (једначина 2.3.8) и опсервациони модел нелинеарна функција стања система (једначина 2.3.9), тј:

$$x_t = g(u_t, x_{t-1}) \tag{2.3.8}$$
$$z_t = h(x_t), \tag{2.3.9}$$

Онда Калманов филтер не даје одговарајуће резултате према реалном стању. Тада се примењује тзв. Линеаризовани Калманов филтер где се модел система и модел перцепције развијају у Тејлоров ред на следећи начина:

$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + \frac{\partial g(u_t, \mu_{t-1})}{\partial x_{t-1}}(x_{t-1} - \mu_{t-1})$$
$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + G_t(x_{t-1} - \mu_{t-1}) \tag{2.3.10}$$

$$h(x_t) \approx h(\overline{\mu}_t) + \frac{\partial h(\overline{\mu}_t)}{\partial x_t}(x_t - \overline{\mu}_t)$$
$$h(x_t) \approx h(\overline{\mu}_t) + H_t(x_t - \overline{\mu}_t). \tag{2.3.11}$$

Према томе линеаризовани Калманов филтер има следећи алгоритам:

ALGORITAM linearizovan_kalmanov_filter$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$

1. $\overline{\mu}_t = g(u_t, \mu_{t-1})$
2. $\overline{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$
3. $K_t = \overline{\Sigma}_t H_t^T \left(H_t \overline{\Sigma}_t H_t^T + Q_t\right)^{-1}$
4. $\mu_t = \overline{\mu}_t + K_t(z_t - h(\overline{\mu}_t))$
5. $\Sigma_t = (I - K_t H_t)\overline{\Sigma}_t$

return μ_t, Σ_t

Корак предикције

Корак корекције

2.3.1. Примена Калмановог филтера

У пројектном решењу мобилног робота примењује се Калманов филтер за локализацију. Локализација подразумева процес одређивања позиције и оријентације мобилног робота у односу на мапу радног окружења које може бити статичко или динамичко, а у пројекту ће бити коришћено статичко окружење. Према [5] основна подела локализације је:

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

У пројекту ће бити коришћена локална локализација. Такође према [5], посматрано са становишта управљања мобилним роботом, локализација може бити:

- *Ативна локализација* – алгоритми локализације имају потпуну контролу над основним алгоритмом управљања мобилним роботом, и
- *Пасивна локализација* – подсистем за локализацију мобилног робота представља само један подсистем система управљања. У том случају локализација има само једну функцију и то одређивање позиције и оријентације мобилног робота, док главни систем надгледа функционисање и извршавање постављеног задатка.

У пројекту ће бити коришћена пасивна локализација у систему управљања мобилним роботом.

Према томе, алгоритми предикције и корекције за случај мобилног робота са вектором стања са две координате за позицију и једну координату за оријентацију, дати су у табели 2.3.1.1 према [5] и према њима се воде одговарајући програми.

***Табели 2.3.1.1**: Алгоритам линеаризованог Калмановог филтера локализације*

Linearizovan_kalmanov_filter_lokalizacija $(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t, m)$

КОРАК ПРЕДИКЦИЈЕ

1. $G_t = \dfrac{\partial g(u_t, \mu_{t-1})}{\partial x_{t-1}} = \begin{pmatrix} \dfrac{\partial x}{\partial \mu_{t-1,x}} & \dfrac{\partial x}{\partial \mu_{t-1,y}} & \dfrac{\partial x}{\partial \mu_{t-1,\theta}} \\ \dfrac{\partial y}{\partial \mu_{t-1,x}} & \dfrac{\partial y}{\partial \mu_{t-1,y}} & \dfrac{\partial y}{\partial \mu_{t-1,\theta}} \\ \dfrac{\partial \theta}{\partial \mu_{t-1,x}} & \dfrac{\partial \theta}{\partial \mu_{t-1,y}} & \dfrac{\partial \theta}{\partial \mu_{t-1,\theta}} \end{pmatrix}$ - јакобијан функције g у односу на положај

2. $V_t = \dfrac{\partial g(u_t, \mu_{t-1})}{\partial u_t} = \begin{pmatrix} \dfrac{\partial x}{\partial v_t} & \dfrac{\partial x}{\partial \omega_t} \\ \dfrac{\partial y}{\partial v_t} & \dfrac{\partial y}{\partial \omega_t} \\ \dfrac{\partial \theta}{\partial v_t} & \dfrac{\partial \theta}{\partial \omega_t} \end{pmatrix}$ - јакобијан функције g у односу на управљање

3. $M_t = \begin{pmatrix} (\alpha_1 |v_t| + \alpha_2 |\omega_t|) & 0 \\ 0 & (\alpha_3 |v_t| + \alpha_4 |\omega_t|) \end{pmatrix}$ - шум (грешка) управљања

4. $\overline{\mu}_t = g(u_t, \mu_{t-1})$ - **предвиђена очекивана вредност**

5. $\overline{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + V_t M_t V_t^T$ - **предвиђена матрица коваријанси**

КОРАК КОРЕКЦИЈЕ

6. $\hat{z}_t = \begin{pmatrix} \sqrt{(m_x - \overline{\mu}_{t,x})^2 + (m_y - \overline{\mu}_{t,y})^2} \\ \operatorname{atan2}(m_y - \overline{\mu}_{t,y}, m_x - \overline{\mu}_{t,x}) - \overline{\mu}_{t,\theta} \end{pmatrix}$ - предвиђена очекивана вредност мерења

7. $H_t = \dfrac{\partial h(\mu_t, m)}{\partial x_t} = \begin{pmatrix} \dfrac{\partial r_t}{\partial \mu_{t,x}} & \dfrac{\partial r_t}{\partial \mu_{t,y}} & \dfrac{\partial r_t}{\partial \mu_{t,\theta}} \\ \dfrac{\partial \varphi_t}{\partial \mu_{t,x}} & \dfrac{\partial \varphi_t}{\partial \mu_{t,y}} & \dfrac{\partial \varphi_t}{\partial \mu_{t,\theta}} \end{pmatrix}$ - јакобијан функције h у односу на положај

8. $Q_t = \begin{pmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_r^2 \end{pmatrix}$ - шум мерења

9. $S_t = H_t \overline{\Sigma}_t H_t^T + Q_t$ - предвиђена матрица коваријанси мерења

10. $K_t = \overline{\Sigma}_t H_t^T S_t^{-1}$ - Калманово појачање

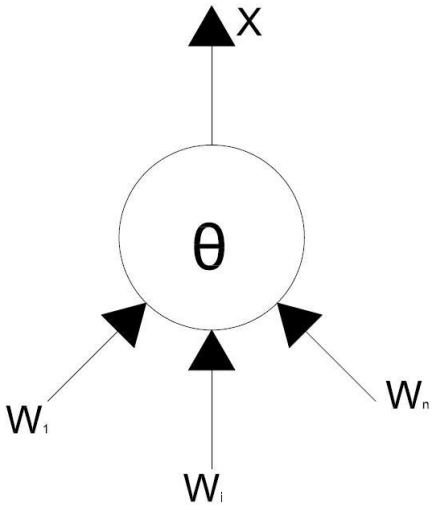
11. $\mu_t = \overline{\mu}_t + K_t (z_t - \hat{z}_t)$ - **предвиђена очекивана вредност**

12. $\Sigma_t = (I - K_t H_t) \overline{\Sigma}_t$ - **нова матрица коваријанси**

2.4. Вештачке неуронске мреже

Вештачке неуронске мреже омогућавају значајне предности при решавању проблема процесирања који захтевају рад у реалном времену и интерпретацију међусобних односа између променљивих у вишедимензионалним просторима [3]. Неуронске мреже се користе као универзални апроксиматори и "софистицирани сензори". Основне категорије употребе вештачких неуронских мрежа су *предикција, класификација* и *функционална апроксимација* [1]. Вештачке неуронске мреже су направљење аналогно мозгу човека. Принцип функционисања је да се на основу основних процесирајућих елемената - неурона извршавају функције процесирања, учења и самоорганизовања. Неуронске мреже представљају скуп неурона међусобно повезаних везама које носе одређене тежинске односе [3]. Имају могућност адаптивног понашања, и то кроз учење на одређеном узорку. Њихова основна предност је што оне на основу тог узорка могу да, посредством пресликавања улазног вектора у излазни, прошире ту адаптивност на делове који се не налазе у улазном (обучавајућем) скупу, односно у том узорку.

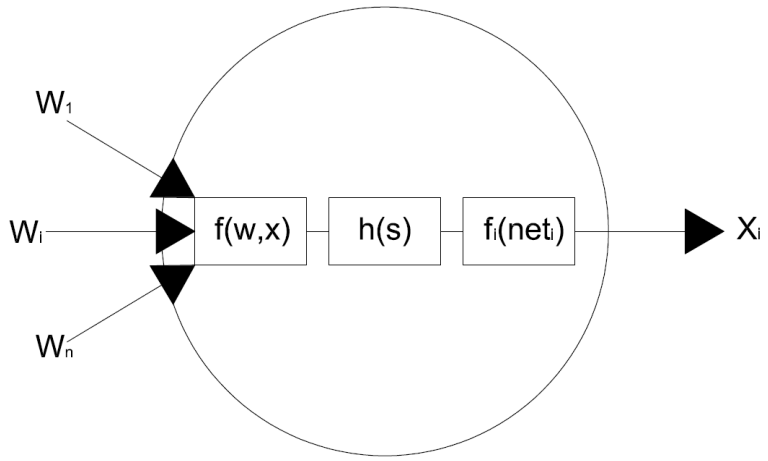
Неурони су основни процесирајући елементи вештачких неуронских мрежа. Они примају улазне сигнале и информације од окружења или других неурона. Ови сигнали се приказују преко улазног вектора, а међусобни односи између неурона представљају њихове тежинске односе. Овај праг је дефинисан тежинским односима [1]. На слици 2.4.1 је приказан неурон са окружењем. Тежински односи (W1,....,Wn) су утицаји других неурона. Сваки неурон поседује потенцијал мировања ( ), који када се прекорачи, побуђује неурон и он постаје акиван за интеракцију са осталим неуронима. Другим речима, неурон ће генеристи излаз X, ако и само ако збир тежиснких односа других неурона прекорачи његов потенцијал мировања.



Слика 2.4.1: Неурон са својим окружењем

Основи елементи неурона су [1] (слика2.4.2):

- **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 активацону функцију.
- **Активациона функција** $f_i=(net_i)$ обрађује излаз функције преноса и обезбеђује излазну вредност неурона



w_{i1}, w_{11}, w_{in} – тежински односи других неурона

$h(s)$ – функција преноса

$f_i=(net_i)$ – активациона функција

x_i – излаз из неурона

Слика 2.4.2: Основни елементи неурона

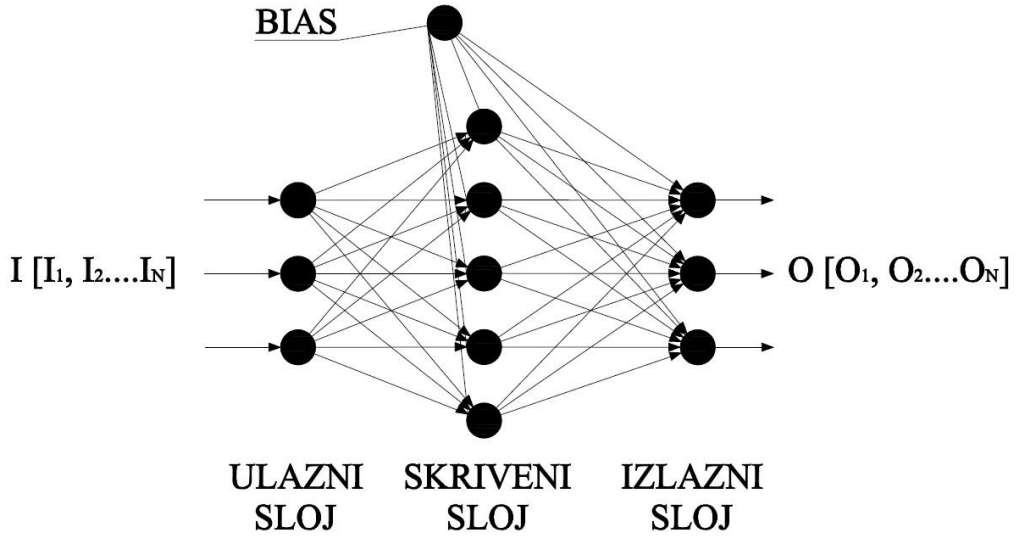
Активационе функције су најважније за правилно функционисање неурона. За различите моделе вештачких неуроских мрежа се користе и различите активационе функције. Неке од активационих функција се могу сврстати у неколико група [3]:

- Линеарне
- Бинарне
- Сигмоидне
- Компетитивне
- Гаусове.

Модели вештачких неуронских мрежа не представљају ништа друго до скупове неурона, поређаних у више слојева. Број слојева варира од случаја до случаја, али сваки модел мора да садржи бар два слоја (улазни и излазни). Међутим, често између ова два слоја постоји један или више скривених слојева [3]. Број неурона у улазном и излазном слоју је дефинисам бројем улаза или излаза у мрежу, док број скривених слојева, као и број неурона у њима варира од случаја до случаја. Варирањем ових вредности се долази до оптималних вредности за конкретан случај. Управо је одређивање броја скривених слојева и броја неурона у њима, тако да се нађе оптимално решење, главни задатак пројектанта.

Најраспрострањенији модели вештачких неуронских мрежа су:

- **Перцептон** је најранији модел вештачке неуронске мреже. Он користи простирање сигнала у једном правцу, има један или више слојева (најчешће два), и функционише на бази супервизорског учења [1]. Користи линеарни тип неурона
- **Backpropagation - BP'** неуронске мреже решавају проблеме нелинеарног пресликавања. Њена главна карактеристика је постојање скривених слојева. Користе градијентни поступак при обучавању (минимизација грешке). Када добије излаз, користи грешку за модификовање тежиских односа што омогућује настављање процеса обучавања до задате грешке. Једна од карактеристика ових мрежа је постојање такозваног "*bias"* неурона, који представља константан члан у тежинским сумама неурона наредног слоја (слика 2.4.3)



Слика 2.4.3: Архитектура BP вештачке неуронске мреже

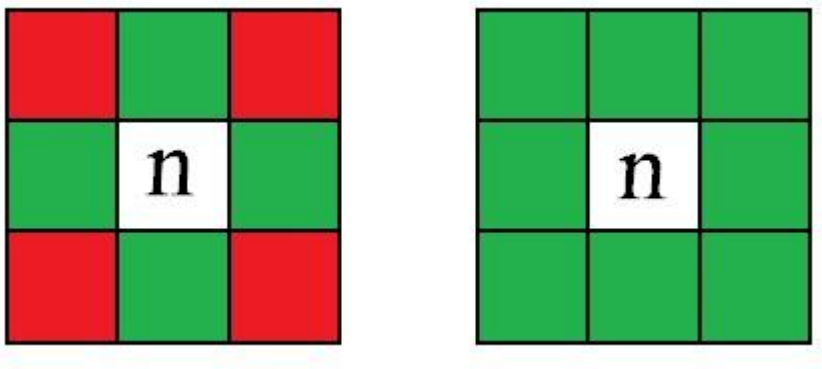
- Асоцијативне неуронске мреже
- Хопфилдове ове неуронске мреже
- **APT (*"Adaptive Responce Theory"*)** неуроске мреже имају способност чувања старих улаза приликом учења нових облика. Највећу примену налази у проблемима класификације. У зависности од врсте користи бинарне, или аналогне улазне облике. Користи компетитивно учење са повратним спрегама [3].
- "Fuzzy logic" неуронске мреже
- Самоорганизујуће неуронске мреже

2.5. Пројектовани A алгоритам претраге*

Алгоритам А звезда је алгоритам претраживања који проналази најкраћу путању између почетне и циљне тачке у одређеном окружењу. Користећи хеуристички прихватљиве процене, алгоритам поступно „пролази кроз задато технолошко окружење" бирајући оптималну путању, у смислу најкраћег пређеног пута. Окружење је неопходно дискретизовати и дефинисати на начин који ће омогућити да се у обзир узму ограничења унутар задатог окружења.

Одабир оптималне путање врши се помоћу основних параметара, према изразу f(n)=h(n)+g(n), где су: n број чворова, h процена најкраћег пута дефинисаног Еуклидском нормом од задатог циљног до посматраног чвора дискретизованог окружења и g цена помераја, која практично представља пређени пут између два суседна чвора. Поред основних параметара, дефинишу се и препреке унутар технолошког окружења и укупан пређени пут који представља збир свих парцијалних цена помераја g у току одабира оптималне путање. На основу наведених параметара, израчунава се оцена најкраћег пута, од стартног до циљног чвора. Израчунавањем и избором минималне вредности параметара f за све суседне чворове, у односуна чвор у коме се тренутно налази, А звезда алгоритам бира следећи чвор. Претходна процедура се понавља све док се не достигне циљни чвор, тако да се на тај начин генерише оптимална путања која представља низ одабраних тачака. У конкретном случају овај алгоритам је коришћен за генерисање оптималне путање кретања мобилног робота унутар лабораторијског модела технолошког окружења, узимајући при томе у обзир пројектовани диспозициони план распореда машина алатки, монтажних станица и складишта [1].

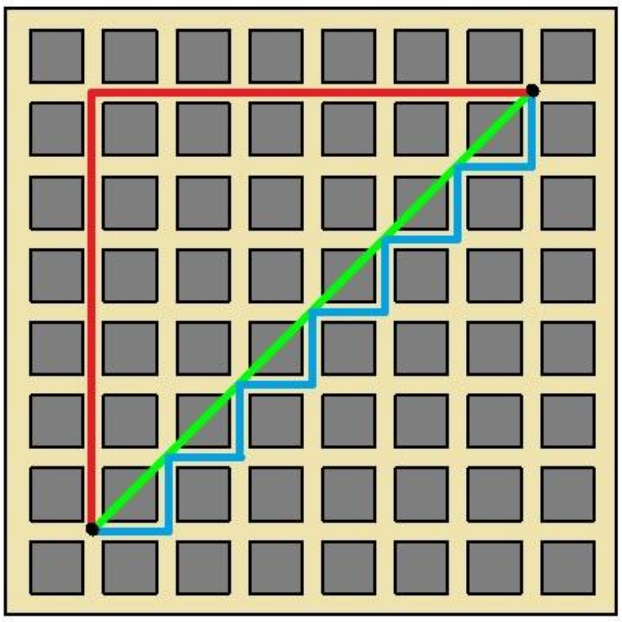
Прелаз са једног на други пиксел може бити остварен као четвороспојиви и осмоспојиви као што је приказано на слици 2.5.1. На слици су зеленом бојом призана поља у која се може прећи, а црвеном поља у која се не може прећи из поља n. Коришћен је принцип осмоспојивих пиксела где је дискретизација могућа са знатно већом прецизношћу. У другом случају да је коришћен четвороспојиви постојале би знатне потешкоће око тражења најкраће путање јер је коришћен пиксел од 10cm, а окружење само је 100cm x 150cm или 10 x 15 пиксела укључујући у тај опсег и целокупе препреке које робот треба да обиђе.



Слика 2.5.1: Четвороспојиви и осмоспојиви пиксели

Када је завршен одабир прелаза са пиксела на пиксел може се прећи на одабир принципа одређивања најкраћег пута. Принципи одређивања најкраћег пута могу бити: Хеуристички и Еуклидски. На слици 2.5.2. су приказане путање које се добијају овим поступцима. Са црвеном бојом је приказана путања добијена еуклидском нормом, док је са плавом означена путања добијена Хеуристичким приступом. Најкраћа путања по принципу ваздушне линије уз помоћ које се рачуна Хеуристичка путања је означена са зеленом бојом.

Фаворит је у сваком случају Хеуристички приступ преко којег је могуће извршити бољу дискретизацију од почетне до циљне тачке. У комбинацији са осмоспојивим пикселима могуће је добити приближно најлогичнију и најкраћу путању којом би се и човек кретао кроз задато окружење. Треба поменути и да ова путања умногоме зависи од величине пиксела. Што је пиксел мањи то ће путања бити боље дискретизована.



Слика 2.5.2: Еуклидска и Хеуристичка норма

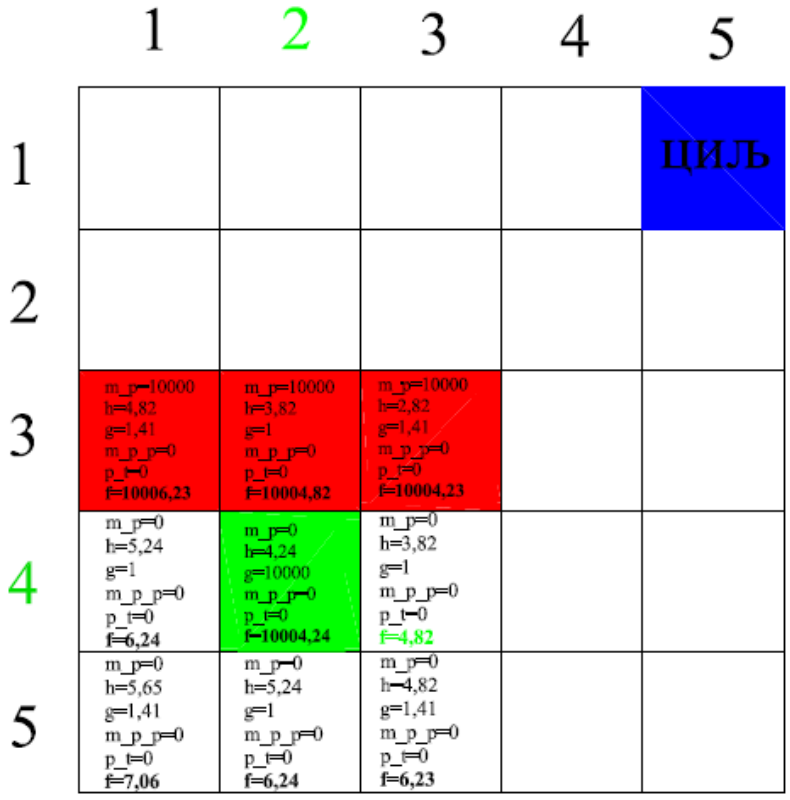
2.5.1. Пројектовање алгоритма у МАТЛАБ окружењу

Сада када је одабран приступ одређивања најкраће путање и да су пиксели осмоспојиви може се прећи на програмирање А звезде. Рачунање најкраћег пута ради се постепено корак по корак што подразумева рачунање најмање вредности f за сваки следећи корак (слике 2.5.1.2 до 2.5.1.5). У рачуну матрице f, која је величине 3x3, фигурише матрица препрека која говори где су поља која робот не сме да посети. Матрица која највише утиче на одабир најмање вредностии и тачке у коју ће робот кренути је матрица h која представља мапу удаљености тренутне од циљне тачке. Матрица g узима вредности 1,41 за прелазак у наредно поље по дијагонали и вредност 1 за кретање по вертикали и хоризонтали у било ком правцу. Поред матрице g јавља се и матрица пређеног пута која је исте величине као као матрица f како би биле компатибилне при рачуну. Разлика између матрице пређеног пута и матице g је у томе што матрица g није променљива док се за матрицу пређеног пута вредност повећава при сваком кораку за инкремент. Ограничавањем ових матрица на 3x3 које одговарају околини тренутне позиције и простим сабирањем ових вредности добија се матрица f. Одређивањем најмање вредности из ове матрице дефинисали смо тачку у коју ће робот да крене.

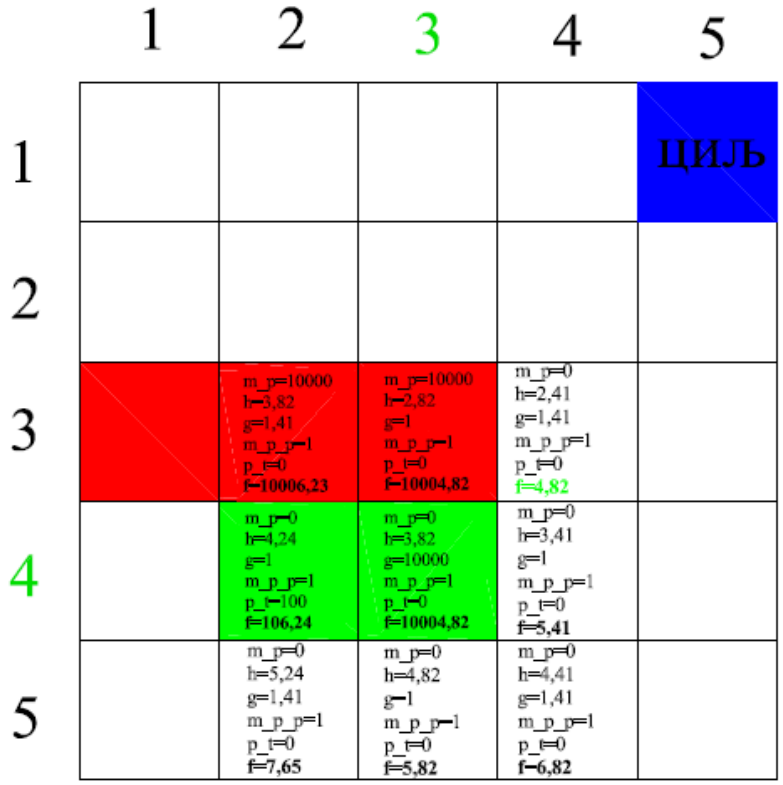
f=matrica _prepreka_3x3+ h_matrica_3x3+g+matrica_predjenog_puta_3x3+posecene_tacke_3x3 (2.5.1.1)

10000	10000	0	2	1	0	1,41	1	1,41	2,41	2,41	2,41	0	0	0
0	0	0	2,41	1,41	1	1	10000	1	2,41	2,41	2,41	100	100	0
0	0	0	2,82	2,41	2	1,41	1	1,41	2,41	2,41	2,41	0	0	0

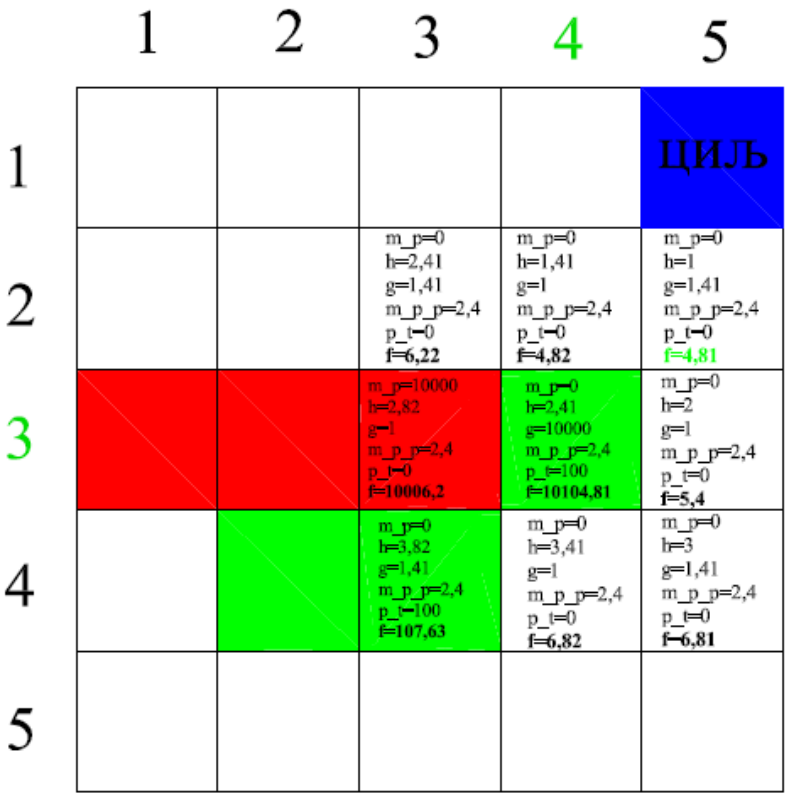
Слика 2.5.1.1: Примери матрица 3x3: а) препрека, б) h, в) g, г) пређеног пута и д) посећених тачака



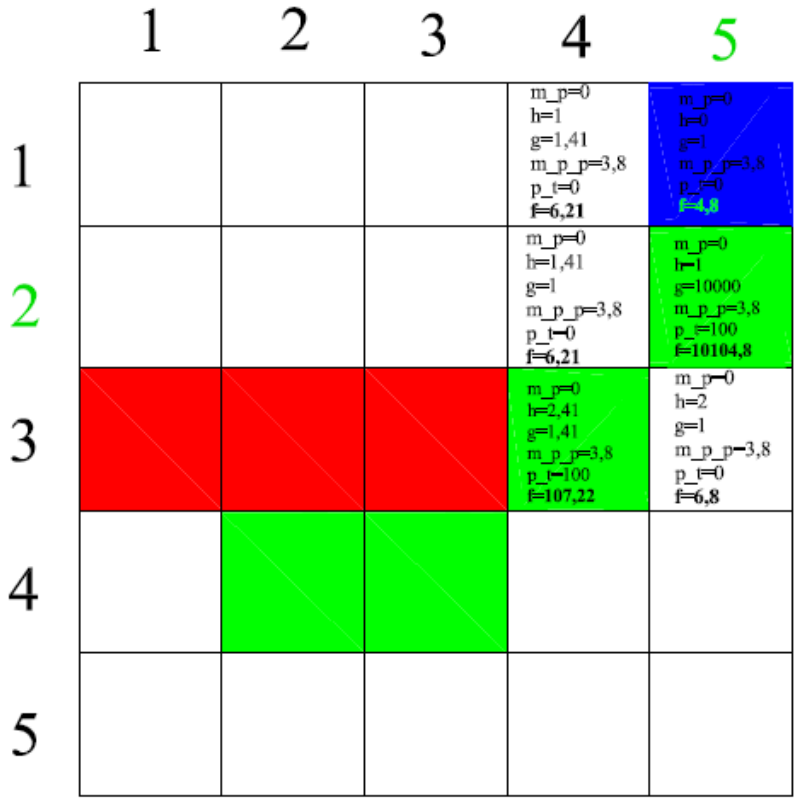
Слика 2.5.1.2: Први корак при налажењу најкраће путање



Слика 2.5.1.3: Други корак при налажењу најкраће путање



Слика 2.5.1.4: Трећи корак при налажењу најкраће путање



Слика 2.5.1.5: Последњи корак при налажењу најкраће путање

Програм који је генерисан у Matlab окружењу заснован је на претходно објашњеном принципу. Програм алгоритма А звезда је подпрограм који се убацује у главни програм. Дефинисање А звезде као подпрограма остварује се у првом реду, а на тај начин се и позива у главни програм. Затим је потребно унети почетну и циљну тачку са тастатуре уз услов је да се тачке налазе у задатом окружењу и да не прелазе вредности 30 за i координату и 20 за j координату. Уколико овај услов није испуњен на монитору ће се исписати „вредности за i и j морају бити у прописаним границама".

Главни програм алгоритма А звезда ради уз помоћ три подпрограма: h_generator_posle_miroslava, pređeni_put_i_nova_pozicija и crtez. Конкретан опис ових програма биће дат у наставку, а за опис главног програма је битно поменути које задатке имају ови подпрограми. H_generator_posle_miroslava има задатак да направи h матрицу која представља удаљености тренутних поља од циља. Pređeni_put_i_nova_pozicija одређује инкремент пређеног пута у зависности од предходне тачке и проглашава нове вредности i и j у које је робот дошао. Подпрограм crtez даје графички приказ окружења и уцртава путању којом ће се робот кретати.

Одређивање матрице препрека је рађено за конкретан пример технолошког окружењау којем ће се робот кретати. Ова матрица је дефинисана са нулама где нема препреке и са вредностима 10000 где постоје препреке. Важно је поменути да се под препрекама сматрају и пиксели око самих препрека јер уколико би робот дошао до њих он би ударио једним својим делом у препреку. То се дешава из разлога што је центар робота на средини осе главног вратила па се размак између точкова, као сами точкови мора узети у обзир. Затим је дефинисана и матрица g на начин како је то објашњено предходно. Pot и krt представљају координате почетне и крајње тачке.

Део програма који је одвојен цртицама односи се на дефинисање почетних вредности пре него што ће отпочети while петља. У while петљи се поступак рачунања најоптималнијег пиксела за прелазак понавља док се не стигне до циљне тачке. Матрица пређеног пута у почетном тренутку мора имати вредности 0 на свакој позицији како би била веродостојна за

први циклус одређивања коначне матрице f. Rezultat_i и rezultat_j представљају коначне матрице излаза из алгоритма а звезда и у њих се пакују тачке у које робот треба да иде. На почетку се дефинишу као празне матрице. Неопходан део за одвијање while петље је дефинисање почетне вредности за i јер while петља повећава i за један све док се не испуни услов да је i=20. Пређени пут у почетном тренутку је нула. Вредности e и d се појављују само због цртежа као почетне тачке за цртање путање.

Са почетком while петље потребно је сваки пут у зависности од тренутне позиције да се исеку матрица препрека и матрица h на 3x3 како би све матрице за рачун биле истог формата и спремне за рачун. Формула 1 сабира све матрице и у следећем кораку се тражи минимална вредност и њена позиција (врста и колона). Затим се позива функција која одређује инкремент пређеног пута и нову позивцију. Укупан пређени пут се одређује тако што се кроз сваки корак сабира са претходним пређеним путем. Потребно је ту вредност сврстати опет у матрицу 3x3 како би била спремна за следећи рачун функције f. Нове вредности за i и j се проглашавају за почетно i и j па се у следећем кораку while петље рачунање врши за нове вредности. Ове вредности се пишу у матрицама rezultat_i и rezultat_j. Потребно је и повећати вредност f у тачкама у којима је робот већ био. Ово је значајно због проблема који се јављају при обиласку дугачке препреке и обилажења једног пиксела који се нашао тачно по дијагонали између тренутне и почетне тачке, а при томе и од тог пиксела по другој дијагонали постоје још по један пиксел препреке у оба правца.

Уколико је испуњен услов да тренутна тачка одговара циљној тачки вредност параметра i се повећава тако да је услов while петље испуњен и да се она прекида. У супротном вредност параметра i се повећава за 1.

По завршетку while петље добили смо тачке у које желимо да робот оде, али је због графичког приказа неопходно проширити матрице за почетну тачку како би путања била представљена од почетне до крајње тачке. Графички приказ се врши позивањем подпрограма crtez.

Како би резултати алгоритма А звезда били прилагођени главном програму потребно је пикселе претворити у центиметре и ради лакшег коришћења спаковати у једну матрицу.

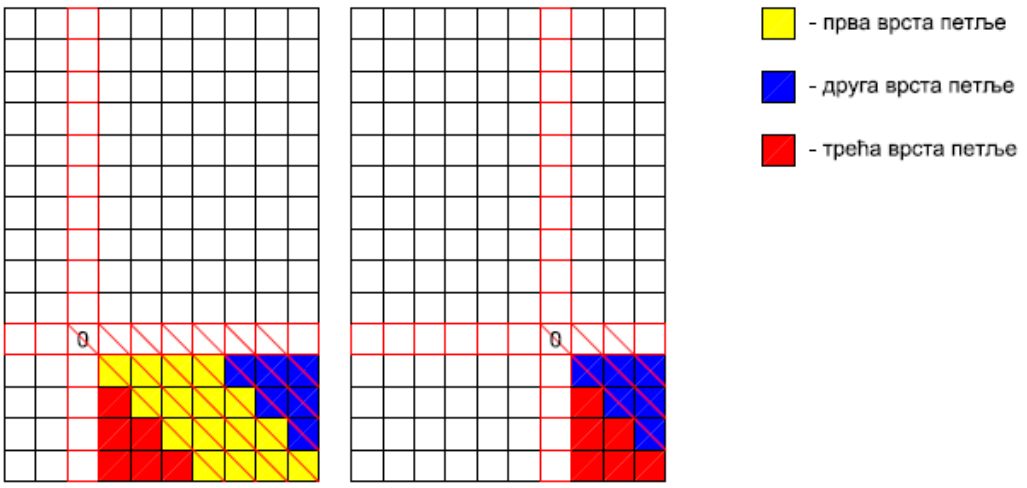
```
function [path] = a_zvezda
clc; clear all; close all

i_pocetno=input('i_pocetno = ');
j_pocetno=input('j_pocetno = ');
i_ciljno=input('i_ciljno = ');
j_ciljno=input('j_ciljno = ');
if i_pocetno<1 || i_pocetno>30 || j_pocetno<1 || j_pocetno>20 || i_ciljno<1 || i_ciljno>30 ||
j_ciljno<1 || j_ciljno>20

disp('vrednosti za i i j moraju biti u propisanim granicama')
else
    format('longG');

    [h] = h_generator_posle_miroslava(i_ciljno,j_ciljno);      load matrica_prepreka
    g=[1.41 1 1.41; 1 1000000 1; 1.41 1 1.41]

    pot=[i_pocetno j_pocetno]
    krt=[i_ciljno j_ciljno]
    %--------------------------------------------------------------------
    matrica_predjenog_puta_3x3=zeros(3,3);
    rezultat_i=[];
    rezultat_j=[];
    i=1;
    predjeni_put=0;
    e=i_pocetno;
    d=j_pocetno;
    posecene_tacke=zeros(30,20);
    %--------------------------------------------------------------------

    while i~=40
        h_matrica_3x3=(h(i_pocetno-1:i_pocetno+1 , j_pocetno-1:j_pocetno+1));
        matrica_prepreka_3x3=(matrica_prepreka(i_pocetno-1:i_pocetno+1 , j_pocetno-1:j_pocetno+1));
        posecene_tacke_3x3=(posecene_tacke(i_pocetno-1:i_pocetno+1 , j_pocetno-1:j_pocetno+1));
```

```
        f= g + matrica_prepreka_3x3 + h_matrica_3x3 + matrica_predjenog_puta_3x3  +
posecene_tacke_3x3;

        [minimum, kolona]= min(min(f));
        k=f(:,kolona)';
        [minimum, vrsta]= min(k);

        [predjeni_put_inkrement,i_novo,j_novo]=
predjeni_put_i_nova_pozicija(vrsta,kolona,i_pocetno,j_pocetno);
        predjeni_put=predjeni_put+predjeni_put_inkrement;
        matrica_predjenog_puta_3x3=ones(3,3)*predjeni_put;

        i_pocetno=i_novo;
        j_pocetno=j_novo;

        rezultat_i=[rezultat_i,i_pocetno];
        rezultat_j=[rezultat_j,j_pocetno];

        posecene_tacke(i_pocetno,j_pocetno)=100;

        if i_pocetno==krt(1) && j_pocetno==krt(2);
            i=39;
        end
        i=i+1;
    end
end

x = [e, rezultat_i];
y = [d, rezultat_j];
[s] = crtez(x,y, pot,krt);

itrans = [rezultat_i.*5-2.5]';
jtrans = [rezultat_j.*5-2.5]';
path = [itrans, jtrans];
```

2.5.1.1. Подпрограм h матрица

H матрица као што је речено у подглављу 2.5.1. користи се за одређивање дискретизованог растојања између почетне и циљне тачке. Растојање се мери од циљне тачке. Пиксели по хоризонтали у сва четири правца од циљне тачке редом се повећавају за вредност 1, а по дијагоналама по хеуристичком приступу добијају вредност 1,4142135623731. Заокруживање на више децимала је врло битно при рачуну матрице f односно при одређивању минималне вредности.

Почетни параметар програмирања је циљна тачка. Потребно је генерисати мапу на број пиксела у оба правца и одредити колико има пиксла од циљне тачке до краја мапе у сва четири правца. Потом се додељују вредности за хоризонталан и вертикалан правац у односу на предходно одређен број пиксела до краја мапе.



Слика 2.5.1.1.1: Пинцип програмирања h матрице

Други део програма дели се на четири дела где се додељују дијагоналне вредности од циљне тачке: горе десно, горе лево, доле десно и доле десно. У оквиру сваке од ових петљи постоје три петље са слике 2.5.1.1.1. Прва врста петље су петље где постоји коначан број пиксела по дијагонали, а у зависности од позивиције циљне тачке могуће је да ова петља није потребна као што се види на слици под б. Другој и трећој врсти петље се број пиксела за доделу смањује са удаљавањем од циљне тачке.

```
functиon [h] = h_generator_posle_миroslava(и_силjno, j_силjno)


for i = 1:30
for j = 1:20
kt = [j_ciljno i_ciljno];

inkh = 1;
inkv = 1;
inkd = sqrt((inkh^2)+(inkv^2));

h = ones(30,20)*0;

x = kt(1,1);
y = kt(1,2);

h(y,x) = 0;

ndes = size(h,2)-kt(1,1);
nlev = size(h,2)-ndes-1;
ndol = size(h,1)-kt(1,2);
ngor = size(h,1)-ndol-1;

%=============== DODELA VREDNOSTI HORIZONTALNO I VERTIKALNO ===============
for k = 0 : ndes
    h(y,x+k) = k*inkh;
end
for k = 0 : nlev
    h(y,x-k) = k*inkh;
end
for k = 0 : ndol
    h(y+k,x) = k*inkv;
end
for k = 0 : ngor
    h(y-k,x) = k*inkv;
end

%================== DODELA VREDNOSTI OSTATKU GORE DESNO ==================
if ndes <= ngor
    for k = 0 : ndes-1
        for e = 1 : ndes-k
            h(y-e,x+(e+k)) = e*inkd+h(kt(1,2),kt(1,1)+k);
        end
    end

else
    for k = 0 : ndes-ngor
        for e = 1 : ngor
            h(kt(1,2)-e,kt(1,1)+(e+k)) = e*inkd+h(kt(1,2),kt(1,1)+k);
        end
    end
    for k = size(h,2)-ngor-nlev : ndes-1
        for e = 1 : ndes-k
            h(kt(1,2)-e,kt(1,1)+(e+k)) = e*inkd+h(kt(1,2),kt(1,1)+k);
        end
    end
end

for k = 1 : ndes
    for e = k+1 : ngor
        h(y-e,x+k) = h(y-k,x+k)+e-k;
end

%================== DODELA VREDNOSTI OSTATKU GORE LEVO ==================
if nlev <= ngor
    for k = 0 : nlev-1
        for e = 1 : nlev-k
            h(y-e,x-(e+k)) = e*inkd+h(kt(1,2),kt(1,1)-k);
        end
    end
```

```
else
    for k = 0 : nlev-ngor
        for e = 1 : ngor
            h(kt(1,2)-e,kt(1,1)-(e+k)) = e*inkd+h(kt(1,2),kt(1,1)-k);
        end
    end


    for k = size(h,2)-ngor-ndes : nlev-1
        for e = 1 : nlev-k
            h(kt(1,2)-e,kt(1,1)-(e+k)) = e*inkd+h(kt(1,2),kt(1,1)-k);
        end
    end
end

for k = 1 : nlev
    for e = k+1 : ngor
        h(y-e,x-k) = h(y-k,x-k)+e-k;
    end
end

%================== DODELA VREDNOSTI OSTATKU DOLE DESNO ====================
if ndes <= ndol
    for k = 0 : ndes-1
        for e = 1 : ndes-k
            h(y+e,x+(e+k)) = e*inkd+h(kt(1,2),kt(1,1)+k);
        end
    end


else
    for k = 0 : ndes-ndol
        for e = 1 : ndol
            h(kt(1,2)+e,kt(1,1)+(e+k)) = e*inkd+h(kt(1,2),kt(1,1)+k);
        end
    end

    for k = size(h,2)-ndol-nlev : ndes-1
        for e = 1 : ndes-k
            h(kt(1,2)+e,kt(1,1)+(e+k)) = e*inkd+h(kt(1,2),kt(1,1)+k);
        end
    end
end

for k = 1 : ndes
    for e = k+1 : ndol
        h(y+e,x+k) = h(y+k,x+k)+e-k;
    end
end

%================== DODELA VREDNOSTI OSTATKU DOLE LEVO ====================
if nlev <= ndol
    for k = 0 : nlev-1
        for e = 1 : nlev-k
            h(y+e,x-(e+k)) = e*inkd+h(kt(1,2),kt(1,1)-k);
        end
    end
else
    for k = 0 : nlev-ndol
        for e = 1 : ndol
            h(kt(1,2)+e,kt(1,1)-(e+k)) = e*inkd+h(kt(1,2),kt(1,1)-k);
        end
    end

    for k = size(h,2)-ndol-ndes : nlev-1
        for e = 1 : nlev-k
            h(kt(1,2)+e,kt(1,1)-(e+k)) = e*inkd+h(kt(1,2),kt(1,1)-k);
        end
    end
end

for k = 1 : nlev
    for e = k+1 : ndol
        h(y+e,x-k) = h(y+k,x-k)+e-k;
    end
end

end
end
```

2.5.1.2. Пређени пут и нова позиција

Подпрограм пређени пут и нова позиција као полазне податке има врсту и колону у којој се налази минимално f односно следећа тачка у коју ће робот отићи и која је то тренутна координата. Овде се рачуна који је то инкремент пређеног пута између два суседна пиксела и проглашава које су то кординате у које ће робот отићи у зависности од тренутних координата i и j. 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. Другим речима ако је тренутни пиксел био (5,6) нови ће бити (4,5). Исти је поступак за свих девет пиксела у матрици. У случају пиксела (2,2) није потребан рачун јер се f минимално никад неће наћи у тој позицији захваљујући великој вредности матрице g и матрице посећених тачака.

```
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
```

2.5.1.3 Подпрограм цртеж

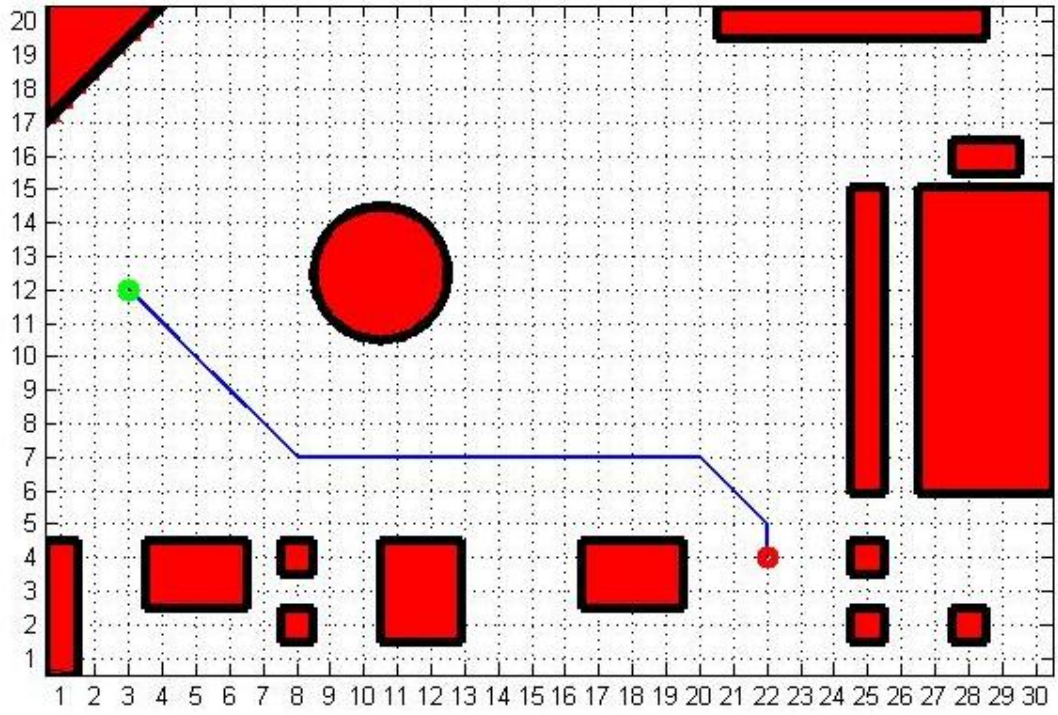
Подпрограм crtez дакле има задатак да графички прикаже окружење и путању робота како би кориснику био доступнији увид у резултате. Са првим редом је дефинисано позивање подпрограма. Одмах затим се врши приказ путање робота позивањем функције plot. Позивање се врши на основу допуњених матрица rezultat_i и rezultat_j за почетну тачку. Тако смо добили матрице x и y које су спремне за штампање.

У графичком приказу на апсциси и ординати су приказани центри пиксела, па је потребно ограничити раван приказа тако да x и y правац почињу бројање од 0,5 а завршавају са 30,5 односно са 20,5. Укључивање мреже се врши функцијом grid on. Затим је потребно дефинисати вредности на координатним правцима.

Препреке се графички приказују помоћу функције rectangle где се прва два параметра односе на координате почетка правоугаоника а друга два на величине страница. Уколико је потребно добити круг у наставку функције се казе „curvature". Проблем се јавља код цртања троугла где је потребно нацртати више паралелних линија са већом дебљином, а потом све се заокружи тамном линијом. Почетна тачка је обојена зеленом, док је циљна тачка представљена црвеном бојом.

```
function [s] = crtez(x,y,pot,krt)

plot(x,y,'LineWidth',2)

xlim([0.5 30.5])
ylim([0.5 20.5])
grid on
 s=1:30;
 p=1:20;
  set(gca,'XTick',s);
  set(gca,'YTick',p);
  hold on
  rectangle('Position',[26.5, 5.9, 4, 9.2],'LineWidth',4,'FaceColor','r')
  hold on
  rectangle('Position',[24.5, 5.9, 1, 9.2],'LineWidth',4,'FaceColor','r')

  hold on
  rectangle('Position',[0.5, 0.5, 1, 4],'LineWidth',4,'FaceColor','r')
  hold on
  rectangle('Position',[3.5, 2.5, 3, 2],'LineWidth',4,'FaceColor','r')
  hold on
  rectangle('Position',[7.5, 1.5, 1, 1],'LineWidth',4,'FaceColor','r')
  hold on
  rectangle('Position',[7.5, 3.5, 1, 1],'LineWidth',4,'FaceColor','r')
  hold on
  rectangle('Position',[7.5, 1.5, 1, 1],'LineWidth',4,'FaceColor','r')
  hold on
  rectangle('Position',[10.5, 1.5, 2.4, 3],'LineWidth',4,'FaceColor','r')
  hold on
  rectangle('Position',[16.5, 2.5, 3, 2],'LineWidth',4,'FaceColor','r')
  hold on
  rectangle('Position',[24.5, 1.5, 1, 1],'LineWidth',4,'FaceColor','r')
  hold on
  rectangle('Position',[24.5, 3.5, 1, 1],'LineWidth',4,'FaceColor','r')
  hold on
    rectangle('Position',[27.5, 1.5, 1, 1],'LineWidth',4,'FaceColor','r')
  hold on
  rectangle('Position',[27.5, 15.5, 2, 1],'LineWidth',4,'FaceColor','r')
  hold on
  rectangle('Position',[20.5, 19.5, 8, 1],'LineWidth',4,'FaceColor','r')
  hold on
  rectangle('Position',[8.5, 10.5, 4, 4],'Curvature',[1,1],'LineWidth',4,'FaceColor','r')
  hold on
  q=[0.5, 4.5];
  w=[21, 21];
plot(q,w,'LineWidth',10,'color','red')
hold on
 q=[0.5, 4.1];
  w=[20.6, 20.6];
plot(q,w,'LineWidth',10,'color','red')
hold on
q=[0.5, 3.7];
```

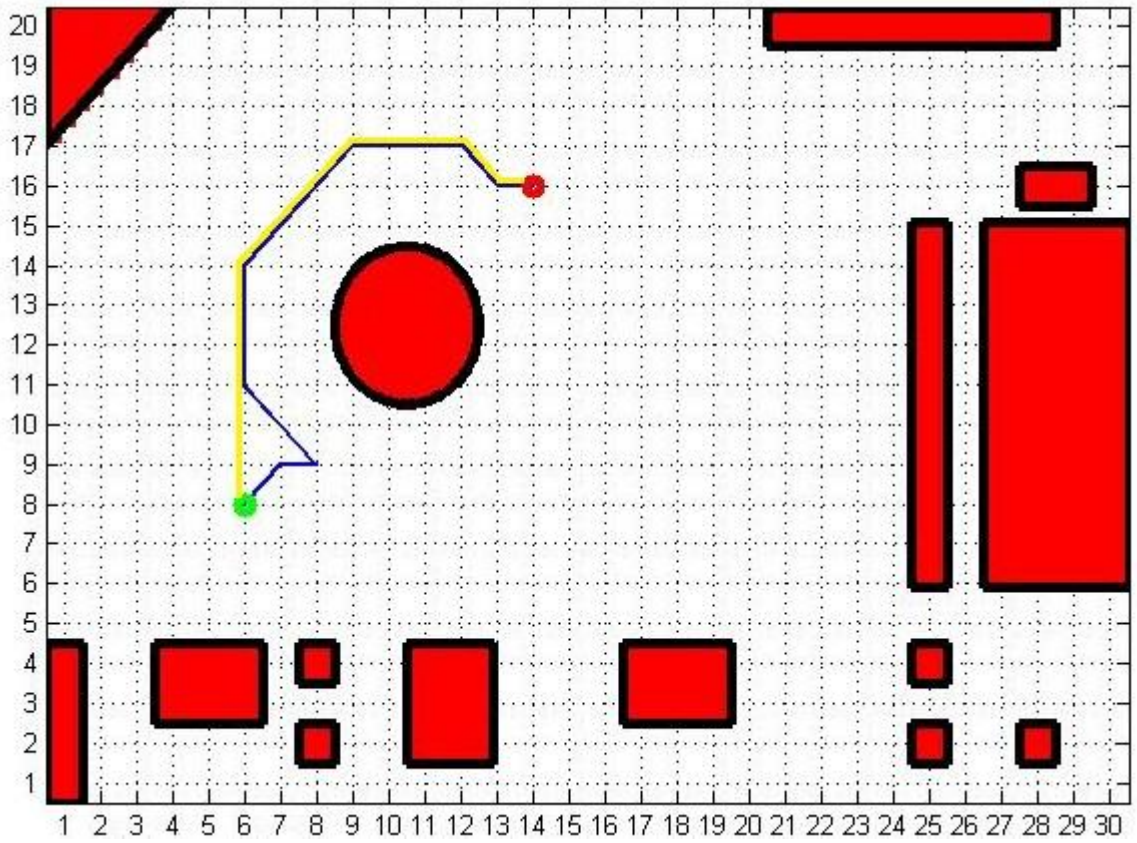
```
  w=[20.2, 20.2];
plot(q,w,'LineWidth',10,'color','red')
hold on
q=[0.5, 3.3];
  w=[19.8, 19.8];
plot(q,w,'LineWidth',10,'color','red')
hold on
q=[0.5, 2.9];
  w=[19.4, 19.4];
plot(q,w,'LineWidth',10,'color','red')
hold on
q=[0.5, 2.5];
  w=[19, 19];
plot(q,w,'LineWidth',10,'color','red')
hold on
q=[0.5, 2.1];
  w=[18.6, 18.6];
plot(q,w,'LineWidth',10,'color','red')
hold on
q=[0.5, 1.7];
  w=[18.2, 18.2];
plot(q,w,'LineWidth',10,'color','red')
hold on
q=[0.5, 1.3];
  w=[17.8, 17.8];
plot(q,w,'LineWidth',10,'color','red')
hold on
q=[0.5, 0.9];
  w=[17.4, 17.4];
plot(q,w,'LineWidth',10,'color','red')
hold on
q=[0.5, 4.5];
  w=[17, 21];
plot(q,w,'LineWidth',5,'color','black')
hold on
q=[0.5, 0.5];
  w=[17, 21];
plot(q,w,'LineWidth',4,'color','black')
hold on
q=[0.5, 4.5];
  w=[21, 21];
plot(q,w,'LineWidth',4,'color','black')
hold on
plot(pot(1),pot(2),'o','LineWidth',4,'color','green')
hold on
plot(krt(1),krt(2),'o','LineWidth',4,'color','red')
hold on
```



Слика 2.5.1.3.1: Графички приказ излаза из алгоритма А звезда

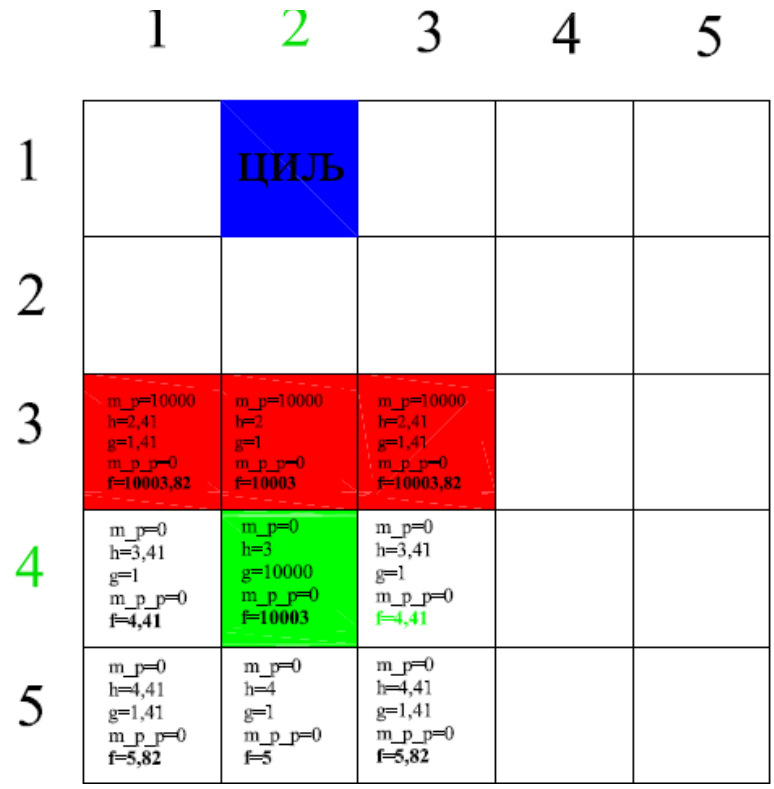
2.5.2. Дискусија и закључак

Пројектовани алгоритам у потпуности извршава задата очекивања. Постоје нека побољшања помоћу којих би алгоритам функционисао боље. На првом месту је то петља која би провлачила за сваку од матрица 3x3 све могуће комбинације не везано за минимални параметар f, а затим за сваки даљи од могућих осам пиксела (изузимајући тренутну позицију) поново претреаживала све комбинације. Оваква петља би имала безброј могућих решења па би се бирало оно које има минимум посећених тачака. Овакав алгоритам већ постоји и познатији је као Дајкстра. Разлика између алгоритма А звезда и Дајкстра дата је на слици 2.5.2.1 где се генерисана путања из алгоритма А звезда види плавом, а из Дајкстре жутом бојом.

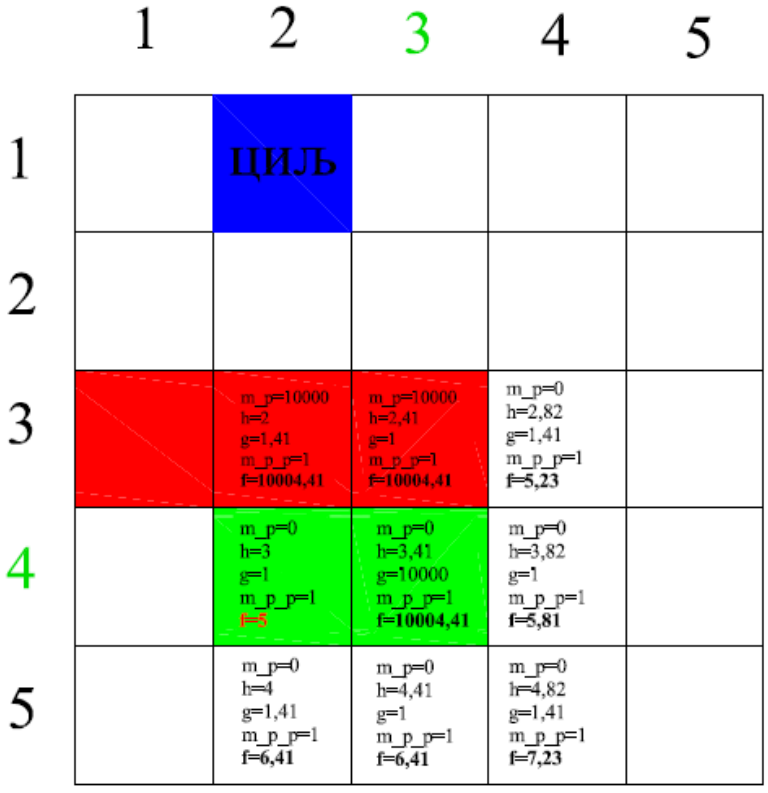


Слика 2.5.2.1: Разлика између алгоритма А звезда и Дајкстра.

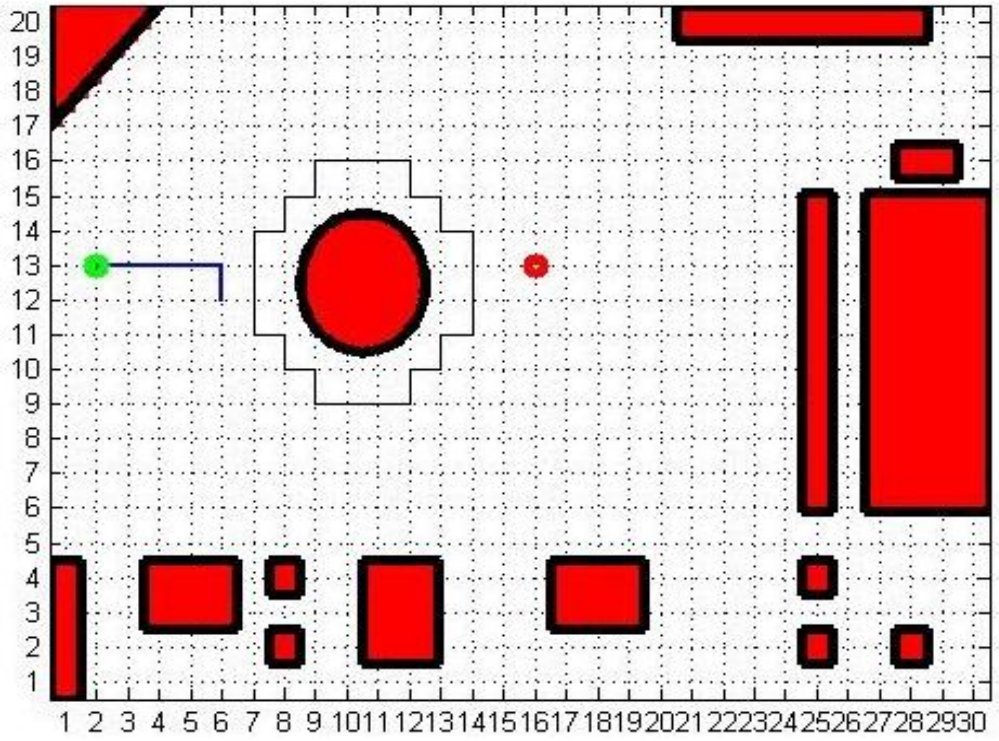
Проблеми који се могу појавити уколико се не дефинише матрица посећених тачака су ти да у неким случајевима алгоритам не би извршио задатак, односно да би се вртео стално око истих тачака. Конкретно постоје два случаја. Први је уколико би робот обилазио препреку која има више пиксела у истом правцу, а циљ се налази иза тих пиксела. По генерисању h матрице добија се иста вредност за пикселе који су лево и десно од колоне или врсте у којој се налази циљна тачка. Када алгоритам дође то тачке која је најближа циљној тачки он практично долази до колоне или врсте у којој се препрека налази. Следећа тачка у коју ће отићи је прва сусена по другој кординати, а да не удари у препреку. Таквих тачака има две при чему алгоритам насумично бира једну од њих. Следећа тачка према h матрици у коју би отишао је опет у колони или врсти циљне тачке. Овај поступак би се понављао све док се не заврши петља и робот не би обишао препреку и не би дошао до циља. Овај проблем је могуће решити и условом да се алгоритам повећа цена само за предходну тачку. Потребно је у подпрограму predjeni_put_i_nova_pozicija мењати матрицу g из пролаза у пролаз while петље и повећавати цену претходном пикселу.



Слика 2.5.2.2: Прва врста проблема уколико се не дефинише матрица посећених тачака - први корак

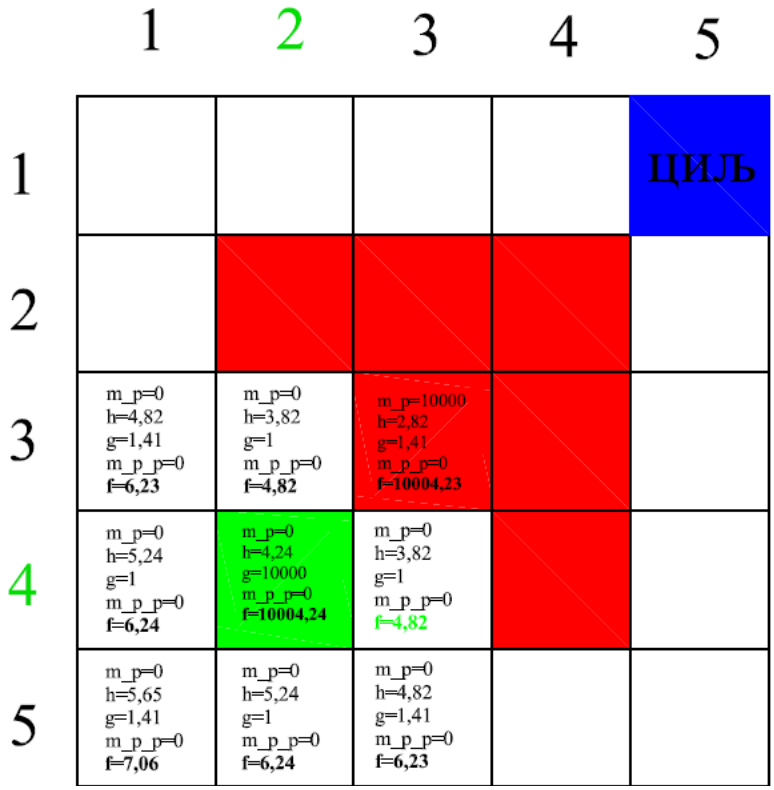


Слика 2.5.2.3: Прва врста проблема уколико се ne дефинише матрица посећених тачака - други корак

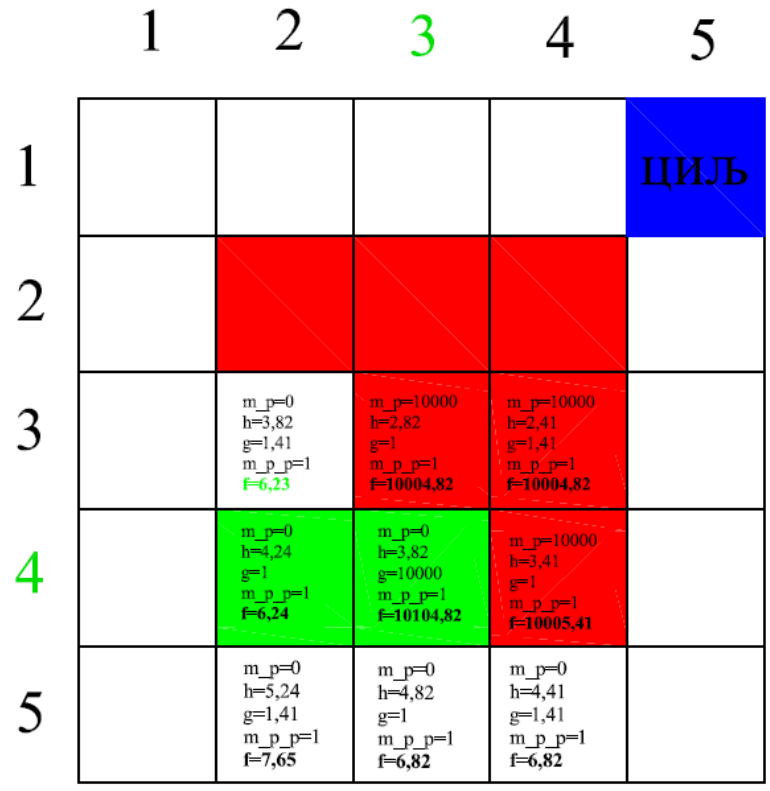


Слика 2.5.2.4: Прва врста проблема уколико се не дефинише матрица посећених тачака – графички приказ излаза из А звезде

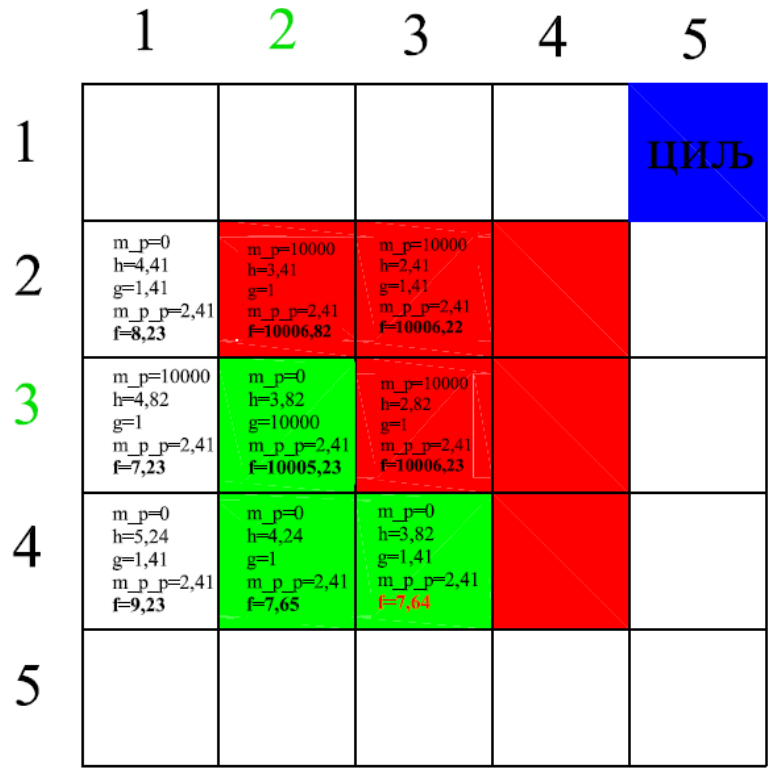
Други проблематични случај је када се тренутна и циљна тачка налазе на истој дијагонали а потребно је обићи препреку величине једног пиксела који се нашао на истој тој дијагонали, а уколико се по другој дијагонали насупрот пикселу препреке налазе још по један пиксел препрека у оба смера. И у овом као у предходном случају долази до понављања тачака и алгоритам завршава петљу не стижући до циља. Како би се оба проблема решила потребно је повећати цену посећеним пикселима и алгоритам ће тражити друге минималне вредности за f и тиме заобићи препреку. То се односи и на случај да је препрека облика ћипиличног слова п. У овом случају алгоритам ће сеући у препреку, али ће после неког времена изаћи из ње и даље стићи до циља. Случај где се не дозвољава роботу да се враћа у пиксел који је већ посећен није могућ из разлога што се робот не може вратити уколико уђе у неке од препрека, на пример уколико уђе у врло уску препреку облика слова п где има само један ред слободних пиксела.



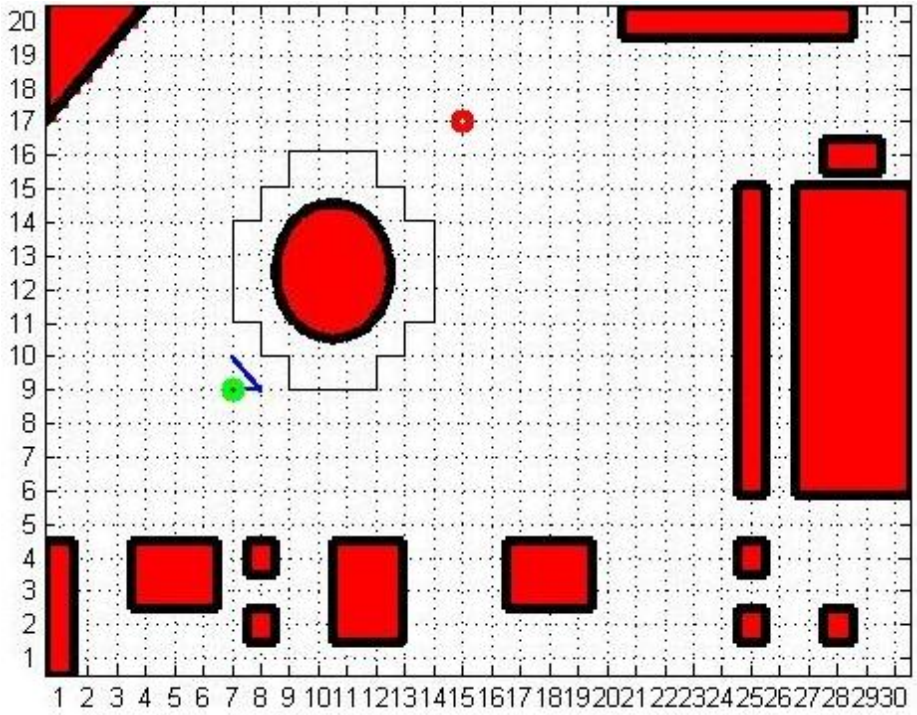
Слика 2.5.2.5: Друга врста проблема уколико се не дефинише матрица посећених тачака - први корак слика се не види цела!



Слика 2.5.2.5: Друга врста проблема уколико се не дефинише матрица посећених тачака - други корак



Слика 2.5.2.6: Друга врста проблема уколико се не дефинише матрица посећених тачака - трећи корак

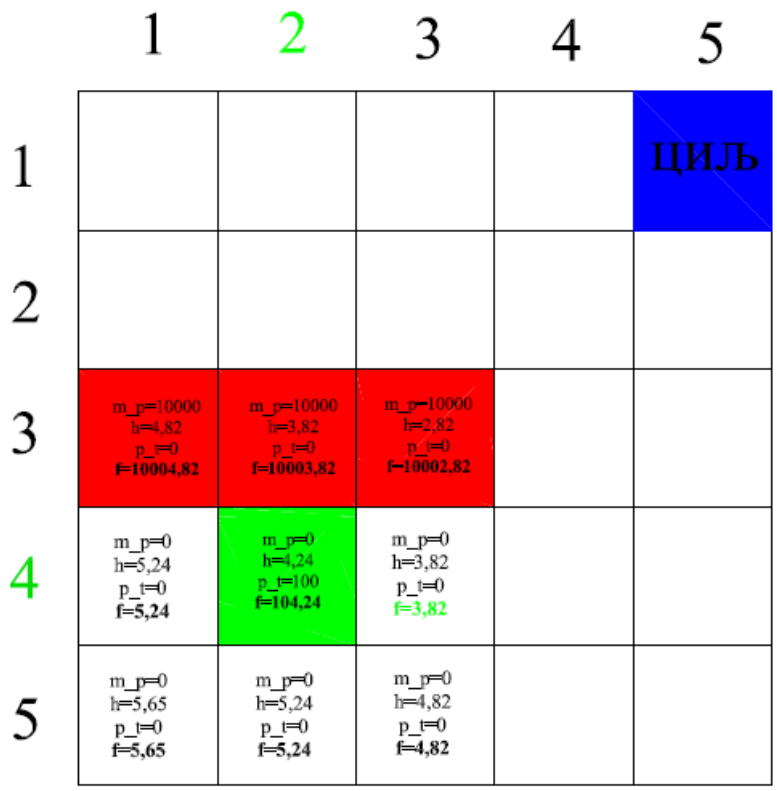


Слика 2.5.2.7: Друга врста проблема уколико се не дефинише матрица посећених тачака – графички приказ излаза из А звезде

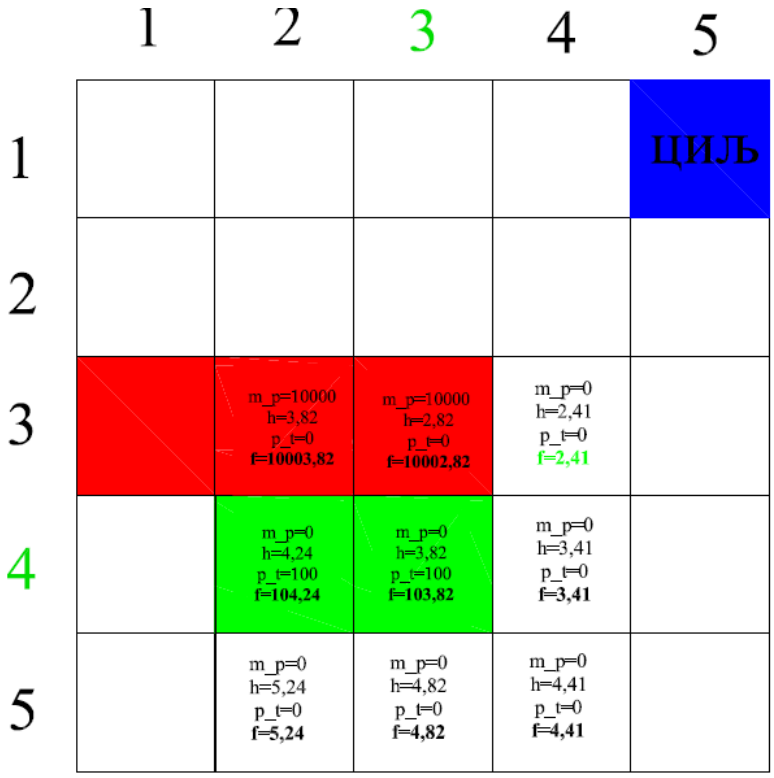
Као могуће побољшање тренутног алгоритма А* је избацивање сувишних параметара при главној формули за налажење матрице f. Наиме, пређени пут и матрица g су константе при сваком кораку рачунања те се оне могу избацити што је потврђено пуштањем алгоритма у рад без ових сабирака. Резултати који су добијени у потпуности одговарају претходно добијеним са овим сабирцима. Објашњење за то треба тражити у томе што је алгоритам А звезда општи алгоритам за различите врсте претраживања. Општи приступ у случају

претраживања путање робота би имао смисла уколико је потребно знати који је пређени пут при сваком кораку или укупан пређени пут. Формула за матрицу f би у случају избацивања ових параметра гласила:

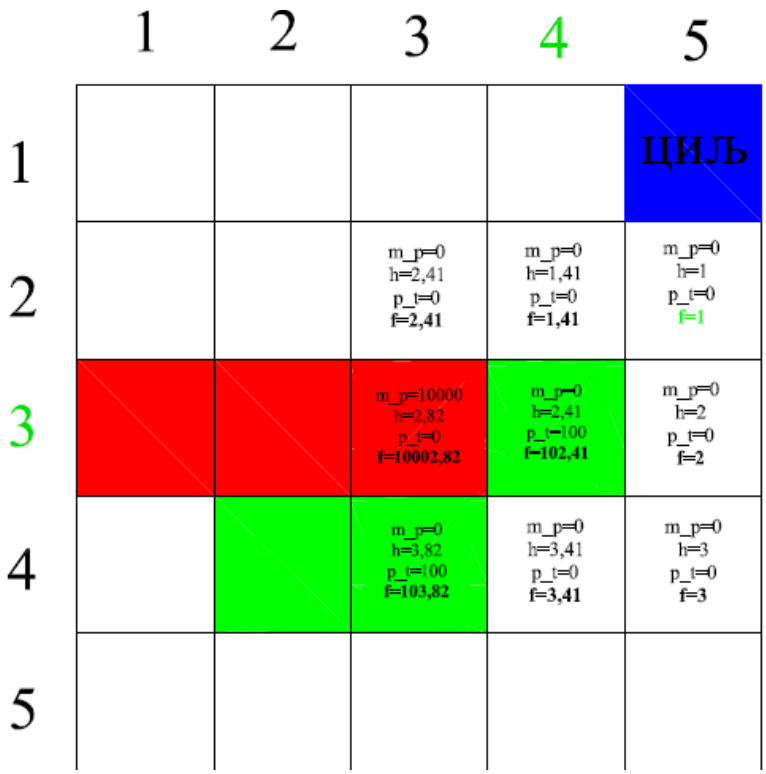
f = matrica_prepreka_3x3 + h_matrica_3x3 + posecene_tacke_3x3 (2.5.2.1)



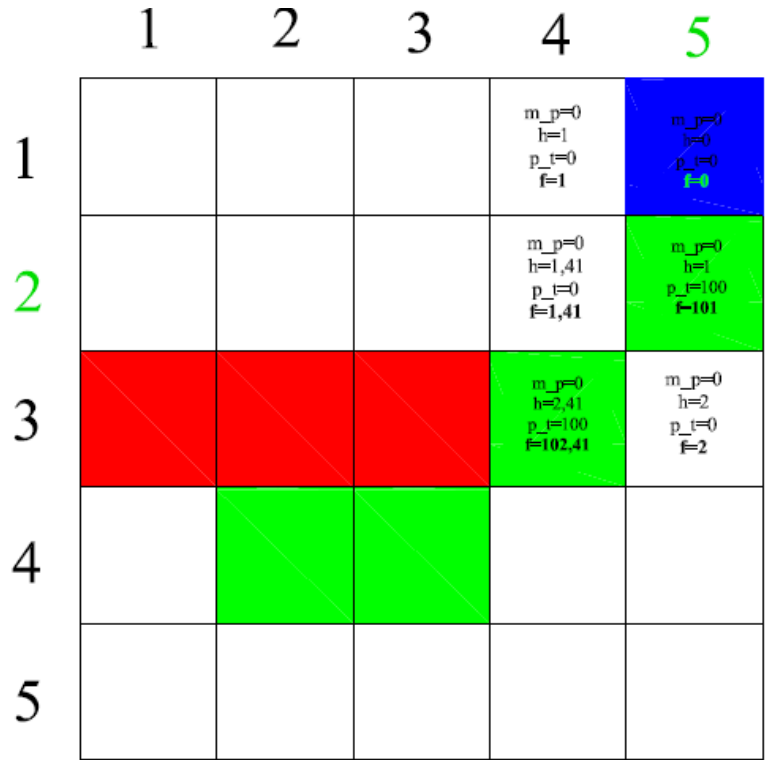
Слика 2.5.2.8: Рачун по избацивању сувишних параметара – први корак



Слика 2.5.2.9: Рачун по избацивању сувишних параметара – други корак



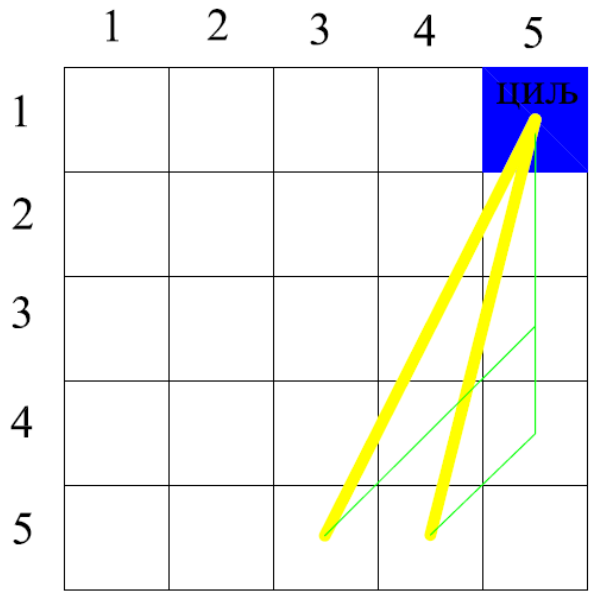
Слика 2.5.2.10: Рачун по избацивању сувишних параметара – трећи корак



Слика 2.5.2.11: Рачун по избацивању сувишних параметара – четврти корак

Доказ који је дат за ову тврдњу указује да је на изглед компликован алгоритам опште А звезде могуће упростити. Овакав вид рачуна умногоме олакшао само схватање алгоритма.

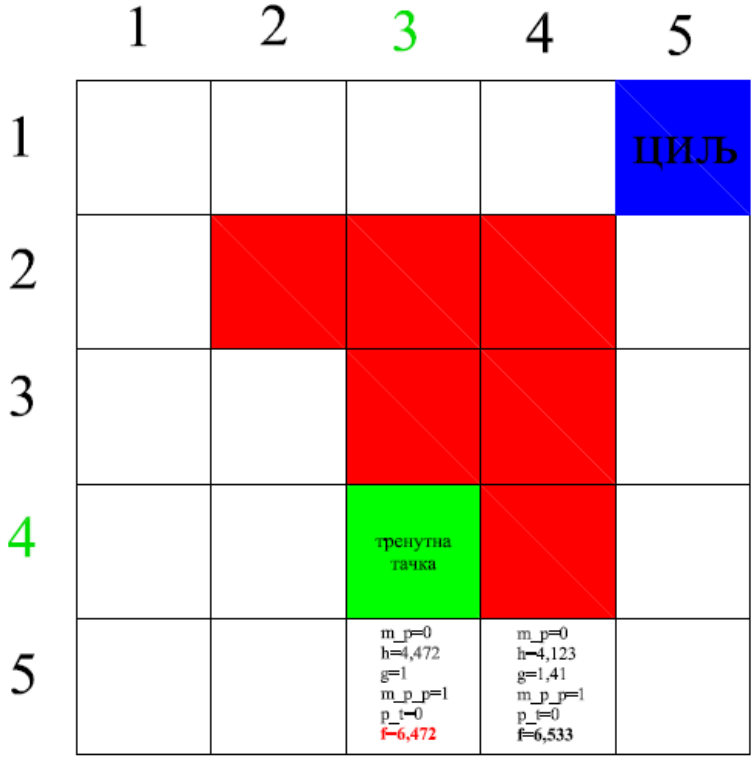
Могуће је и побољшање h матрице у смислу да се удаљеност од циљне тачке рачуна хеуристичким приступом без дискретизације по пикселима као што је приказано на слици 2.5.2.12. жутом бојом је приказана нова раздаљина, а зеленом приступ који је био објашњен у поглављу h матрица.



Слика 2.5.2.12: Хеуристички приступ без дискретизације

Међутим, све ово има смисла уколико смо већ избацили из рачуна непотребну матрицу пређеног пута, а пре свега матрицу g. Проблем који може настати не избацивањем матрице g може се видети на слици 2.5.2.12. Из тренутне тачке се неће обићи препрека на начин који је то логично.

Овим смо добили јако велико скраћење писања програма h матрице и другачије понашање, односно кретање робота. Преама еуклидској хеуристици, робот би тежио да се креће директно ка тачки, док са пројектованом h матрицом робот има тенденцију да дође на правац ка циљној тачки и потом да иде право ка њој.



Слика 2.5.2.12: Хеуристички приступ без дискретизације – уколико се не избаци матрица г

2.6. Имплементација пројектованих подсистема у пројектном решењу мобилног робота

Претходна поглавља била су намењена облашњењу принципа и решења појединих подсистема мобилног робота. Под тим се подразумевају механички, актуациони, сензорски и управљачки подсистем. У предходним поглављима описани су механички, актуациони и сензорски подсистем и на крају је остао још управљачки подсистем мобилног робота. Поред тога у овом поглављу биће изложена комплетна импленетација наведених подсистема у једну целину коју називамо интелигентним мобилним роботом.

2.6.1 Управљачки систем мобилног робота

Управљачки систем мобилног робота је базиран на PC рачунару и контролној јединици LEGO Mindstorms NXT пакета. Контролна јединица је већ детаљно описана и њено упутство за употребу и руковање се може наћи у LEGO пакету.

PC рачунар је у овквој концепцији главни управљачки систем мобилног робота јер је доступан свима и сем софтверског пројектовања управљања никакве друге интервенције нису потребне, што додатно олакшава посао јер хардверско пројектовање управљачких система захтева посебна знања и вештине.

2.6.2. Управљање кретањем робота

Роботом се управља помоћу програма који је написан у MatLAB окружењу. Да би била могућа контрола мобилног робота, MatLAB окружењу је додат RWTH Mindstorms NXT toolbox пакет [10]. Овај командни пакет је развијен на RWTH Aachen Универзитету као студентски пројекат под називом *„MATLAB meets LEGO Mindstorms“*. Овај пакет се једноставно снима на меморију рачунара и његова локација се повезује са локацијама програмске претраге MatLAB окружења. Оваква платформа ради преко оригиналног софтвера LEGO MINDSTORMS NXT у коме мора да се конфигурише, али даља употреба оригиналног LEGO софтвера није потребна.

Мотори се контролишу помоћу сета команди које су дате у табели 2.5.2.1 заједно са одговарајућим облашњењима. Мотори су прикачени на портове A и B контролне јединице LEGO Mindstorms NXT. Овај сет команди издаје се редно као што је то у табели 2.5.2.1.

Табела 2.5.2.1: *Команде за контролу кретања мобилног робота*	
КОМАНДА	ОПИС
`StopMotor('all', 'off');`	Стоп свих мотора
`ResetMotorAngle(MOTOR_A);`	Ресетовање мерног система мотора А
`ResetMotorAngle(MOTOR_B);`	Ресетовање мерног система мотора В
`SetMotor(MOTOR_A);`	Постављање мотора А као главни мотор
`SyncToMotor(MOTOR_B);`	Постављање мотора В као пратећи мотор
`SetPower 20`	Дефинисање снаге мотора 0-100
`SetTurnRatio 100`	Дефинисање окретања мобилног робота 100 – окретање у месту у једну страну -100 – окретање у месту у другу страну
`SetAngleLimit 60`	Дефинисање дужине окретања мотора >0 - број који нема јединицу мере већ представља управљачку величину
`SendMotorSettings`	Слање подешавања
`WaitForMotor(MOTOR_B);`	Чекање пратећег мотора
`MA = GetMotorSettings(MOTOR_A);`	Очитавање вредности угла са енкодера А
`MB = GetMotorSettings(MOTOR_B);`	Очитавање вредности угла са енкодера В

Пре ових команди морају се издати две команде за отварање комуникације између рачунара и робота, а на крају команди из табеле 2.5.2.1. издаје се команда за затварање комуникације са роботом. Те три команде дате су у табели 2.5.2.2.

Табела 2.5.2.2*: Команде за комуникацију рачунара и мобилног робота*	
КОМАНДА	ОПИС
`h = COM_OpenNXT('USB.ini', 'check');`	Отварање комуникације
`COM_SetDefaultNXT(h);`	Подешавање комуникације
`COM_CloseNXT('USB.ini', 'check');`	Затварање комуникације

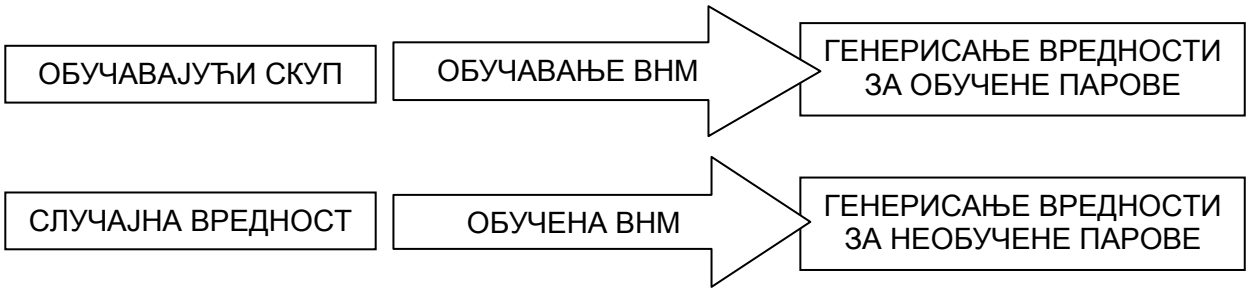
На крају, комплетан код за управљање моторима заједно са очитавањем углова и рачуном пређеног пута левог и десног точак дат је у наставку.

```
StopMotor('all', 'off');
ResetMotorAngle(MOTOR_A); ResetMotorAngle(MOTOR_B);
SetMotor(MOTOR_A); SyncToMotor(MOTOR_B);
SetPower 20
SetTurnRatio (trrot);
SetAngleLimit (alrot);
SendMotorSettings
pause(0.5)
WaitForMotor(MOTOR_B);
pause(0.5)
MA = GetMotorSettings(MOTOR_A);
MB = GetMotorSettings(MOTOR_B);
dsl = MA.Angle/57.3*r;            %predjeni put levog tocka
dsr = MB.Angle/57.3*r;            %predjeni put desnog tocka
```

2.6.3. Обучавање окретања мобилног робота помоћу вештачке неуронске мреже

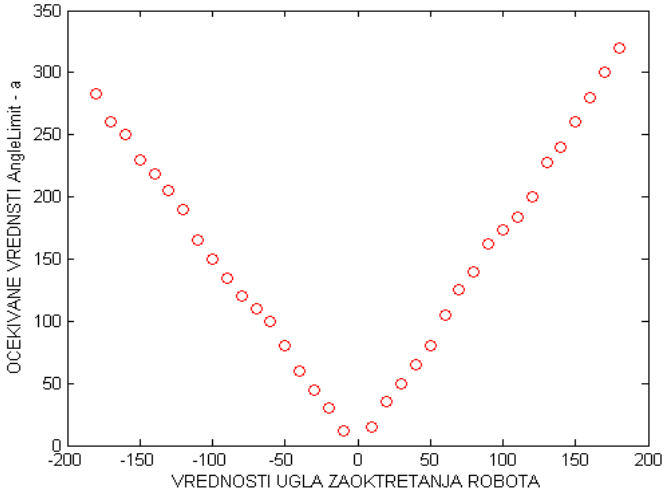
Већ је речено да су вештачке неуронске мреже добро решење за решавање нелинеарности у систему. Проблем који су вештачке неурнонске мреже успешне решиле у пројекту је одређивање вредности за *AngleLimit.* Ову вредност користи *NWTX toolbox* у *MatLab* софтверу. Вредност представља управљачку команду за број обртаја излазног вратила мотора броја обртаја мортора. У зависности од ове вредности робот се заокреће за одређен угао у глобалном координатном систему.

Експериментално су утврђене вредности *AngleLimit* – а за углове у табели 2.6.3.1. Подаци из таблице представљају улазни и излазни обучавајући вектор за неуронску мрежу. Идеја је да се мрежа тренира за обучавајуће парове из таблице, а да затим генерише одговарује вредности *AngleLimit* – а за необучене вредности жељеног угла заокретања робота [2]. Концепт је приказан на слици 2.6.3.1.



Слика 2.6.3.1: Концепт примене ВНМ за одређивање вредности угла рорације излазног вратила мотора

Обучавајући скуп је приказан у табели 2.6.3.1. Анализом обучавајућег вектора је утврђена нелинеарна зависност између вектора улаза и вектора излаза. Графички се то показује на слици 2.6.3.2.



Слика 2.6.3.2: Нелинеарност обучавајућег вектора

Даље је лако уочити да потребне вредности које се добијају за *AngleLimit* нису по апсолутној вредности једнаке за промену оријентације робота у позитивном и негативном математичком смеру. Ово практично значи да је неопходно да обучавајући скуп узима вреднсти од -180^o до 180^o. Немогуће је одредити вредност *AngleLimit* за један угао, а потом променити предзнак за заоктретање робота у другу страну.

Табла 2.6.3.1: *Обучавајући парови за ВНМ*									
Редни број мерења	1	2	3	4	5	6	7	8	9
Вредност жељеног угла заокретања робота	10	20	30	40	50	60	70	80	90
Потребна вредност AngleLimit	15	35	50	65	80	105	125	140	162
Редни број мерења	10	11	12	123	14	15	16	17	18
Вредност жељеног угла заокретања робота	100	110	120	130	140	150	160	170	180
Потребна вредност AngleLimit	173	184	200	228	240	260	280	300	320
Редни број мерења	19	20	21	22	23	24	25	26	27
Вредност жељеног угла заокретања робота	-10	-20	-30	-40	-50	-60	-70	-80	-90
Потребна вредност AngleLimit	12	30	45	60	80	100	110	120	135
Редни број мерења	28	29	30	31	32	33	34	35	36
Вредност жељеног угла заокретања робота	-100	-110	-120	-130	-140	-150	-160	-170	-180
Потребна вредност AngleLimit	150	165	190	205	218	230	250	260	283

Приступа се одређивању оптималне архитектуре и других параметара вештачке неуронске мреже. У циљу проналажења оптималне, тестирано је више мрежа са различитим параметрима. Варирани су број скривених слојева, број неурона у сваком скривеном слоју, активационе функције неурона, као и вредност параметра учења [2]. Поред вредности које су вариране, константне током целокупног процеса остају:

- *Param.show* = 50; број итерација након ког се приказује промена. Ова вреднст не утиче на перформансе ВНМ, већ је значајна за цртање
- *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
- *max_fail* = 10; број валидација на којима је вештачкој неуронској мрежи дозвољено да не задовољи захтевану грешку
- *trainlm*; све мреже користе Левенберг – Маркеов алгоритам обучавања

Експеримент је приказан у табели 2.6.3.2. Прикзане су све мреже које су тестиране са свим факторима релевантним за одређивање оптималне.

Табела 2.6.3.2: *Обучаване мреже за заокретање робота*

Број скривених слојева	Редни број	Функције у скривеним слојевима	Број неурона у скривеним слојевима	Очекивана грешка	Параметар учења	Број итерација
1	1	Tansig	20	0.01	0.1	590
	2	Tansig	25	0.5	0.9	39
	3	Purelin	26	0.1	0.5	1000[5]
	4	Tansig	50	10^{-5}	0.2	95
2	5	Purelin tansig	26 - 50	0.9	0.5	120
	6	**Purelin tansig**	**20 - 25**	**0.001**	**0.1**	**18**
	7	Tansig Tansig	15 - 30	0.1	0.5	69
	8	Tansig Tansig	40 - 10	10^{-5}	0.01	36
3	9	Tansig Tansig tansig	26 - 50 - 20	0.5	0.5	487

Као оптимална је изабрана вештачка неуронска мрежа под редним бројем 6. Она има по један неурон у улазном и излазом слоју и два скровена слоја са 20 и 25 неурона респективно. Очекивана грешка је 0.001, а параметар учења је 0.1. Мрежа је постигла жељени резултат већ после 18 итерација. Мали број итерација је кључан за избор ове мреже [2]. Овако мали број итерација је веома битан због софтверске имплементације мреже. Обучена мрежа има јако мало време одзива, што је важно због рада имплементираног решења у реалном времену.

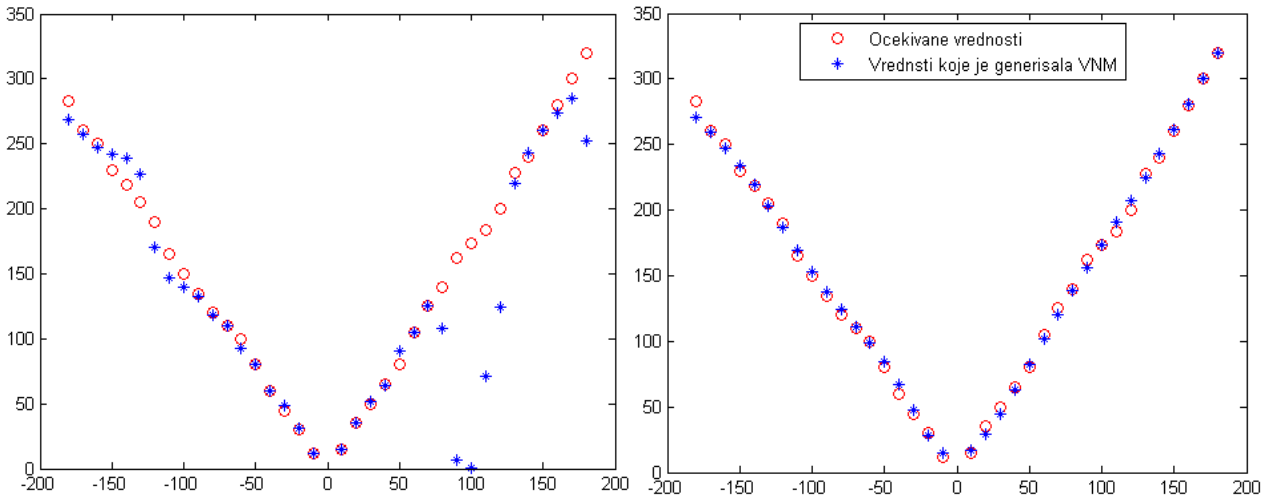
[5] Мрежа чак ни после 1000 итерација није постигла задату грешку

Какви су резултати? Пропуштањем вредности за жељени угао кроз вештачку неуронску мрежу за обучене парове, и упоређивајући генерисане резултете са жељеним добијају се вредности грешке које се крећу од 0.1 до 2.1% (табела 2.6.3.3).

Табела 2.6.3.3: *Вредности параметара одабране мреже за обучаване вредности*

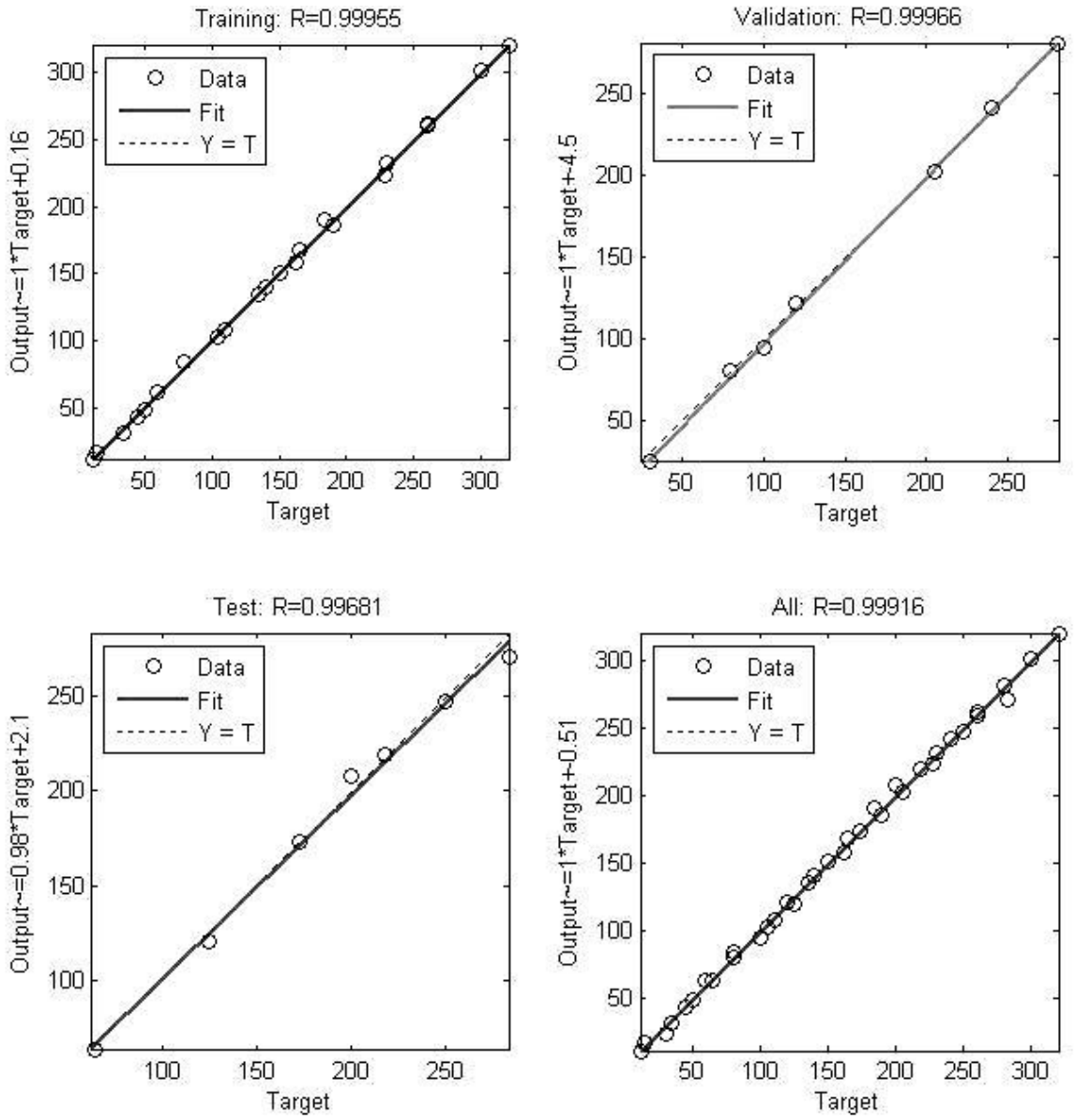
Вредност угла	30	90	150	-30	-90	-150
Очекивана вредност	50	162	260	45	135	230
Вредност добијена из мреже	49.95	164.3432	261.4234	45.3278	137.7468	234.2469
Грешка [%]	0.1	1.44642	0.547462	0.728444	2.034667	1.846478

Графички приказано (слика 2.6.3.3.а) се види колико добро мрежа генерише излазе. Црвени кругови представљају обучавајуће парове док су плаве звездице вредности које се добију када се жељена вреднст пропусти кроз вештачку неуронску мрежу. Добро генерисане вредности се одликују преклапањем звездица и кругова. Што је преклапање веће, то су вреднсти које мрежа генерише боље, односно тачније. У циљу упоређивања, на слици поред је приказан график мреже која не даје добре резултате (слика 2.6.3.3.б).



Слика 2.6.3.3: а) добри и б) лоши резултати генерисаних ВНМ

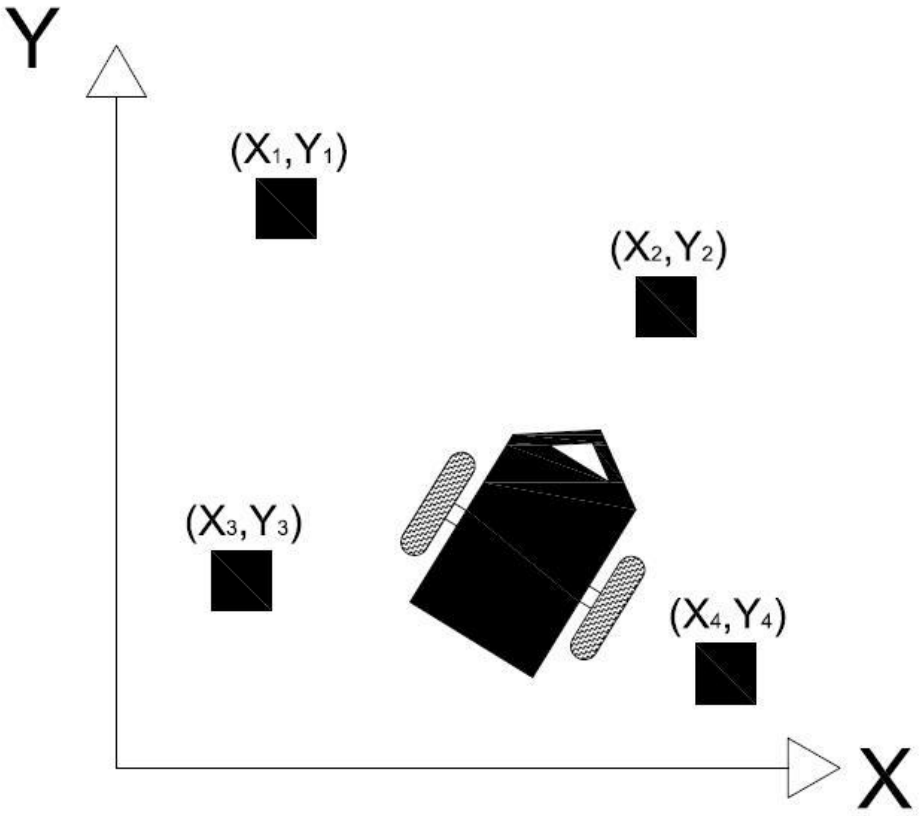
Још једна од каракретистика која приказује перформансе ВНМ је регресија. На слици 2.6.3.4 је приказана реграсија одабране вештачке неуронске мреже. Регресија се оцењује тако што се узимају вредности из обучавајућег скупа, а потом се од мреже тражи да генерише резултате. Кругови треба да буду постављени око линије тако да грешка буде минимална. Приказане су генерисане вредности за обучавајуће парове који су искоришћени за обучавање, валидацију и тестирање [2].



Слика 2.6.3.4: Регресија одабране ВНМ

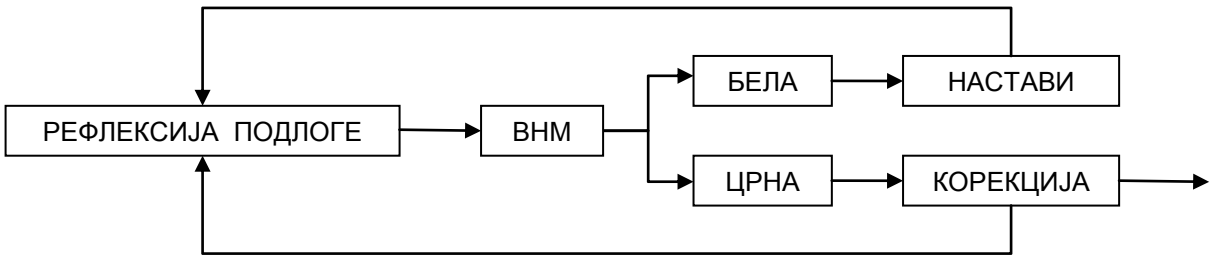
2.6.4. Обучавање светлосног сензора помоћу вечтачке неуронске мреже

Референтне тачке се налазе у окружењу и служе за кориговање тренутне позиције робота. Ове тачке су црне боје, и разликују се од окружења, које је бело. Када робот светлосним сензором пређе преко неке од црних тачака, неопходно је да је препозна, а затим коригује своју позицију. Помоћу алгоритма Линеатизован Калманов филтер локализације, на основу свог уверења о томе где се налази (сензорске информације) и мапе црних тачака (унапред дефинисаних позицијом), робот ће бити у стању да препозна о којој се тачки ради (слика 2.6.4.1) [5].



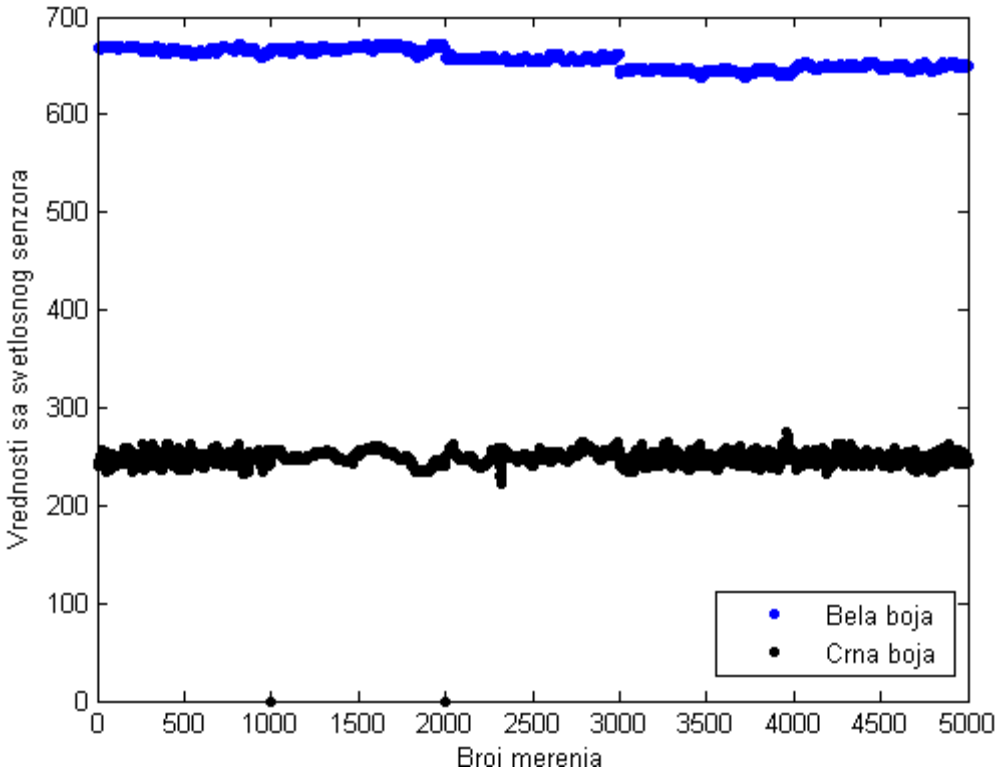
Слика 2.6.4.1: Контролне тачке за кориговање позиције робота

Вештачка неуронска мрежа у овом проблему има улогу класификатора. Мрежа треба да обради сигал добијен са светлосног сензора, а да на излазу генерише да ли се ради о црној или белој боји. Ако се робот нађе изнад једне од контролних тачака, светлосни сензор ће детектовати мању рефлексију, и када се тај податак проследи ВНМ, она ће на излазу генерисати црну боју. Општи алгоритам је приказан на слици 2.6.4.2. Овај алгоритам је активан током целог извршавања кретања.



Слика 2.6.4.2: алгоритам препознавања контролних тачака

Индетично као у претходном делу, неопходно је прво дефинисати обучавајуће парове. Потребно је прикупити сензорске информације о што више различитих белих и црних подлога. Снимања су вршена за пет раличитих белих и још пет различитих црних подлога. У сваком снимању има по хиљаду узорака. Да би узорци били што бољи, снимање је извршено док се робот кретао. Овакво прикупљање обучавајућих парова симулира рад у реалном окружењу. Наиме, робот се све време креће, па ће и подлогу све време да снима у стању кретања. Графички приказ добијених вредности за сва мерења је приказан на слици 2.6.4.3.



Слика 2.6.4.3: обучавајући парови за вештачку неуронску мрежу

Подаци о боји уствари представљају податке о рефлексије о подлогу. Податак о рефлексији се из сензора шаље управљачкој јединици у виду 1024 – битног податка. Вредности од 0 до 1023 представљају одговарајућу вредност рефлексије. Према слици се може закључити да бела боја има рефлексију од 600 до 700, а црна од 200 до 300.

У ова мерења је укључен и шум. Шум се елиминише нормализацијом података [4]. Подаци се скалирају, да би се утицај шума на мерење смањио што је више могуће. Скалирање се врши тако да вредност мерења за црну боју добије вредности које су границама -1 ≤ X ≤ 0, док се скалирањем вредности за белу боју добију вредности које су у границама 0 ≤ X ≤ 1. Скалирање података се врши помоћу формуле:

$$x_i = \frac{x - \frac{\max(x) + \min(x)}{2}}{\frac{\max(x) - \min(x)}{2}} \qquad (2.6.4.1)$$

Велики проблем у току прикупљања експерименталних података је било осветљење. Наиме, подаци за сваку од боја су доста варирали у зависности од временских прилика, локације на којој се робот налази, доба дана и слично. Овај проблем је ублажен коришћењем вештачког осветљења. Потпуно би могао бити превазиђен додавањем осветљења у непосредној близини светлосног сензора, као и прикупљањем података на тачној локацији на којој би робот радио.

Вештачке неуронске мреже су добре колико је добар њихов обучавајући скуп. Током даљих експеримената, утврђено је да одабрана вештачка неуронска мрежа греши. Разлог грешке је била вредност са сензора која им се проследи. При нижем осветљењу, вредности рефлексије се смањују. Тако се дешава да се рефлексија беле боје приближи граници разграничења. Ова вредност може бити проблем чак и поред скалирања улазних података, и поред робусне мреже.

Још један од разлога за скалирање обучавајућег скупа је повећавање робусности мреже. Наиме, мрежа оперише са много мањим бројевима, па је и могућност грешке знатно смањена. Такође, ако се појаве вредности које су на линији разграничења, мрежа ће лакше да реши која је боја детектована. У циљу још већег поједностављења, вреднсти излазног вектора за белу боју су узете као 1, а за црну -1.

Слично као код тражења оптималне структуре ВНМ за заокретање робота, и овде се приступа развијању модела за имплементацију. Експериментално се утврђује оптимална топологија, као и параменти учења, а константни остају:

- *Param.show* = 50; број итерација након ког се приказује промена. Ова вреднст не утиче на перформансе ВНМ, већ је значајна за цртање
- *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
- *max_fail* = 10; број валлидација на којима је вештачкој неуронској мрежи дозвољено да не задовољи захтевану грешку
- *trainlm* све мреже користе Ливенберг – Марке –ов алгоритам обучавања.
- Захтевана грешка 10^{-10}

Примећује се да су фактори исти као код обучавања ВНМ за заокретање робота. Разлог за ово је што ови фактори немају превелик утицај на сам процес обучавања. Ни једна мрежа није пала на дозвољеном броју валидација, и све су конвергирале ка минималној грешки (како ће касније бити показано). Приказ тестираних мрежа је у табели 2.6.4.1.

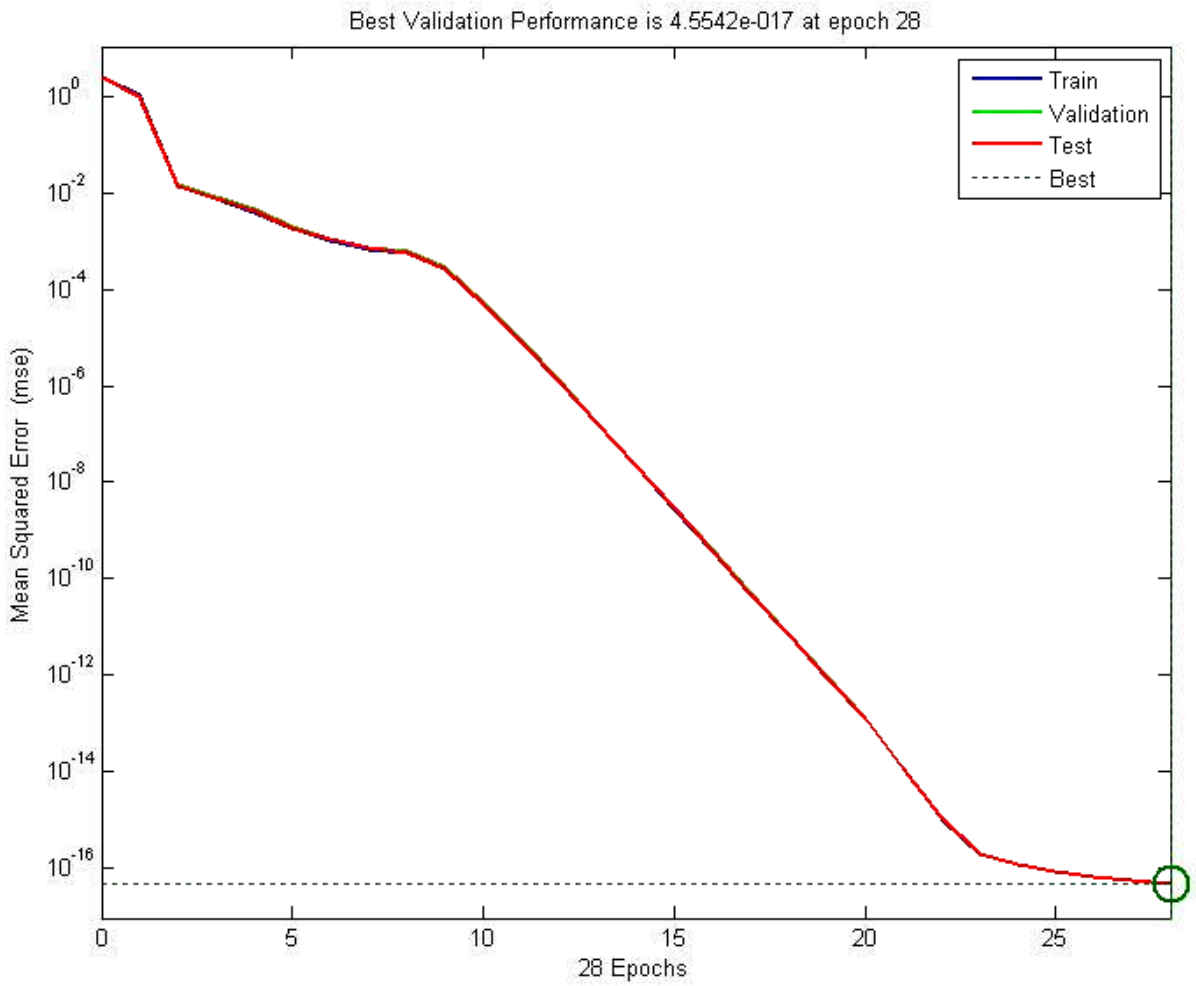
***Табела 2.6.4.1**: Обучаване мреже за детекцију контролних тачака*

Број неурона у скривеном слоју	Параметар учења	Број итерација до остварења минималне грешке
2	0.1	41
	0.5	**27**
	0.05	31
3	0.1	83
	0.5	49
	0.05	46
5	0.1	174
	0.5	35
	0.05	77

Како је поменуто, свака од тестираних ВНМ је конвергирала до решења. Конвергенција је најбрже напредовала код мреже са два неурона у скривеном слоју, и са параметром учења 0.5. ова ВНМ је узета као оптимална. Приказ конвергенције ка решењу, према методи најмањег квадрата грешке (*MSE method*) је приказан на слици 2.6.4.4. У циљу валидације узете су вреднсти са сензора и пропуштене кроз ВНМ. Очекиване вредности, генерисане вредности и грешка су приказани у табели 2.6.4.2. У циљу превазилажења

проблема препознавања боја је могао бити и примењен перцептрон, који се много бољи показао за решавање проблема класификације.

***Тебела 2.6.4.2**: Упоредне вредности очеиваних резултата и резултата генерисаних из ВНМ*								
Боја	Црна				Бела			
Улазни податак	670	685	690	693	230	257	260	300
Очекивана вредност	-1	-1	-1	-1	1	1	1	1
Генерисана вредност	-0.9923	-0.9928	-1.0017	-1.009	0.9987	0.9815	1.003	1.021



Слика 2.6.4.4: Конвергенција ВНМ ка решењу

2.6.5. Обрада података са светлосног сензора и корекција позиције робота

Подаци са сензора се обрађују кроз програм посебним алгоритмом. За то се користи Калманов филтер. По очитавању вредности за светлосног сензора врши се поменуто скалирање вредности и скалирана вредност се пропушта кроз обучену вештачку неуронску мрежу. На крају се за вредност излаза из мреже врши провера вредности и уколико је вредност негативна, тј. детектована је црна боја, врши се корекција тренутне позиције мобилног робота. Код помоћу којег се очитава вредност сензора, скалира, позива обучена мрежа и врши корекција је:

```
%======Korekcija polozaja - crne tacke======
s = GetLight(SENSOR_2);                          %ocitavanje
y = 2*(s-min_black)/(max_white-min_black) -1;  %skaliranje
o = sim(netff,y)                                 %simuliranje
if o < 0
    [x,C] = ITS_data_association(x,C,Q,map);   %korekcija
end
%=========================================
```

Корекција путање се извршава позивом функције `ITS_data_association` која за улаз има вредност вектора стања робота, матрицу коваријансе, матрицу шума мерења и мапу црних тачака у виду матрице са координатама. Програмски код са побољшањем дат је у наставку текста.

```
function [x,C] = ITS_data_association(x,C,Q,map)
xls = 2;   %rastojanje senzora o dose obrtanja robota po x
yls = 0;   %rastojanje senzora o dose obrtanja robota po y
N = [];    %prazna matrica
e = 1;
for k = 1 : size(map,1)
    [zp,Hp] = predict_measurements_with_lihgt_sensors(x,map(k,:), xls, yls);  %predikcija

    z = map(k,:)';               %k-ta crna tacka
    S= Hp*C*Hp' + Q;             %predvidjena matrica kovarijansi merenja
    nis= (z-zp)'*inv(S)*(z-zp);  %
    N = [N;nis]
end
for g=1:size(N)
    if N(g,:)==min(N)
        j=g                      %broj vrste sa najmanjom vrednoscu
    end
end
if isempty(j)
    return
else
    Z = map(j,:)'
    [x, C] = EKF_update_light_sensors(x, C, Z, zp, Hp, Q)   %korekcija
end
```

Такође у наставку текста дате су две функције које се позивају у претходној функцији, а служе за претварање координата сензора у координатном систему робота у глобални координатни систем и корекцију према алгоритму Калмановог филтера из табеле 2.3.1.1.

```
function [pose, Cov] = EKF_update_light_sensors(pose, C, z, zp, H, Q)
if isempty(zp) || isempty(z)
    return
else
    invec = z - zp;
    CHt = C*H';
    S = H*CHt + Q;
    Sinv = inv(S);
    K = CHt*Sinv;
    up= K*invec;
    pose = pose + up;
    Cov = (eye(3) - K*H)*C;
end
```

```
function [varargout] = predict_measurements_with_lihgt_sensors(pose,map, xls, yls)
if isempty(map)
    varargout{1} = [];varargout{2} = [];
    return;
end;

dx = pose(1) + xls*cos(pose(3)) - yls*sin(pose(3));
dy = pose(2) + xls*sin(pose(3)) + yls*cos(pose(3));

h11 = 1; h12 = 0; h13 = -xls*sin(pose(3)) - yls*cos(pose(3));
h21 = 0; h22 = 1; h23 = xls*cos(pose(3)) - yls*sin(pose(3));
h = [h11 h12 h13; h21 h22 h23];

zp = [dx;dy];
h = h;

varargout{1} = zp; varargout{2} = h;
```

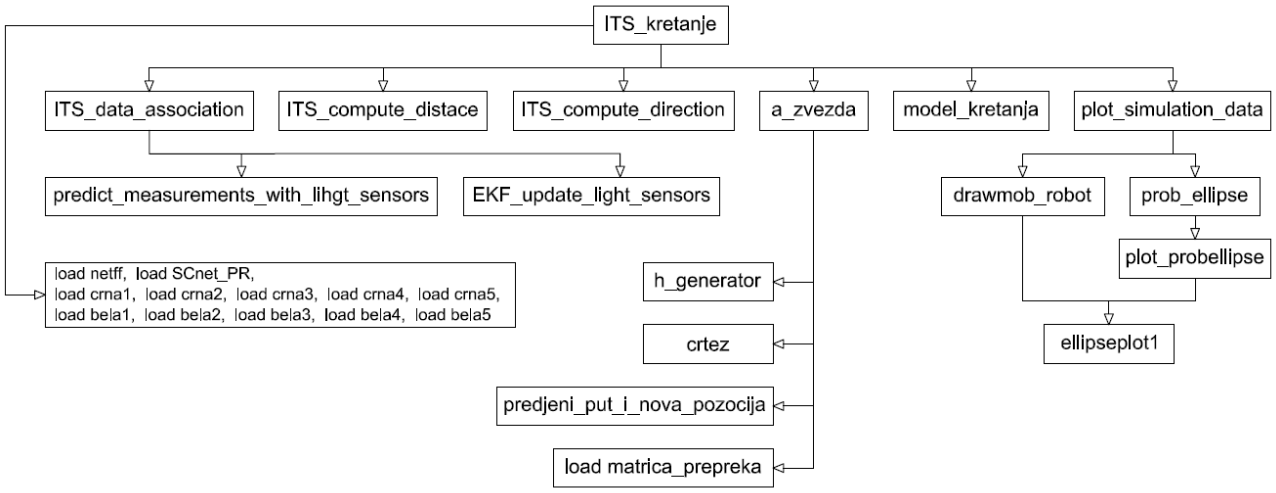
Функција `predict_measurements_with_lihgt_sensors` има задатак да претвори координате сензора у координатном систему робота у глобални координатни систем и да јакобијан функције h (у Калмановом филтеру) у односу на положај мобилног робота.

Функција корекције `EKF_update_light_sensors` има задатак да изврши корекцију вектора стања и да израчуна нову матрицу коваријанси према алгоритму корака корекције Калмановог филтера.

По завршетку оваквог рачуна добијају се нове, кориговане вредности вектора стања мобилног робота у радном окружењу, чиме се постиже смањење акумулиране грешке током рада.

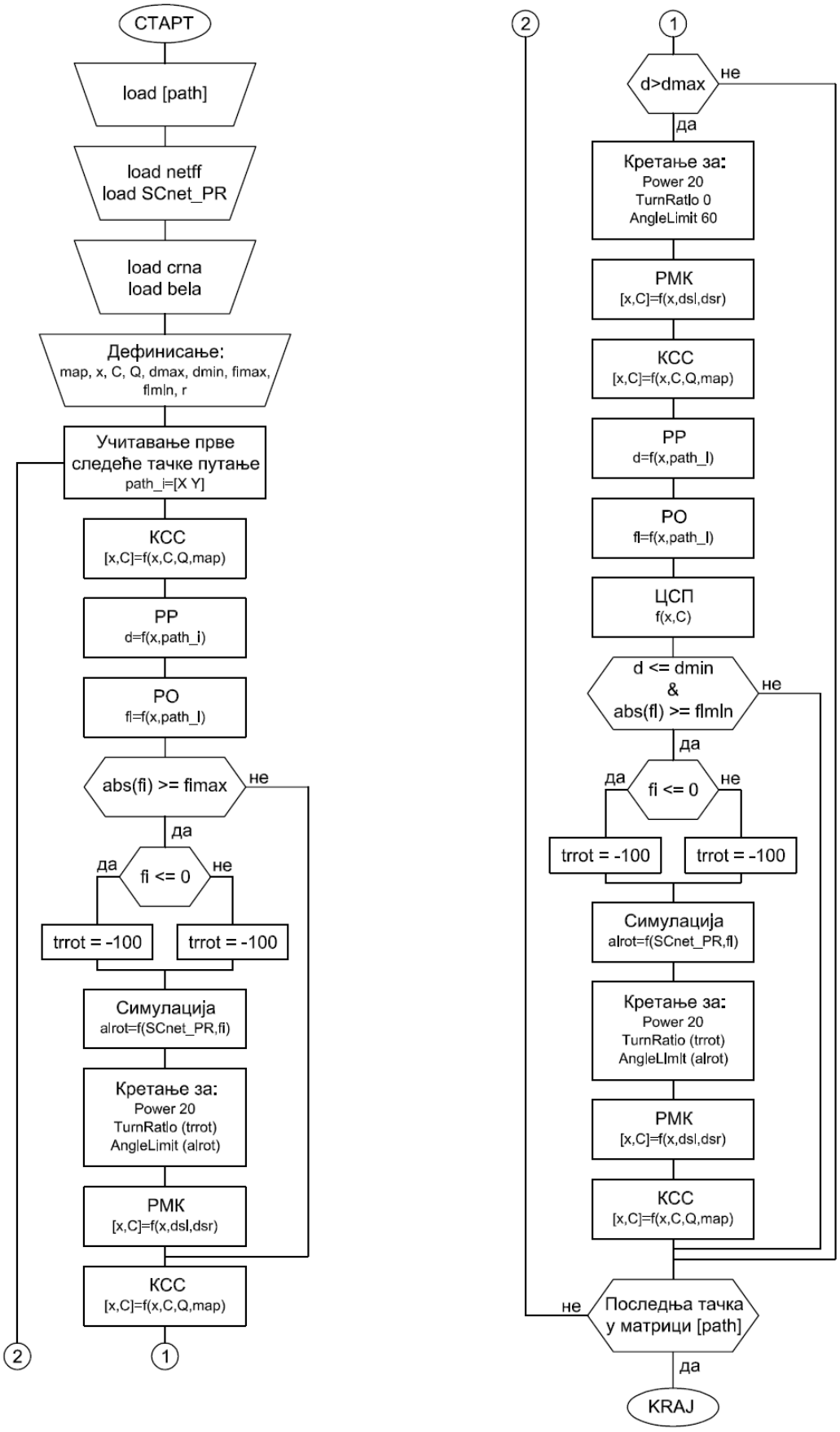
2.6.6. Потпуни програмски код за управљање мобилног робота

Програмски код за управљање радом мобилног робота, према предходно добијеним путањама из алгоритма A*, састоји се из главног програма, низа подпрограма и сачуваних матрица које се учитавају приликом покретања програма. Преглед главног програмског кода `ITS_kretanje` дат је на слици 2.6.6.1 заједно са везама са подпрограмима.



Слика 2.6.6.1: Програмски дијаграм

Алгоритам главног програма управљања, `ITS_kretanje`, дат је на слици 2.6.6.2, и позива се на подпрограме као што је дато на слици 2.6.6.1.

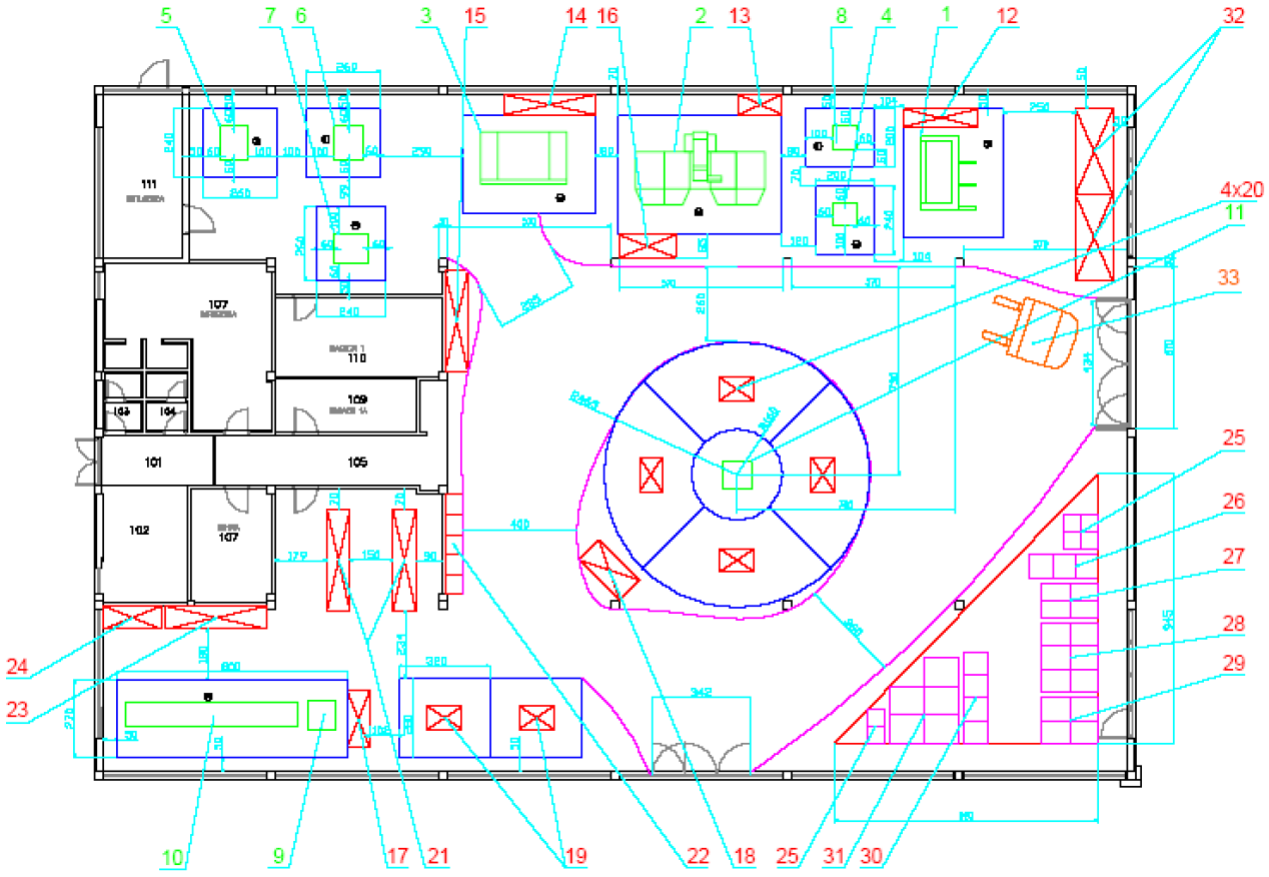


Слика 2.6.6.2: Алгоритам главног програма управљања

Основна идеја управљачког програма је да се робот креће у дискретним интервалима. На почетку се задају границе толеранције dmax, dmin, fimax и fimin. Толеранције dmax и dmin служе да се робот креће све док је у зони већој од dmax, а да стане када дође у зону мању од dmax. Углови fimax и fimin служе као углови толеранције правца ка циљној тачки.

2.7. Верификација резултата пројектног решења мобилног робота

Предходно конципиран целокупни систем мобилног робота тестиран је у симулираном радном окружењу фабрике у којој се врши производња и монтажа производа. Симулирано окружење направљено је према стварном производном погону чији је диспозициони план дат на слици 2.7.1 према [5].



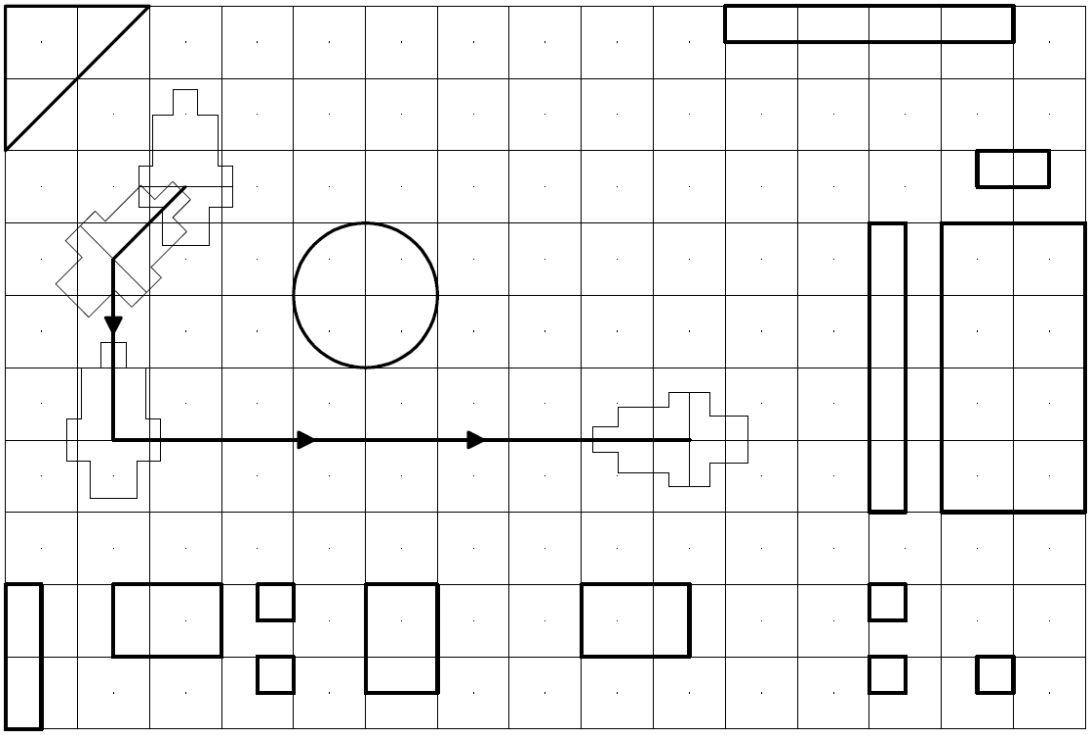
Слика 2.7.1: Диспозициони план симулираног окружења

У овом производном погону које представља радни простор мобилног робота, позиције које су означене црвеним словима представљају радна места предвиђена за раднике запослене у производном погону, као и одговарајућа магацинска места у доњем левом углу. Зеленим бројевима означене су позиције машина алатки око којих су љубичастом бојом уоквирени одговарајући радни простори. Такође, зеленом бојом нацртане су контуре и шематски прикази машина алатки. Све остало на цртежу пропада самом објекту производног погона, а то су спољни зидови, преградни зидови, стубови, врата и слично.

Симулирано окружење фабрике са моделима машина дато је на слици 2.7.2.

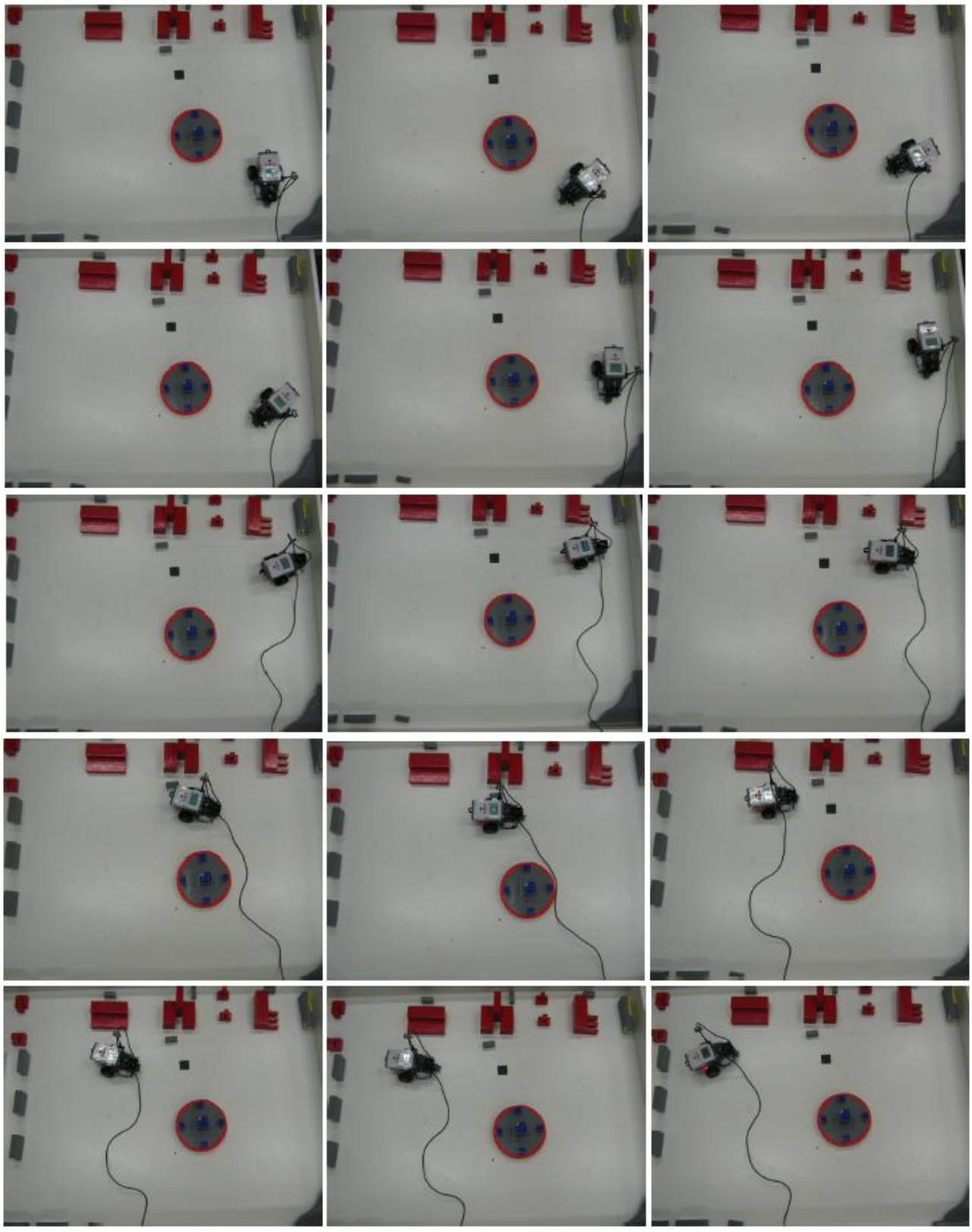
Слика 2.7.2: Симулирано радно окружење производног погона

Тестиране су функције претраге простора помоћу *A** алгоритма и добијање координата пиксела према жељеној путањи кретања. После тога је у рад пуштен програм управљања кретањем који је мобилним роботом управљао по жељеној путањи. Постављене су црне тачке по радном окружењу на основу којих је мобилни робот успешно кориговао координате позиције. Изведена путања мобилног робота дата је на слици 2.7.3.



Слика 2.7.3: Изведена путања мобилног робота

На слици 2.7.4 дати су прикази мобилног робота током кретања кроз окружење по задатој трајекторији.



Слика 2.7.4: Прикази мобилног робота током кретања

2.8. Дискусија и закључак

Целокупни систем мобилног робота који је конципиран у овом елаборату, представља савремени приступ развоју технолошких система са својим елементима. Акценат је стављен на сам појам интелигенције система што је и био циљ студентског пројекта. Са појмом интелигенције аутоматски се везује и појам аутономности што представља јединствену особину опште познатих интелигентних система од човека до вештачких система препознавања..

Пројектовани робот садржи поједине карактеристике вештачке интелигенције што се може објаснити коришћењем једне од парадигми вештачке интелигенције, а то су вештачке неуронске мреже. Такође, мобилни робот има особину „расуђивања“ између тога да ли је нека боја црна или бела, на основу чега се доносе одговарајуће одлуке о даљем раду мобилног робота.

Ове карактеристике вештачке интелигенције су на основном нивоу и подразумевају елементарне алгоритме који поред своје једноставности ипак захтевају велику посвећеност и пажњу у циљу оспособљавања система мобилног робота за активан рад у свом окружењу.

Поред наведених карактеристика, конципиран мобилни робот има и своје недостатке. Ти недостатци могу бити од круцијалног значаја за рад овог система. Првенствено, робот конструисан помоћу LEGO пакета показао се доста нетачаним и непрецизним у раду. То не представља проблем у генералном смислу јер се са бољим компонентама може конструисати мобилни робот врло добрих карактеристика. Разлог грешака LEGO мобилног робота зазнивају се на више чињеница:

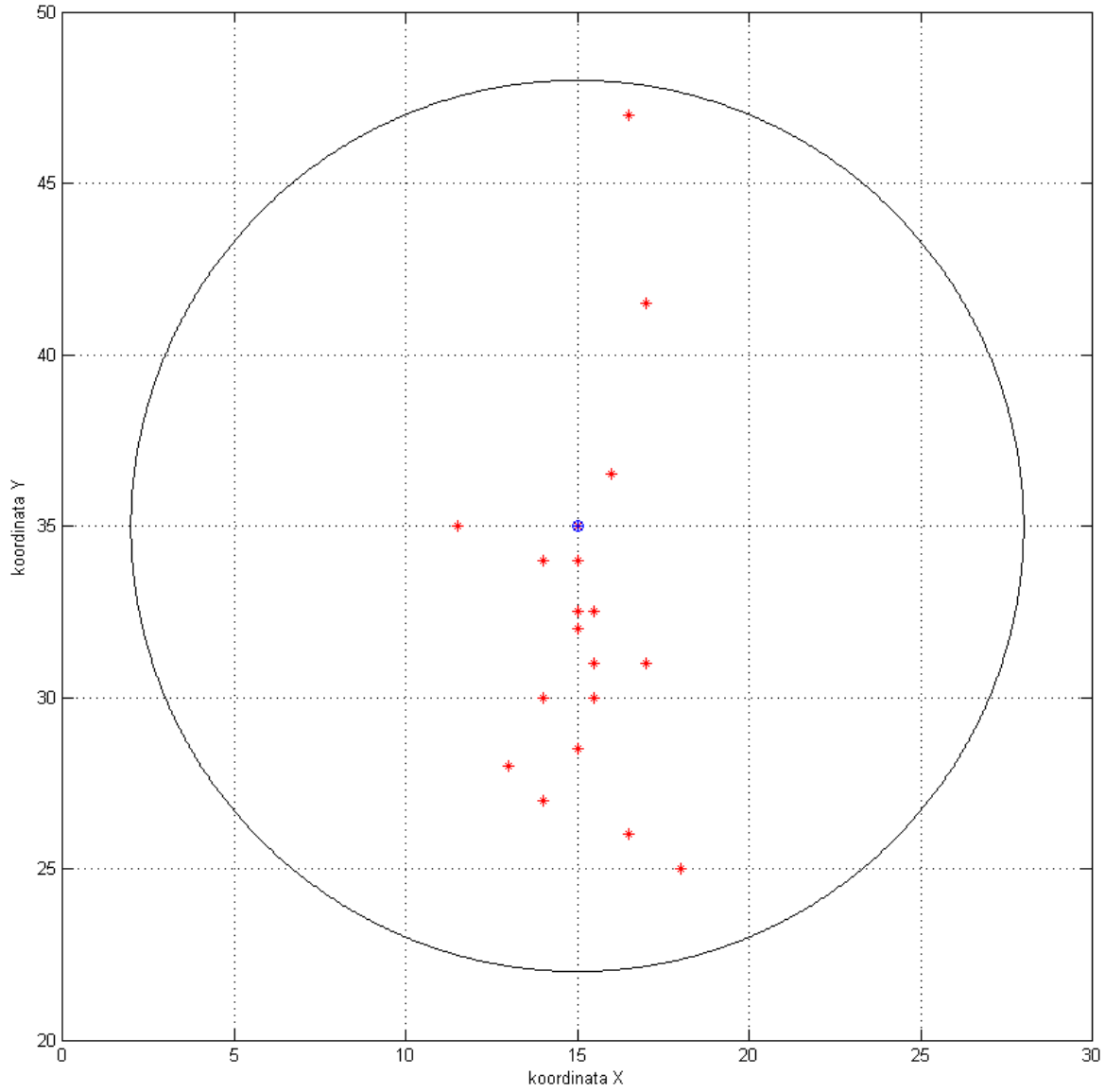
1. Управљачки алгоритам није пројектован да коригује акумулирану рачунску грешку
2. Увођењем вештачких неуронских мрежа као основ за окретање мобилног робота, уносе се и грешке које су последица нетачних мерења за обучавајући вектор вештачких неуронских мрежа, и због самих неуронских мрежа
3. Коришћење недовољно тачног мерног система мотора који је 12bit-ни инкрементални енкодер
4. Постојање шестостепеног зупчаног преносника у мотору и самим квалитетом израде целокупног мотора и његових делова.

У суштини, због саме цене LEGO пакета која је веома приступачна, не могу се ни очекивати боље перформансе система.

У циљу показивања тачности и прецизности, направљено је двадесет мерења при чему је робот из позиције (15,35) одлазио у (85,35) и враћао се у почетну позицију. Тиме је показано колики су тачност и поновљивост робота при постизању исте тачке. Параметри главног управљачког програма који су овде коришћени су:

$$d\max = 1$$
$$d\min = 1$$
$$fi\max = 20$$
$$fi\min = 10$$

Добијени резултати повратка у почетну позицију дати су на слици 2.8.1. Може се уочити да у овом случају робот више греши по X координати него по Y координати.



Слика 2.8.1: Остварене циљне позиције мобилног робота

Приликом тестирања јавио се још један проблем који ремети несметани рад мобилног робота у окружењу. Проблем представља однос величина робота и окружења. Мобилни робот је великих димензија у односу на лабораторијски модел технолошког окружења производног погона, а због своје нетачности и непрецизности често се дешава колизија са објектима у окружењу што ремети његов рад. Из тог разлога потребно је експериментално утврдити опсег координата робота у радном простору при којима не долази до сударања робота и објеката.

Путања која је коришћена у поглављу 2.7 коригована је у односу на добијену путању из алгоритма A*. Део путање који је коригован јесте део од тачке (15,35) до (95,35). Са том корекцијом робот несметано пролази између монтажног стола и машина.

3. Технолошко окружење производног предузећа

У овом делу пројектног задатка потребно је, на основу анализе тренутних технолошких поступака обраде делова и распореда машина и радних места, направити симулациони модел диспозиционог плана погона за производњу лименки у задатком задатом предузећу. Након извршене симулације, анализирати добијене резултате и предложити нови диспозициони план технолошког окружења.

Уређење производног простора може да се изврши на два начина [5]:

1. *Размештај производне опреме према врсти процеса*, подразмева да се машине и уређаји групишу, према функцији коју обављају, у специјалним одељењима.

2. *Размештај производне опремепрема редоследу технолошких операција*, подразумева да се машине и уређајипостављају у линијском распореду, где је редослед извођења операција условљен пројектованим технолошким поступком за сваки део који се израђује, а који директно утиче на квалитативне међузависности машина и опреме који се користе у том погону.

У предузећу задатом у тексту задатка је примењен принцип уређења производног простора базиран на другом начину. Током процеса пројектовања распореда машина и радних места коришћен је квалитативан критеријум међузависности, с обзиром на то да су у разматрање биле укључене све активности које су значајне за одвијање основног технолошког процеса [5].

Симулација је имитација рада реалног процеса или систематском времена и омогућава генерисање вештачке историје и посматрање те вештачке историје у циљу извођења закључака о радним карактеристикама система. Понашање сисстема које се мења током времена проучава се развијањем симулационог модела [8].

Разлози за примену симулације: [8]

- Симулација омогућује проучавање интеракција унутар комплексног система;
- Могу се симулирати и изучавати промене информационе структуре, организационе промене и промене окружења;
- Симулациони модел подстиче унапређење знања о систему;
- Проналажење важних улазних параметара кроз промену симулационих улаза;
- Експериментисање са новим пројектима и стратегијама пре иплементације;
- Симулирање различитих могућности машина ради одређивања потреба;
- Симулациони модели за обуку омогућавају учење без великих трошкова;
- План се може визуелизовати короз анимирани приказ;
- Савремени систем (фабрике, постројење, сервиси итд.) су толико комплексни да се интеракције унутар система могу обрађивати кроз саму симулацију.

3.1. Симулациони модел у Anylogic окружењу

Програмски пакет који је био коришћен за симулацију тока производње лименки у предузећу задатом задатком, је *AnyLogic*. *AnyLogic je* алат опште намене за моделирање и симулацију дискретних, континуалних и хибридних система. Његова област примене обухвата: контола система, саобраћај, динамика система, производња, ланац снабдевања, логистика, компјутерски системи, мреже, механика, војни, едукативни итд.

Симулација је прављена на основу података о припремним временима и временима обраде који су дефинисани технолошким поступком. Табела 3.1.1 приказује времена трајања технолошких поступака за изабрани део.

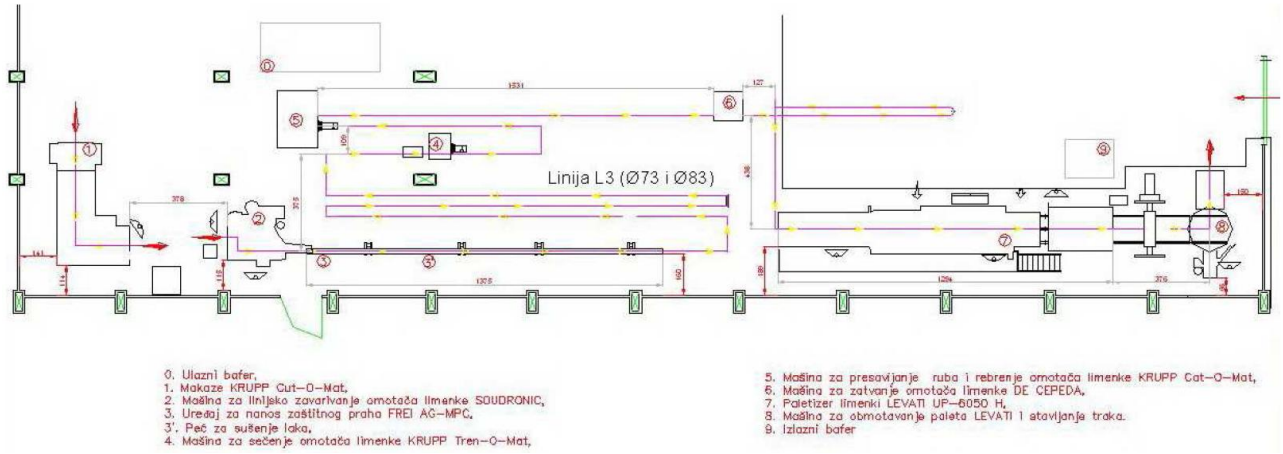
Табела 3.1.1: *Трајање технолошких поступака за изабреани део из производног асортимана*

Р.Б.	Технолошки поступак	Машина	Време [s]
1.	Транспорт палете са лимовима од улаза у халу до маказа и спушање	Виљушкар-бензинац	25
2.	Постављање палете на машину	Виљушкар-електрични	179
3.	Отпакивање палете	Ручна манипулација	25
4.	Подизање палете до вакуумских сисаљки	Аутоматски	25
5.	Сечење припремка 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 слоја	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.	Транспотр	Магнетни транспортер	21
15.	Монтирање дна	Аутомат	2.5
16.	Транспорт	Магнетни транспортер	17
17.	Палетизација	Палетизатор	64
18.	Звезивање и облепливање	Машина	269
19.	Одвожење палете са лименкама до врата погона	Виљушкар-бензинац	27

У табели 3.1.2 дат је списак и опис сваке машине понаособ у производној линији.

Табела 3.1.2: *Списак и опис машина у проижводној линији*	
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	Палетизер
M8	Машина за увезивање и улепљивање

Производни погон чији је технолошки процес описан у табели 3.1.1 и чије су машине дате у табели 3.1.2, дат је као диспозициони план на слици 3.1.1.

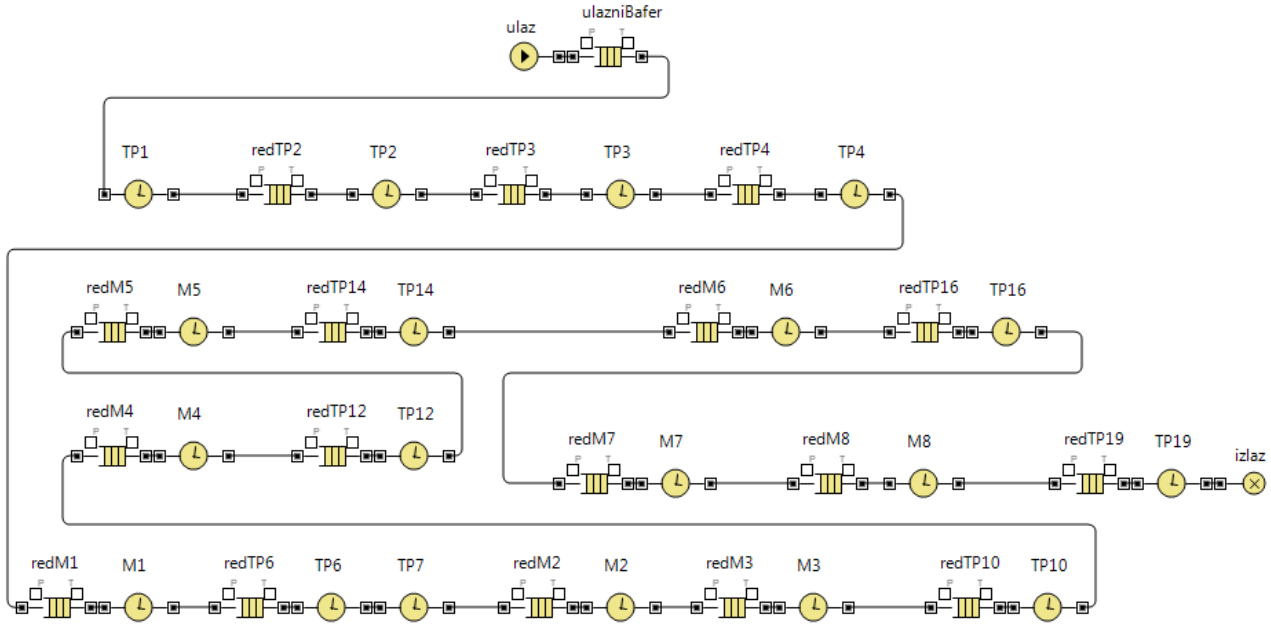


Слика 3.1.1: Диспозициони план прдузећа задатог задатком

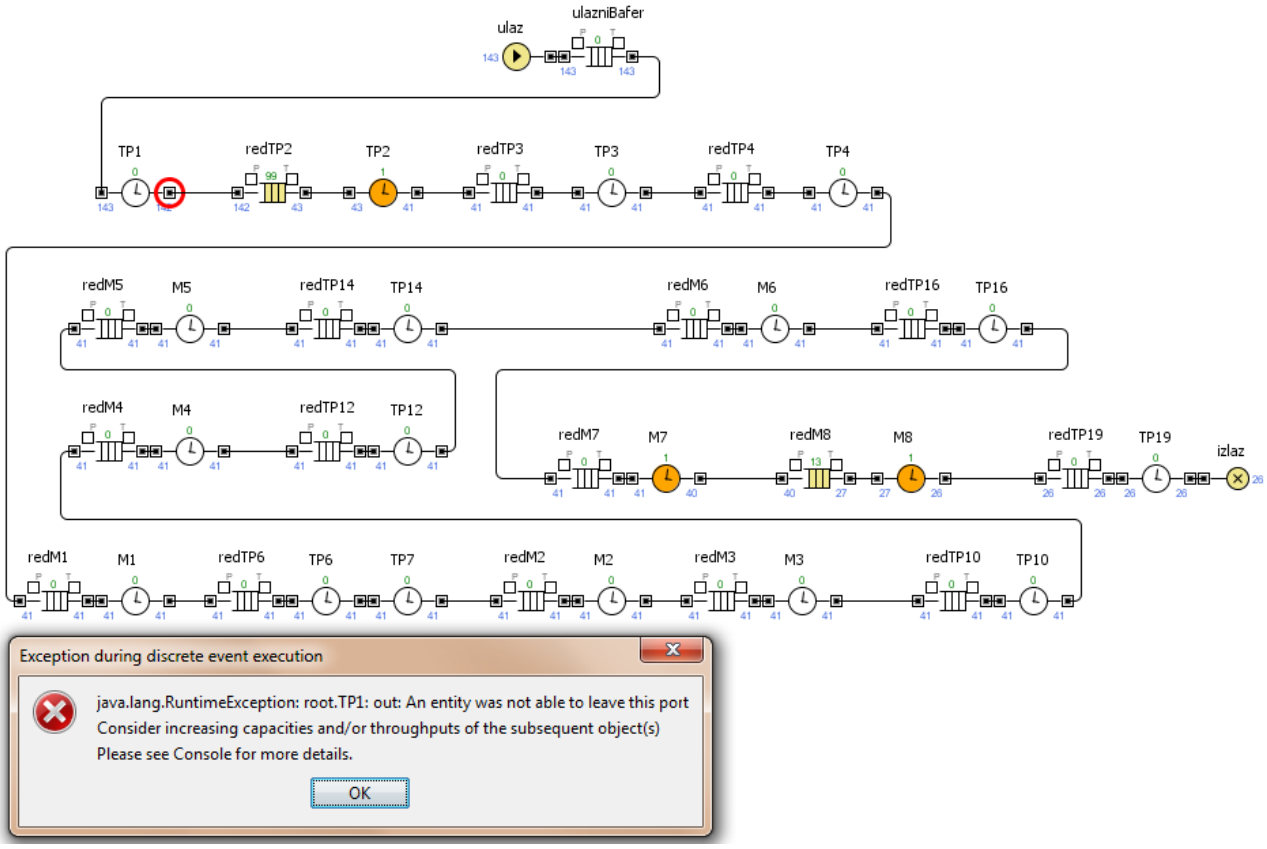
Помоћу података из табеле 3.1.1 извршена је симулација. Од ентитета у симулацији су коришћени:

- *Souce* - 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

На слици 3.1.1 је приказан симулациони модел производног тока. Моделиран је тако да приближно изгледа као диспозициони план, ради лакшег сналажења. Време трајања симулације је еквивалентно трајању једне смене од 8 сати. Да би симулација могла да функционише испред сваке машине и сваког поступка транспорта је постављен ред чекања. Узима се у обзир да је број припремака једнак броју готових делова јер није наглашено у поставци задатка колико се из једног листа лима може добити готових делова и колико се припремака на палети унесе у производни погон.



Слика 3.1.1: Симулациони модел производног тока



Слика 3.1.2: Симулациони модел на крају тока симулације

На слици 3.1.2 приказан је модел добијен на крају симулације. Види се да постоји грешка и након анализе утврђено је да се она десила у тренутку 7586 од почетка симулације. Разлог заустављања симулације је тај сто је дошло до загушења у реду чекања за технолошки поступак бр. 2. Такође се може приметити да се јавља ред чекања од 13 припремака на масини М8 тј. за технолошки поступак 18.

Разлози због којих долази до загушења код технолошког поступка 2 и до реда чекања код технолошког поступка 18 су времена трајања тих технолошких поступака.

На осталим деловаима линије за производњу делова нису се појавили проблеми. У производни процес је ушло 148 припремака од којих је израђено 26 готових делова.

3.2. Пројектовање диспозиционог плана технолошког окружења

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

Троугаона матрица успоставља зависност машина по критеријумима из табеле под а и са разлогом из табеле под б са слике 3.2.2. Оцењено је да је најважнији критеријум оцењивања ток материјала. Палетизатор, а поготово машина за улепљивање и везивање не морају бити у непосредној близини осталих машина са производне линије. Аутомат за заваривање и пећ имају велики утицај са становишта заштите животне средине и радног простора што се наводи под разлогом екологије.

Слика 3.2.1: Троугласта матрица пројектовања

ознака	степен зависности
а	апсолутно неопходно
е	веома важно
и	важно
о	потребно
у	неважно
х	непожељно
w	веома непожељно

а)

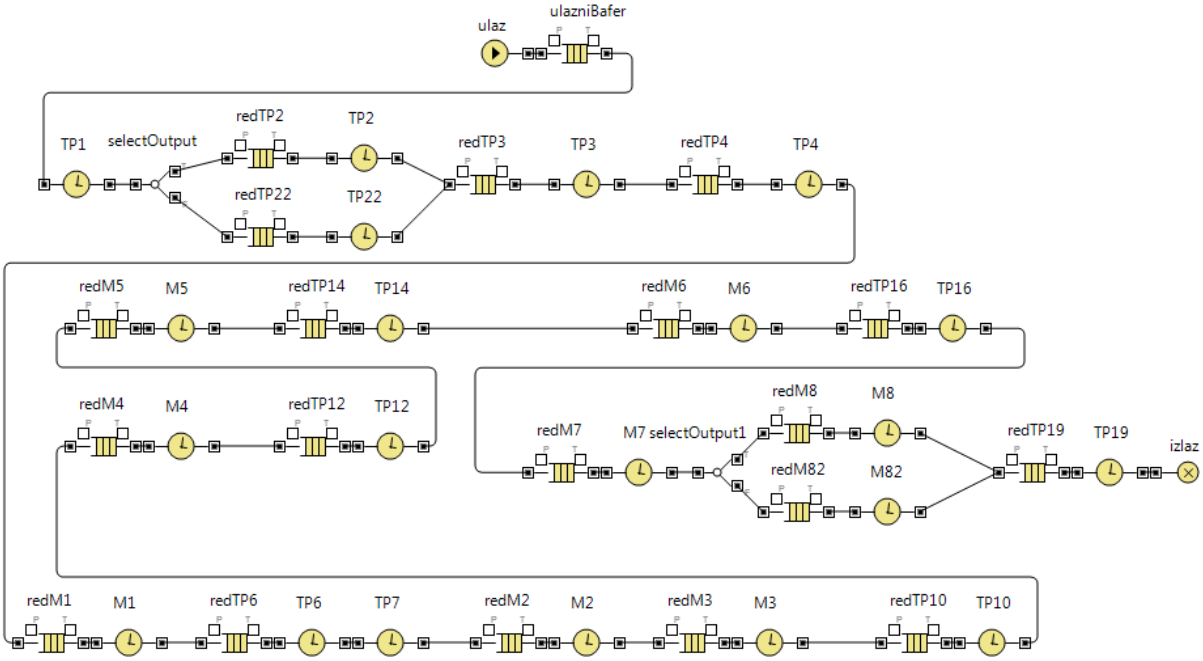
ознака	разлог
1	ток материјала
2	тип машине
3	одржавање обрадног система
4	еколошки

б)

Слика 3.2.2: Табеле критеријума и разлога веза између машина

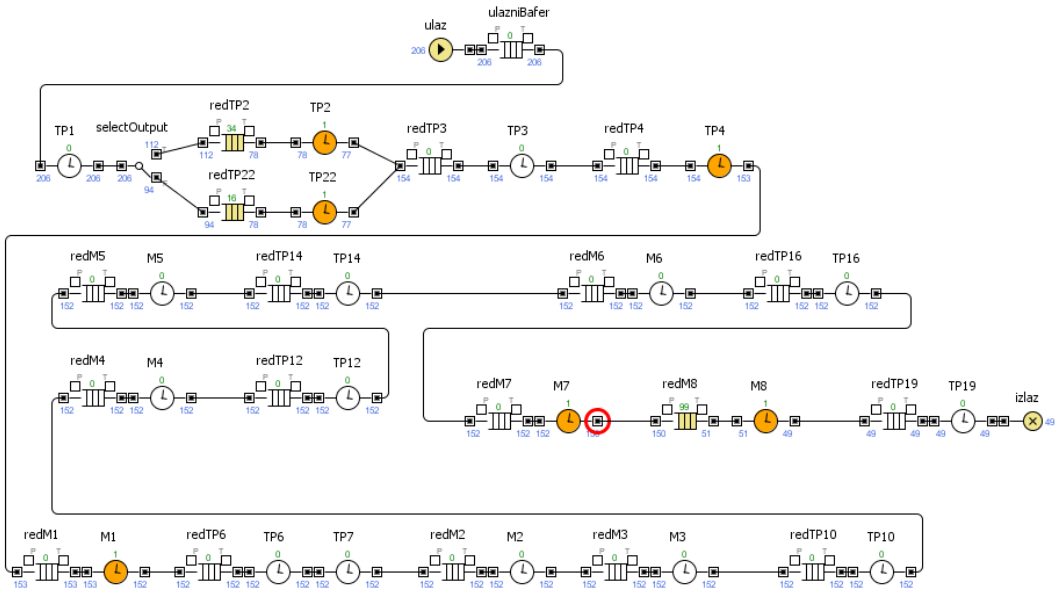
3.3. Ново предложено решење технолошког окружења

Загушење у производном процесу се може отклонити тако сто се у симулацију уведу додатане машине на местима где долази до застоја и чекања на ред. На слици 3.2.1 је дат симулациони модел оваквог решења. Симулиран је рад једне смене у трајању од 8 сати.



Слика3.2.1: Симулациони модел производног тока након измене

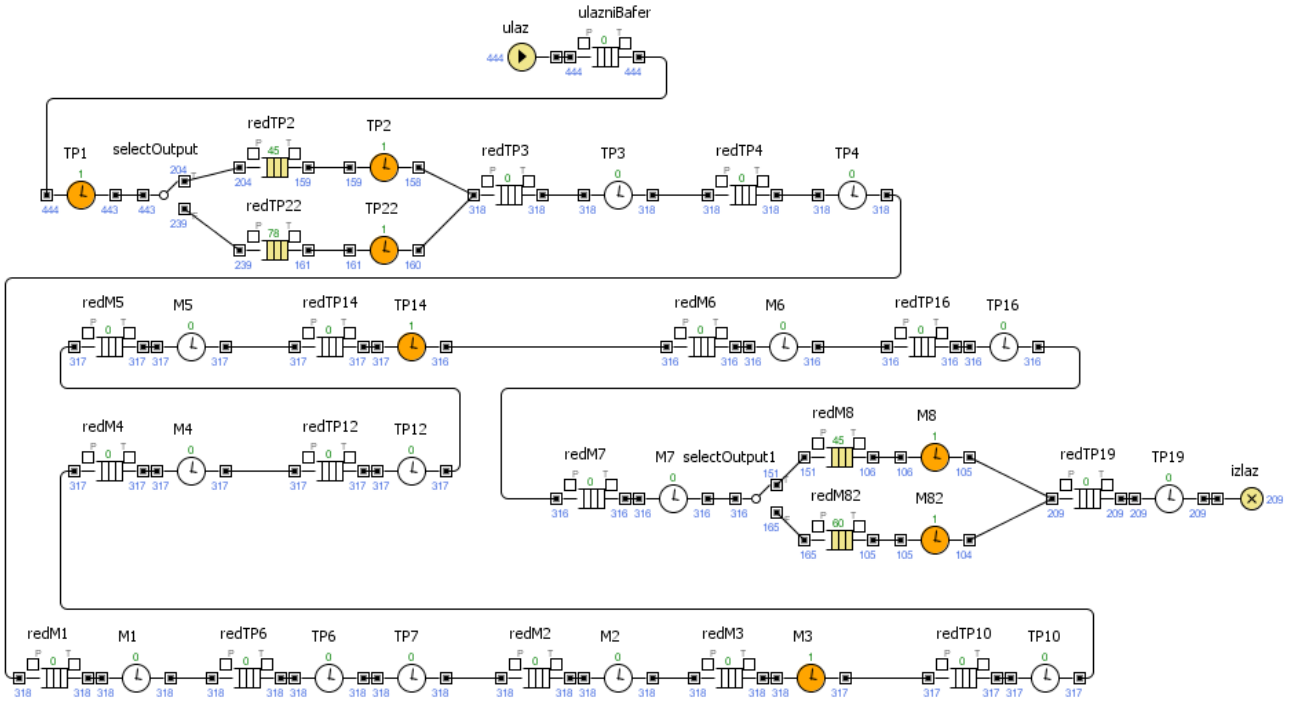
На слици 3.2.1 се може уочити измена две радне јединице, на месту TP2 и M8 тј. још једног електричног виљушкара и још једне машине за увезивање и улепљивање. Разлог додавања још једне машине за увезивање и улепљивање је тај што се након додавања електричног виљушкара јавља застој на месту M8 што се може видети на слици 3.2.2.



Слика 3.2.2: Симулациони модел на крају симулације након прве измене

Слика 3.2.3 приказје симулациони модел на крају симулације након унетих измена. Може се спримети да је загушење отклоњено и да нема застоја. Даљим анализирањем се може, на крају симулације, закључити следеће:

- Да је на месту TP1 један припремак још увек у транспорту
- Јавља се ред чекања од 45 и 78 припремака и по 1 припремак је у транспорту на местима TP2 и TP22 из истих разлога који су наведени у првој симулацији
- На месту M3 један припремак је још увек у обради
- На месту Месту TP14 такође је један део у транспорту
- На местима M8 и M82 се јавља ред на чекање од 45 и 60 делова као и по један део у обради тј. паковању
- Из складишта полуфабриката послато је 444 припремака, а произведено је 209 готових делова



Слика 3.63: Симулациони модел производног тока на крају симулације након коначних измена

3.4. Дискусија и закључак

Симулациони модел токова материјала предузећа „XY“ је направљен да би смо добили бољи увид у процес производње ради његовог анализирања и касније његовог унапређења. Резултати ове симулације дали су добар увид у тренутно стање производног процеса, као и могућности унапређења ради ефикаснијег искоришћења постојећих ресурса, самим тим и бољих производних резултата.

Анализирањем поменутог симулационог модела је показала да је тренутни распоред машина прилично добар, нема већих застоја и губитака времена осим на пар места, али они не представљају велики проблем у производњи. Уз мале измене у смислу увођења нових машина и промене унутрашњег транспорта побољшала би се ефикасност машина па самим тим и производност.

Треба напоменути да овај модел није веродостојан због мањка информација о производном току у симулираном предузећу. На пример, колико се припремака налази на палети приликом допремања из магацина у погон, колико се обрадака добије из једног припремка и колико се обрадака налази на палети приликом паковања и одношења из погона. Због тога није добијен реалан модел, па самим тим није оправдано применити предложени концепт.

Решавањем ових проблема добили бисмо реалну слику производног процеса трансформације материјала и резултати који су добијени би били боље анализирани и могли би се упоредити са реалним резултатима. Тиме би, наравно, добили и боље решење које би било могуће применити и као реално.

4. Литература

[1] 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.

[2] Миљковић, З. Алексендрић, Д. *Вештачке неуронске мреже – збирка задтака са изводима из теорије*, Универзитет у Београду -Машински факултет, Београд, 2009.

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

[4] Петровић, П. *Мехатронски системи - белешке са предавања*, Универзитет у Београду - Машински факултет, Београд, 2010

[5] Миљковић, З. *Интелигентни технолошки системи – изводи са предавања*, Универзитет у Београду -Машински факултет, Београд, 2010

[6] Jones, F., *Mobile robots – inspiration to implementation*, IEEE Publishing Service, New York, 2002.

[7] Јуришић, Л., Олетић, Д., Пекарчић, Е., *Калманов филтер Семинар из предмета случајни процеси у суставима*, Загреб, 2009.

[8] Бабић, Б., Предавања на предмету Компјутерска симулација и вештачка интелигенција Универзитет у Београду – Машински факулте, Београд, година?

[9] LEGO DIGITAL DESIGNER, http://ldd.lego.com/download/, датум полседњег приступа: 05.02.2011.

[10] RWTH Mindstorms NXT Toolbox, http://www.mindstorms.rwth-aachen.de/ , датум последњег приступа: 10.02.2011.

[11] MATLAB R2009a, Product Help, http://www.mathworks.com/help/, датум последњег приступа: 10.02.2011.

[12] Robotic Industries Association, http://www.robotics.org/product-catalog-detail.cfm/Robotic-Industries-Association/Robot-Terms-and-Definitions/productid/2953, датум последњег приступа: 15.02.2011.