

МОНГОЛ УЛСЫН ШИНЖЛЭХ УХААН  
ТЕХНОЛОГИЙН ИХ СУРГУУЛЬ  
МЭДЭЭЛЭЛ, ХОЛБООНЫ ТЕХНОЛОГИЙН СУРГУУЛЬ



Батнасангийн Ганзориг

**Симуляцийн орчинд бататган  
суралцах аргыг робот системд  
хэрэгжүүлэх судалгаа**

БАКАЛАВРЫН ТӨГСӨЛТИЙН АЖИЛ

Улаанбаатар хот

2020 он

МОНГОЛ УЛСЫН ШИНЖЛЭХ УХААН  
ТЕХНОЛОГИЙН ИХ СУРГУУЛЬ  
МЭДЭЭЛЭЛ, ХОЛБООНЫ ТЕХНОЛОГИЙН СУРГУУЛЬ

Электроникийн салбар

Симуляцийн орчинд бататган  
сурх аргыг хэрэгжүүлэх судалгаа

Хөтөлбөрийн индекс: D071401  
Хөтөлбөр: Электроник  
Мэргэжил: Автоматжуулалтын систем

Үүдирдагч: Док (Ph.D) Б.Дорж  
Зөвлөгч: Док (Ph.D) Дэд проф Д.Эрдэнэчимэг, Док (Ph.D)  
Ц.Тэнгис  
Гүйцэтгэгч: Б.Ганзориг

Улаанбаатар хот  
2020 он 1 сар

Батлав. Электроникийн салбарын эрхлэгч:

..... / Док (Ph.D) А.Одгэрэл/

Удирдагч:

..... / Док (Ph.D) Б.Дорж /

## ТӨГСӨЛТИЙН АЖЛЫН ТӨЛӨВЛӨГӨӨ

СЭДЭВ: " Симуляцийн орчинд бататган суралцсан аргыг  
хэрэгжүүлэх судалгаа"

№	Хийгдэх ажлын жагсаалт	Дуусах хугацаа	Эзлэх хувь
1	Онолын судалгаа хийх	2019-10-15	20%
2	Симуляцийн орчин бэлдэх, онолын хэсэг бичих	2019-10-30	10%
3	Симуляцийн хэсэг	2019-11-15	20%
4	Бодит робот системийг хэрэгжүүлэх хэсэг	2019-12-15	30%
5	Дүгнэлт	2019-12-25	10%

Төлөвлөгөөг боловсруулсан оюутан: ..... /Б.Ганзориг/

## ТӨГСӨЛТИЙН АЖЛЫН ЯВЦ

№	Хийж гүйцэтгэсэн ажил	Биелсэн хугацаа	Удирдагчийн гарын үсэг
1	Үдиртгал	2019-10-05	
2	Онолын хэсэг	2019-10-30	
3	Судалгааны хэсэг	2019-11-15	
4	Төслийн хэсэг	2019-12-20	
5	Дүгнэлт	2019-12-25	

## Ажлын товч дүгнэлт

.....  
.....  
.....  
.....  
.....

Удирдагч: ..... /Док (PhD) Б.Дорж /

ЗӨВШӨӨРӨЛ

Оюутан Б.Ганзориг –н бичсэн төгсөлтийн ажлыг УШК-д хамгаалуулахаар тодорхойлов.

Салбарын эрхлэгч: ..... /Док (Ph.D), А.Одгэрэл/

МОНГОЛ УЛСЫН ШИНЖЛЭХ УХААН  
ТЕХНОЛОГИЙН ИХ СУРГУУЛЬ  
МЭДЭЭЛЭЛ, ХОЛБООНЫ ТЕХНОЛОГИЙН СУРГУУЛЬ

**ШҮҮМЖИЙН ХУУДАС**

Электроникийн салбар-н төгсөх курсийн оюутан Б.Ганзориг-н "Симуляцийн орчинд бататган сурх аргыг хэрэгжүүлэх судалгаа" сэдэвт төгсөлтийн ажлын шуумж.

1. Төслөөр дэвшүүлсэн асуудал, үүнтэй холбоотой онолын материал уншиж судалсан байдал. Энэ талаар хүмүүсийн хийсэн судалгаа, түүний үр дүнг уншиж тусгасан эсэх.

.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....

2. Төслийн ерөнхий агуулга. Шийдсэн зүйлүүд, хүрсэн үр дүн. Өөрийн санааг гарган, харьцуулалт хийн, дүгнэж байгаа чадвар.

.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....

3. Эмх цэгцтэй, стандарт хангасан өөрөөр хэлбэл диплом бичих шаардлагуудыг биелүүлсэн эсэх./Төсөлд анзарагдсан зөв бичгийн, өгүүлбэр зүйн, найруулгын алдаанууд болон хувилан олшруулсан материал олон тоогоор оруулсан/

Шүүмж бичсэн: ..... /Магистр (MSc) Б.Луубаатар/

Ажлын газар: . . . . .

## Хаяг (Утас) . . . . .

# Зохиогчийн эрхийн хамгаалал

Миний бие Б.Ганзориг, "Симуляцийн орчинд бататган сурхадарыг хэрэгжүүлэх судалгаа" сэдэвт энэ ажил нь минийх бөгөөд дараахыг нотолж байна.  
Үүнд:

- Горилогч энэ ажлыг тус сургуулиас боловсролын зэрэг авахаар бүхэлд нь буюу голлон хийсэн болно.
- Энэ ажлын аль нэг хэсгийг тус сургуульд эсвэл өөр байгууллагад боловсролын зэрэг, мэргэшил авахаар өмнө нь илгээсэн бол түүнийгээ тодорхой заасан болно.
- Бусад хүмүүсийн хэвлүүлсэн ажлаас зөвлөгөө авсан бол түүнийгээ үндэслэсэн болно.
- Бусад хүмүүсийн ажлаас ишлэл хийсэн бол эх үүсвэрийг нь заасан болно.
- Миний ажилд тусалсан голлох бүх эх үүсвэрт талархаж байна.
- Ажлыг бусадтай хамтарсан бол алийг нь бусад хүмүүс хийсэн болохыг тодорхой заасан болно.

Гарын үсэг: \_\_\_\_\_

Огноо: \_\_\_\_\_

“Энгийн нэгэн агуу байхыг сонгох боломжтой.”

Элон Маск

МОНГОЛ УЛСЫН ШИНЖЛЭХ УХААН  
ТЕХНОЛОГИЙН ИХ СУРГУУЛЬ  
МЭДЭЭЛЭЛ, ХОЛБООНЫ ТЕХНОЛОГИЙН СУРГУУЛЬ

## Хураангуй

Симуляцийн орчинд бататган суралцах аргыг робот  
системд хэрэгжүүлэх нь

Б.Ганзориг  
gbatnasan5@gmail.com

*Түүхүүр үсг: Reinforcement Learning , value based method, Q-Learning, Double Deep Q network, Машин сургалт, Робот үйлдлийн систем, газебо, саад тойрдог робот*

Энэхүү төгсөлтийн төслийн ажлын хүрээнд машин сургалтын нэг арга болох бататган сурх (Reinforcement Learning) аргыг робот системд хэрхэн хэрэгжүүлэхийг судалж, симуляцийн орчинд саад тойрох робот дээр туршилт хийсэн ба бодит робот дээр хэрэгжүүлсэн. Симуляц хийхдээ робот үйлдлийн систем [1] ашигласан бөгөөд роботийн 3D загварыг urdf форматаар гарган газебо симулятор дээр туршилт явуулсан. Туршилтыг үнэлгээнд суурилсан (Value based) Q сургалт болон давхар гүн Q нейрон сүлжээний аргуудыг ашиглан амжилттай гүйцэтгэсэн. Симуляцийн орчинд сургасан гүн нейрон сүлжээний загварыг бодит робот дээр амжилттай хэрэгжүүлсэн.

# ТӨВЧИЛСОН ҮГС

<b>ML</b>	Machine Learning
<b>RL</b>	Reinforcement Learning
<b>DL</b>	Deep Learning
<b>ROS</b>	Robot Operating System
<b>API</b>	Application Programming Interface
<b>CPU</b>	Central Processing Unit
<b>GPU</b>	Graphics Processing Unit
<b>ATX</b>	Аналог Тоон Хувиргуур
<b>ИӨМ</b>	Импульсийн Өргөний Модуляц
<b>UART</b>	Universal Asynchronous Receiver Transmitter

*Өөрийн гэр бүлд зориулав*

# Гарчиг

Зохиогчийн эрхийн хамгаалал	i
Хураангуй	iii
Товчилсон үгс	iv
<b>1 Онолын хэсэг</b>	<b>1</b>
1.1 Машин сургалт (Machine Learning) . . . . .	1
1.1.1 Машин сургалтын төрлүүд . . . . .	1
1.2 Бататган сургалт (Reinforcement Learning) . . . . .	2
1.2.1 Марковийн шийдвэрийн процесс . . . . .	2
Марковийн чанар . . . . .	3
1.2.2 Бодлого(Policy) . . . . .	3
1.2.3 Үнэлгээ . . . . .	4
Төлөв-үйлдлийн үнэлгээ буюу Q-утга . . . . .	4
1.2.4 Q-Сургалт . . . . .	5
Epsilon greedy . . . . .	6
1.3 Гүн Q-сүлжээ (Deep Q-Network) . . . . .	6
1.3.1 Гүн давших чиглэлт нейрон сүлжээ (Deep feedforward networks) . . . . .	6
1.3.2 Шугаман бус идэвхжүүлэгч функц . . . . .	7
1.3.3 Experience replay . . . . .	7
1.3.4 Double Q-Learning . . . . .	8
1.4 Робот үйлдлийн систем . . . . .	8
1.4.1 ROS дэмждэг роботууд болон мэдрүүрүүд . . . . .	9
1.4.2 ROS filesystem level . . . . .	9
1.4.3 ROS тооцооллын графын түвшин . . . . .	10
1.4.4 ROS Хэрэглүүрүүд . . . . .	12
Rviz(ROS Visualizer) . . . . .	12
rqt_plot . . . . .	12
rqt_graph . . . . .	12
1.4.5 ROS simulator . . . . .	12
<b>2 Төслийн хэсэг</b>	<b>14</b>
2.1 Симуляц дахь туршилт . . . . .	14
2.1.1 Робот үйлдлийн систем дээрх роботийн симуляцийн тохиргоо . . . . .	14
2.1.2 Q-Learning . . . . .	17
2.1.3 DQN . . . . .	18
2.2 Бодит туршилт . . . . .	21

2.2.1	Техник хангамж . . . . .	21
	Зай хэмжих хэт улаан түяаны мэдрүүр . . . . .	21
	Тооцоолол хийх компьютер . . . . .	22
2.2.2	Програм хангамж . . . . .	26
2.3	Дүгнэлт . . . . .	28
<b>A Хавсралтын нэр</b>		<b>29</b>
<b>Ном зүй</b>		<b>40</b>

# Зургийн жагсаалт

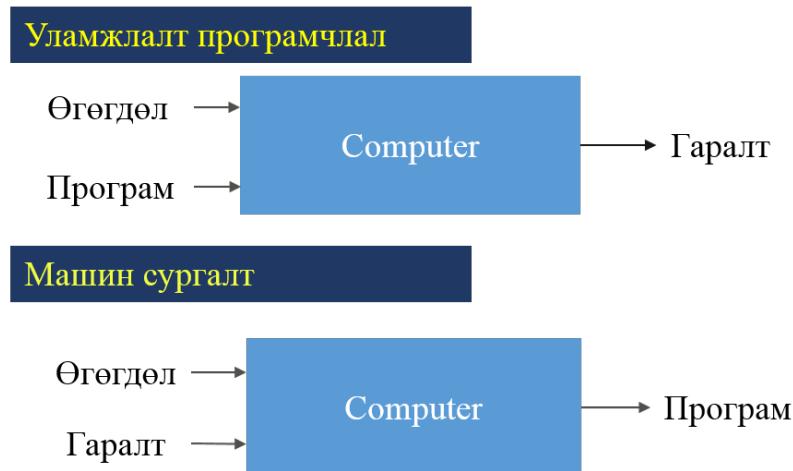
1.1	Уламжлалт програмчлал болон машин сургалт . . . . .	1
1.2	RL -ийн диаграм . . . . .	2
1.3	Бататган суралцах аргууд . . . . .	4
1.4	ROS ашигладаг өргөн хэрэглэгдэг роботууд . . . . .	9
1.5	ROS дэмждэг роботийн мэдрүүрүүд . . . . .	9
1.6	ROS файлсистем . . . . .	10
1.7	ROS тооцооллын граф . . . . .	10
1.8	ROS өгөгдөл солилцох процесс . . . . .	12
1.9	Газебо симулятор . . . . .	13
2.1	Нокиу лазерийн газебо симуляторт холбох холболт . . . . .	15
2.2	Роботийг дифференциал удирдлагын драйвераар удирдах симуляторийн холболт . . . . .	16
2.3	Симуляцийн орчин дахь ROS тооцооллын граф . . . . .	16
2.4	Роботийн симуляцийн орчин . . . . .	17
2.5	Дундаж шагнал . . . . .	18
2.6	Сургасны дараах роботийн явсан зам . . . . .	18
2.7	Q network архитектур . . . . .	19
2.8	Дундаж шагнал . . . . .	19
2.9	Double-DQN сургах алгоритм . . . . .	20
2.10	Бодит туршилтын робот . . . . .	21
2.11	Бодит туршилтын робот . . . . .	22
2.12	SHARP GP2Y0A21YK0F зайд хэмжих мэдрүүр . . . . .	22
2.13	Мэдрүүрээс авсан өгөгдөл болон шүүлтүүрээр шүүсэн утга . . . . .	23
2.14	Мэдрүүрээс объект хүртэлх зайд болон ATX ын утгын хамаарал . . . . .	23
2.15	Зарчмын схем . . . . .	24
2.16	NVIDIA Jetson TK1 . . . . .	24
2.17	Бодит роботийн бүтцийн схем . . . . .	26
2.18	Бодит роботийн тооцооллын граф . . . . .	27

## БҮЛЭГ 1

Онолын хэсэг

## 1.1 Машин сургалт (Machine Learning)

Машин сургалтын арга нь өгөгдөл дээрээс суралцан сайжрах боломжтой алгоритм юм. Компьютерийн програм суралцах гэдэг нь *Mitchell(1997)* тодорхойлсноор "Даалгавар  $T$  д харгалзах туршлага  $E$ , гүйцэтгэлийн хэмжүүр  $P$  ийн хувьд хэрэв  $P$  ээр хэмжигдэх даалгавар  $T$  ийн гүйцэтгэл туршлага  $E$  ээр нэмэгдэх юм" Зураг 1.1 д уламжлалт програмчлал болон машин сургалтын



ЗУРАГ 1.1: Уламжлалт програмчлал болон машин сургалт

алгоритмуудын ялгааг харуулав.

### 1.1.1 Машин сургалтын төрлүүд

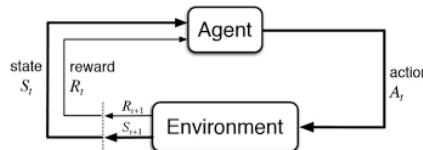
- **Supervised Learning:** Өгөгдлийн хэлбэр нь оролт, гаралт өгөгдсөн бөгөөд оролт болон гаралтын хоорондын хамаарлыг сурх ёстой.
- **Unsupervised Learning:** Өгөгдөл нь зөвхөн оролт байна. Оролтын өгөгдлийн нуугдмал бүтцийг загварчлах ёстой.
- **Reinforcement Learning:** Тухайн оролтын нөхцөлд хамгийн оновчтой шийдвэрийг гаргаж сурх ёстой. Шагналыг хамгийн их байлгахын тулд оновчтой шийдвэр гаргаж сурна.

Дээрх 3 машин сургалтын аргуудаас бид бататган сургалт буюу Reinforcement Learning арга нь робот системийн удирдлагад шууд хэрэгжүүлэх боломжтой бөгөөд цаашид Бататган суралцах аргын талаар дурьдах болно.

## 1.2 Бататган сургалт (Reinforcement Learning)

Бататган сургалтын арга нь ямар нэг програмийн агент оновчтой шийдвэр буюу үйлдэл хийж сурх арга юм. Агент нь тухайн нөхцөлд оновчтой шийдвэр гаргахын тулд хугацааны эгшин бүрд гэдрэг холбооны дохио болох шагнал ирнэ. Уг скаляр шагналын дохиог хамгийн их байлгахын тулд одоогийн нөхцөл байдалд ямар үйлдэл хийхийг сурна. Бусад машин сургалтын хэлбэрүүдээс ялгаатай нь суралцаж буй агентад ямар үйлдэл хийхийг зааж өгөхгүй ба ямар үйлдэл нь их шагнал авахыг олж мэдэх ёстой. Шагналын дохио нь бататган суралцах бодлогын зорилгыг тодорхойлж өгнө. Хугацааны алхам бүрт үйлдэл хийж буй орчин бататган суралцах агентад шагнал болох скаляр утгыг илгээнэ. Шагнал нь ямар үйлдэл оновчтой, ямар үйлдэл тааруу болохыг тодорхойлж өгнө. Биологийн системийн хувьд шагналыг бид тааламжтай байдал эсвэл өвдөлттэй ижил байдлаар ойлгож болно. Агентийн хувьд нэг алхмын дараах шагналыг их байлгах нь учир дутагдалтай. Жишээ нь шатар тоглох програмийн хувьд нэг алхмын дараах шагналыг их байлгавал занганд орж болно. Тэгэхээр агентийн гол объектив нь нэг алхмын дараах биш урт хугацаан дахь нийт шагналыг хамгийн их байлгах болно. Агентийн авах шагнал нь агентийн одоогийн хийх үйлдэл болон орчны төлвөөс хамаарна. Бататган суралцах аргыг математик загвараар илэрхийлэх шаардлагатай. Энэ нь Марковийн шийдвэрийн процесс байна. [2].

### 1.2.1 Марковийн шийдвэрийн процесс



ЗУРАГ 1.2: RL -ийн диаграм

Марковийн шийдвэрийн процесс нь дараах зүйлсүүдээр тодорхойлогоно.

- Төлвүүдийн олонлог  $S$
- Эхлэх төлөв  $s_0$
- Үйлдлүүдийн олонлог  $A$
- Шилжилт  $\Pr(s' | s, a)$
- Шагнал  $R(s, a, s')$

### Марковийн чанар

"Одоогийн нөхцөл өгөгдсөн үед ирээдүй өнгөрснөөс үл хамаарна" Хэрэв  $\forall s' \in S$  болон бүх шагнал  $r \in R_s$  төлөв нь дараах нөхцлийг хангаж байвал Марковийн нөхцөлийг хангана:

$$\begin{aligned} \Pr(S_{t+1} = s_{t+1} \mid S_t = s_t, a_t, S_{t-1} = s_{t-1}, A_{t-1} = a_{t-1}, \dots, S_0 = s_0, A_0 = a_0) &= \\ &= \Pr(S_{t+1} = s' \mid S_t = s_t, a_t) \end{aligned} \quad (1.1)$$

Энд  $S_1, \dots, S_{t-1}$  бүх боломжит өмнөх төлвүүд Марковийн чанарын гол санаа нь одоогийн төлөв орчны бүх мэдээллийг агуулсан буюу одоогийн нөхцөлөөс үйлдэл хийхэд дараагийн төлөв рүү шилжих шилжилт нь зөвхөн одоогийн нөхцөлөөс хамаарна.

$$G_t = R_{t+1} + R_{t+2} + \dots \quad (1.2)$$

Тухайн даалгавар нь төгсгөлгүй буюу эцсийн төлөв байхгүй бол хязгааргүй болох тул бууруулах утга  $\gamma$  (discount\_factor)-аар ирээдүйн шагналуудыг үржүүлж өгнө.

$$G_t = R_{t+1} + \gamma \cdot R_{t+2} + \gamma^2 \cdot R_{t+3} \dots \quad (1.3)$$

Энд  $0 \leq \gamma \leq 1$  байна.

#### 1.2.2 Бодлого(Policy)

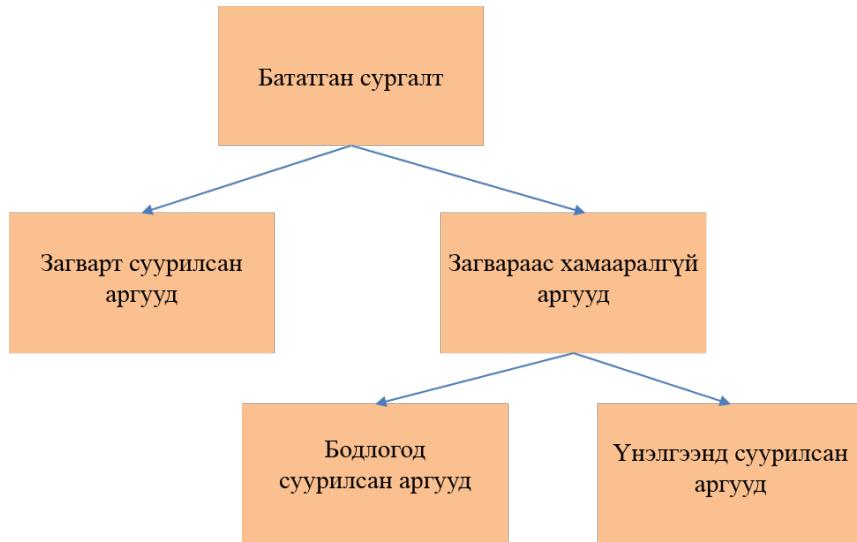
Бодлого нь төлөв бүрийн ( $s \in S$ ) хувьд үйлдлүүдийн ( $a \in A$ ) магадлалын тархалтыг тодорхойлдог функц. Хэрэв агент  $t$  хугацааны агшинд  $\pi$  бодлогыг дагадаг бол  $\pi(a|s)$  нь тухайн  $s$  төлөвд үйлдлийг хийх магадлал болно. Бататган сургалтад агентийн туршлага нь бодлогын өөрчлөлтийг тодорхойлж өгнө. Математик тэмдэглэгээгээр бодлого нь дараах байдлаар тодорхойлждоно:

$$\pi(a|s) = \Pr[A_t = a, S_t = s] \quad (1.4)$$

Бататган суралцах аргын объектив нь нийт шагналыг их байлгах бодлогыг олох явдал болно.

$$\arg \max_{\pi} \mathbb{E}_{\pi} \left[ \sum_{t=1}^T r_t \right] \quad (1.5)$$

Бататган суралцах аргын объектив нь нийт шагналыг их байлгах бодлогыг олох явдал болно.



ЗУРАГ 1.3: Бататган суралцах аргууд

- Загварт суурилсан аргууд: Системийн динамикийг буюу  $\Pr(S_{t+1} = s' | S_t = s_t, a_t)$  болон  $R(s, a, s')$  утгуудыг олсны дараа төлөвлөлт хийх аргууд
- Үнэлгээнд суурилсан аргууд: Төлөв бүрийг тоон үнэлгээгээр үнэлэх бөгөөд тухайн үнэлгээний загвараасаа бодлогыг тодорхойлно.
- Бодлогод суурилсан аргууд: Төлвийн утгаас тухайн бодлогыг дагахад авах шагналын хэмжээ их байх бодлогыг загварчилна.

### 1.2.3 Үнэлгээ

Төлвийн үнэлгээ:

$$V^\pi(s) = \mathbb{E}[G_t | S_t = s, \pi] \quad (1.6)$$

$V$  утга тухайн төлөв  $s$  ээс  $\pi$  бодлогыг даган явахад авах хуримтлагдсан шагналын дундаж утга болно. Беллман тэгшитгэл:

$$\begin{aligned} V^\pi(s) &= \mathbb{E}[R_{t+1} + \gamma \cdot G_{t+1} | S_t = s, \pi] = \\ &= \mathbb{E}[R_{t+1} + \gamma V^\pi(S_{t+1}) | S_t = s, A_t \sim \pi(S_t)] = \\ &= \sum_a \pi(a|s) \sum_r \sum_{s'} \Pr(r, s'|s, a)(r + \gamma \cdot V^\pi(s')) \end{aligned} \quad (1.7)$$

#### Төлөв-үйлдлийн үнэлгээ буюу Q-утга

$V^\pi(s)$  утга нь тухайн  $s$  төлвөөс тухайн  $\pi$  бодлогыг даган явахад авах шагналуудын нийлбэрийн дундаж утгыг илэрхийлэх бол  $Q^\pi(s, a)$  (*Quality*) нь  $s$

төлвөөс  $a$  үйлдлийг хийсний дараа  $\pi$  бодлогыг даган явахад авах шагналуудын нийлбэрийн дундаж утгыг илэрхийлнэ.

$$V^*(s) = \max_a Q^*(s, a) \quad (1.8)$$

Төлөв, үйлдэлийн утга:

$$\begin{aligned} Q^\pi(s, a) &= \mathbb{E}[R_{t+1} + \gamma \cdot V^\pi(S_{t+1})] = \\ &= \mathbb{E}[R_{t+1} + \gamma \cdot Q^\pi(S_{t+1}, A_{t+1}) | S_t = s, A_t = a] = \\ &= \sum_r \sum_{s'} \Pr(r, s' | s, a) (r + \gamma \cdot \sum_{a'} \pi(a' | s') Q^\pi(s', a')) \end{aligned} \quad (1.9)$$

Төлвийн утга болон төлөв-үйлдлийн утгын хоорондын хамаарал нь дараах хэлбэртэй байна.

$$V^\pi(s) = \sum_a \pi(a | s) Q^\pi(s, a) = \mathbb{E}[Q^\pi(S_t, A_t) | S_t = s, \pi], \forall s \quad (1.10)$$

**Оновчтой төлвийн үнэлгээ болон төлөв-үйлдлийн үнэлгээ:**

$$Q^*(s, a) = \max_\pi Q^\pi(s, a) \quad (1.11)$$

$$Q^*(s, a) = \sum_{s'} \Pr(s, a, s') [R(s, a, s') + \gamma * V^*(s')] \quad (1.12)$$

#### 1.2.4 Q-Сургалт

Үнэлгээ давтах алгоритмууд (Value iteration)  $i \rightarrow \inf$  үед  $Q_i$  нь оновчтой төлөв-үйлдлийн функц руу нийлнэ [3].

$$Q^\pi(s, a) = R + \gamma \cdot \max_{a'} Q(s', a') \quad (1.13)$$

$$Q(s, a) \leftarrow (1 - \alpha) \cdot Q(s, a) + \alpha \cdot (R + \gamma \cdot \max_{a'} Q(s', a')) \quad (1.14)$$

Энд  $\alpha$  нь сурх хурд (learning rate). Ямар нэгэн төлвөөс үйлдэл хийх болгонд бид дараагийн төлөв болон шагнал авна. Эндээс бид энэ төлвийн үнэлгээ болон шагналыг ашиглан хуучин төлвийн үнэлгээг бодон хуучин төлвийн утгыг шинэчилнэ.  $\alpha$  утгыг суралцаж эхлэх үед өндөр утгатай байх бөгөөд суралцах явцад багасгах хэрэгтэй.

## Epsilon greedy

Агент суралцаж эхлэх үед огт мэдлэг байхгүй буюу Q утгууд зөв мэдээлэл өгөхгүй учир санамсаргүй үйлдэл олон хийж турших (*exploration*) шаардлагатай. Суралцах явцдаа санамсаргүй үйлдэл хийхээ багасгаж оновчтой үйлдлүүдээ хийх (*exploitation*) хэрэгтэй.  $\epsilon$ -greedy нь  $\epsilon$  магадлалтайгаар санамсаргүй үйлдэл хийж  $1 - \epsilon$  магадлалтайгаар оновчтой үйлдлээ хийнэ. Сургалт эхлэх үед  $\epsilon$  нь өндөр утгатай байх ба суралцах явцдаа бууруулж хангалттай суралцаж дуусах үед 0 болох хэрэгтэй.

## 1.3 Гүн Q-сүлжээ (Deep Q-Network)

Tabular Q-Learning ашиглан олон хэмжээст мэдрүүрийн оролтоос (Зураг, яриа ) төлвүүдийг үүсгэн шийдвэр гаргаж сурах нь практикийн хувьд хүндрэлтэй. Олон хэмжээст мэдрүүрийн өгөгдлөөс Q утгыг тооцоолох функц  $f : S \rightarrow Q$  хэрэгтэй. Олон давхаргат нейрон сүлжээ нь төлөв болон Q утгуудын хамаарлыг илэрхийлэх ойролцоологч функц байх боломжтой. Бид энэ бүлэгт гүн нейрон сүлжээний ойлголт болон DQN алгоритмийн талаар дурьдах болно.

### 1.3.1 Гүн давших чиглэлт нейрон сүлжээ (Deep feedforward networks)

Гүн давших чиглэлт сүлжээ, эсвэл олон давхаргат персептрон нь гүн сургалтын загварын чухал нэг хэсэг билээ. Олон давхаргат нейрон сүлжээний зорилго нь  $f^*$  функцийг ойролцоолон загварчлах юм. Жишээ нь ангилал хийх загварын хувьд  $y = f^*(x)$  нь оролт  $x$  ийг  $y$  ангилалд харгалзуулна. Feedforward сүлжээ нь  $y = f(x; \theta)$  харгалзааг тодорхойлж өгөх бөгөөд оновчтой функцийн ойролцолчлогч үүсгэх  $\theta$  параметрүүдийн утгыг тохируулна. Мэдээлэл  $x$  ээс зөвхөн нэг  $f$  функцээр дамжин өнгөрч  $y$  гаралт руу гарч байгаа учир эдгээр загварийг **feedforward** гэж нэрлэдэг. Моделийн гаралтаас оролт руу буцах гэдрэг холбоо байхгүй. Feedforward нейрон сүлжээг гэдрэг холбоотой болгож өргөтгөсөн загвар рекуррент нейрон сүлжээ(Recurrent Neural Network) болно. Нейрон сүлжээ нь анх үүсэхдээ хүний тархиний бүтцээс санаа авсан ба олон ялгаатай функцийг агуулдаг учир нейрон **сүлжээ** гэсэн гэж нэрлэсэн. Зөвхөн нэг давхар шугаман загвар (шугаман регресс эсвэл логистик регресс) нь зөвхөн шугаман загвар гаргах боломжтой. Гэвч бид шугаман бус функцийг загварчлах шаардлагатай үед оролтын өгөгдөл  $x$  ийг бус  $\phi(x)$  шугаман бус хувиргалт хийсэн гаралтыг шугаман хувиргалт

хийн загварчлах боломжтой. Оролтын өгөгдөл шугаман хувиргалт хийсний дараа шугаман бус функцээр хувиргах үйлдлийг олон дахин хийснээр бид шугаман бус комплекс загварчлагч функц үүсгэх боломжтой. Гүн сургалтын гол зорилго нь  $y = f(x; \theta, \omega) = \phi(x; \theta)^\top \omega$  байх  $\phi$  г сурах явдал юм. Оролтын өгөгдлийг шугаман бус хувиргалт хийж буй  $\phi(x; \theta)$  функцийг бид **нууц давхарга** гэж нэрлэнэ. Нейрон сүлжээг сургах нь  $y = f(x; \theta)$  оновчтой  $\theta$  параметрийг олох ба үр дүнд нь функцийг ойролцоологч оптимал загвар гарна. [4]

### 1.3.2 Шугаман бус идэвхжүүлэгч функц

Хэрэв нэг давхаргаас дараагийн давхарга руу шилжихдээ шугаман бус хувиргалт хийхгүй бол эцсийн дүндээ нэг удаагийн шугаман үйлдэл болно. Хамгийн өргөн хэрэглэгддэг шугаман бус идэвхжүүлэгч функц уламжлалын утгыг бодоход төвөггүй байх шаардлагатай бөгөөд өргөн хэрэглэгддэг функцууд нь сигмойд, релү, гиперболлог тангенс зэрэг функцууд байна. Нэг давхаргаас дараагийн давхарга хооронд шугаман хувиргалт хийсний дараа ямар нэг шугаман бус үйлдэл хийж олон давхартгат нейрон сүлжээг хүчирхэг шугаман бус загварчлагч болгоно.

$$L_i(\theta_i) = \mathbb{E}_{s, a \sim \rho(\cdot)} [(y_i - Q(s, a; \theta_i))^2] \quad (1.15)$$

,энд  $y_i = \mathbb{E}_{s' \sim \mathcal{E}} [r + \gamma \cdot \max_{a'} Q(s', a'; \theta_{i1}) | s, a]$  нь  $i$  дахь давталтын зорих утга(target) ба  $\rho(s, a)$  нь төлөвүүд  $s$  болон үйлдлүүд  $a$  ийн магадлалын тархалт байна.

$Q$ -сүлжээг давталтын  $i$  утга бүрт тэгшигтгэл 1.15 -д  $L_i(\theta_i)$  алдааны функцийн утгыг багасгах чиглэлд нейрон сүлжээний параметруудийг тохируулан сургаж болно.

### 1.3.3 Experience replay

Агшин бүрт агентийн авч буй дараалсан  $(s, a, r, s')$  хосууд хоорондоо өндөр хамааралтай тул нейрон сүлжээг агшин бүрт авч буй туршлагаар сургах нь хүндрэлтэй. Бид *experience replay* [5] аргыг ашиглана. Энэ арга нь агентийн туршлага  $e_t = (s_t, a_t, r_t, s_{t+1})$  -г хугацааны агшин бүрт  $D = e_1, \dots, e_N$  санд хадгална.  $Q$  утгыг тооцоолох нейрон сүлжээг сайжруулахдаа сангаас санамсаргүйгээр багц авах бөгөөд тэр багцаас  $Q$ -сургалтын алгоритм ашиглан  $Q$ -утгыг шинэчлэн алдааны функцийг бодон параметруудийг өөрчлөнө. Агшин бүрт  $\epsilon$ -greedy бодлогоор үйлдэл сонгоно.

### 1.3.4 Double Q-Learning

Тэгшитгэл 1.16 -д дараагийн төлвийн үнэлгээг бодохдоо үйлдэл сонгох болон тооцоолох үйлдлийг нэг ижил функц ашиглан боддог. Гэвч энэ үнэлгээ нь ойролцоолчилсон утга буюу зөв утга биш. Тэгэхээр өндөрөөр үнэлсэн утгуудыг(overestimation bias) сонгох эрсдэлтэй.

$$\max_a Q_t(S_{t+1}, a) = Q_t(S_{t+1}, \operatorname{argmax}_a Q_t(S_{t+1}, a)) \quad (1.16)$$

Давхар Q-Сургалт нь хоёр ялгаатай  $Q, Q'$  функцийг ашиглах ба  $Q$  утгыг таамаглахдаа 1.17 болон 1.18 тэгшитгэлийн аль нэгийг санамсаргүйгээр сонгоно. [6]

$$R_{t+1} + \gamma Q'_t(S_{t+1}, \operatorname{argmax}_a Q_t(S_{t+1}, a)) \quad (1.17)$$

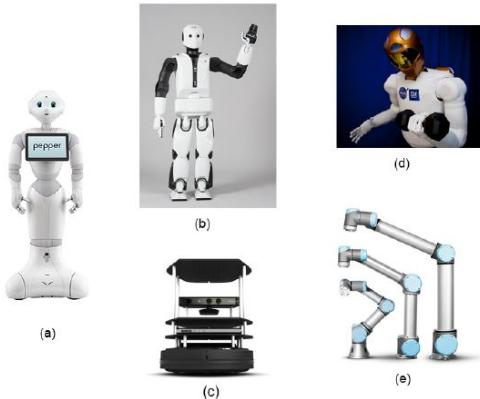
$$R_{t+1} + \gamma Q_t(S_{t+1}, \operatorname{argmax}_a Q'_t(S_{t+1}, a)) \quad (1.18)$$

Үйлдэл хийхдээ хоер функцийг давхар ашиглаж болно. ( $\hat{Q}(s, a) + Q'(s, a)$ )

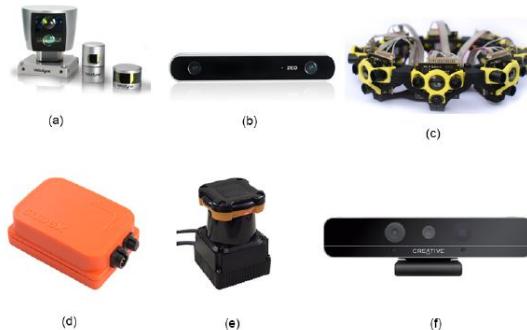
## 1.4 Робот үйлдлийн систем

ROS(Robot Operating System) нь робот програмчлахад зориулсан нээлттэй эхтэй, уян хатан програм хангамжийн фреймворк юм. ROS нь хөгжүүлэгчид роботик хэрэглээг техник хангамжид санаа зовохгүйгээр програмчлах боломжтой. ROS нь мөн роботийн өгөгдлийг дүрслэх олон програмийг агуулдаг. ROS фреймворкийн гол цөм нь хоёр өөр төхөөрөмж дээр ажиллаж байсан ч өгөгдөл солилцдог мессеж дамжуулах холбогч суурь програм хангамж юм. ROS дахь програм хангамж нь package уудыг бүрдүүлдэг бөгөөд энэ нь модульчлал болон код дахин ашиглах давуу талыг олгодог. ROS ын өгөгдөл дамжуулах дундын програм хангамж болон техник хангамж хийсвэрчлэх давхаргыг ашиглан хөгжүүлэгчид олон төрлийн роботийн хэрэглээг шийдэх боломжтой. Ихэнх роботийн чадамжууд роботоос хамааралгүй учраас бүх төрлийн роботууд ашиглах боломжтой. Шинээр хийсэн робот тухайн үйлдлийг хийх package доторх програмийг өөрчлөхгүйгээр шууд ашиглах боломжтой. ROS нь их сургуулиудын үүсгэсэн өргөн хамтын ажиллагаатай бөгөөд олон хөгжүүлэгчид хувь нэмрээ оруулдаг. Идэвхтэй хөгжүүлэгчдийн экосистем нь ROS ийг бусад фреймворкуудаас ялгаж өгдөг. ROS нь албан ёсоор Linux Ubuntu үйлдлийн системийг дэмждэг.[7]

### 1.4.1 ROS дэмждэг роботууд болон мэдруүрүүд



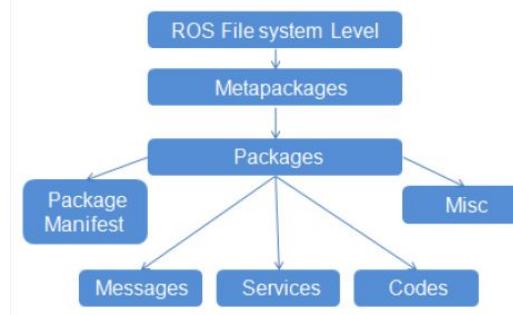
ЗУРАГ 1.4: ROS ашигладаг өргөн хэрэглэгддэг роботууд  
Зураг дахь роботуудийн нэр нь: Pepper (a), REEM-C (b), TurtleBot (c),  
Robonaut (d), Universal Robots (e).



ЗУРАГ 1.5: ROS дэмждэг роботийн мэдруүрүүд  
Зураг харуулсан мэдруүрүүдийн нэр нь: Velodyne (a), ZED Camera (b),  
Teraranger (c), Xsens (d), Hokuyo Laser range finder (e), and Intel RealSense  
(f).

### 1.4.2 ROS filesystem level

- **Metapackages:** Metapackage нь цугтаа тодорхой үйлдлийг гүйцэтгэх package уудаас тогтоно. Жишээ нь: мобайл роботийн удирдлагад зориулсан metapackage байж болно. Энэ нь харгалзах package уудын мэдээллийг хадгалах ба эдгээр package уудыг автоматаар татаж авна.
- **Packages:** ROS доторх програмууд ROS package уудаас бүрддэг. ROS package нь ROS ийн үндсэн нэгж гэж ойлгож болно. Package нь ROS nodes/processes, өгөгдлийн сан, тохиргооны файлуудыг агуулан нэг модуль болно.

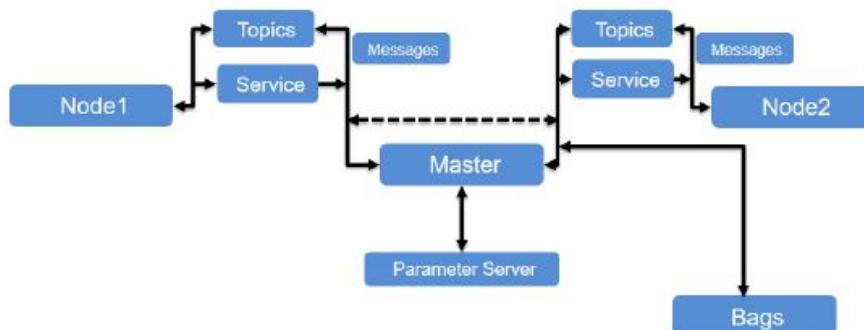


ЗУРАГ 1.6: ROS файлсистем

- **Package manifest:** Package бүр дотроо package.xml файлийг агуулна. Энэ файл нь нэр, хувилбар, зохиогч, лиценз баолон package ийн хамаарлуудын мэдээллийг агуулна. Metapackage ийн package.xml файл харгалзах package уудын нэрүүдээс тогтоно.
- **Messages(msg):** ROS нь мессежээр хоорондоо холбоо тогтоодог. Мессеж өгөгдлийн төрлийг .msg өргөтгөлтэй файл дотор хадгалж болно. our\_package/msg/message\_files.msg. файл дотор мессеж файлийг хадгалж өгдөг тогтсон журамтай.
- **Service(srv):** Тооцооллын графын түвшний нэг ойлгол нь service. ROS мессажтэй төстэйгээр our\_package/srv/service\_files.srv. файлд service ийн тодорхойлолтыг бичиж өгнө.

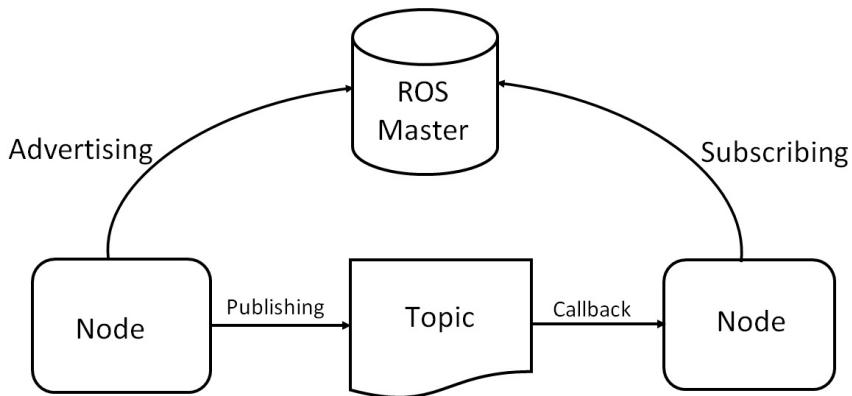
#### 1.4.3 ROS тооцооллын графын түвшин

The ROS computation graph is the peer-to-peer network of the ROS process, and it processes the data together. The ROS computation graph concepts are nodes, topics, messages, master, parameter server, services, and bags:



ЗУРАГ 1.7: ROS тооцооллын граф

- **Nodes:** ROS node үүд нь бусадтайгаа харилцахын тулд ROS API ашиглаж буй процессууд юм. Нэг робот тооцоолол хийхийн тулд олон node тэй байж болно. Жишээ нь автомат робот лазер унших, мотор удирдах, баиршилаа тогтоох гэх мэт үйлдэл бүрт харгалзах node тэй байж болно.
- **Master:** ROS мастер нь node- үүдийн хоорондох холбоог тогтоох дундын node ийн үүргийг гүйцэтгэдэг. Master -аас ROS ийн орчин дахь бүх node үүдийн мэдээллийг авч болно. Хоёр node хоорондоо холболт үүсгэхийн тулд node бүрийн хооронд мэдээлэл солилцно. Мэдээлэл солилцсоны дараа хоёр ROS node холбоо эхэлнэ.
- **Parameter server:** Node параметер серверт хувьсагч агуулж болох ба түүний нууцлалыг тохируулж болдог. Хэрэв параметер глобалт хүрээтэй бол бусад бүр node үүд хандаж болно. ROS параметер нь ROS мастертай цуг ажилладаг.
- **Messages:** ROS node үүд хоорондоо олон замаар харилцах боломжтой. Бүх аргуудын хувьд node үүд өгөгдлийг ROS мессежийн хэлбэрээр илгээх болон хүлээж авна. ROS мессеж нь .ROS node үүдийн өгөгдөл солилцох өгөгдлийн формат.
- **Topics:** Хоёр ROS node хооронд ROS мессеж солилцох нэг арга нь ROS topic. Topic нь ROS мессеж ашиглан өгөгдөл солилцох бааз болно. Topic бүр тодорхой нэртэй байх ба нэг node topic руу өгөгдөл шидэх ба өөр нэг node уг topic руу буртгүүлэн өгөгдлийг уншин авах боломжтой. Зураг 1.8 д хоёр node topic оор дамжуулан өгөгдөл солилцох процесийг харуулав.
- **Services:** Service үүд нь topic- тай төстэй өөр нэг харилцааны арга. Topic нийтлэх(subscribe) болон бүртгүүлэх(subscribe) үйлдлүүдийг ашигладаг бол service хүсэлт(request) болон хариу(reply) ашигладаг. Нэг node service гүзүүлэх ба client node server ээс service ийн хүсэлт явуулна. Server нь хүсэлтийн програмийг биелүүлэх ба үр дүнг client node руу буцааж илгээнэ. Client node server хариу үйлдэл үр дүнтэй хамт илгээх хүртэл хүлээх шаардлагатай.
- **Bags:** Bag ууд нь ROS topic уудын утгыг хадгалах болон дахин тоглуулахад хэрэгтэй хэрэглүүр юм. Роботтой ажиллаж байх үед бодит техник хангамж байхгүй байх тохиолдолд бид rosbag ашиглан мэдрүүрийн өгөгдлийг хурааж аван bag file ийг бусад компьютерт хуулан өгөгдлийг тоглуулан шинжлэх боломжтой.



ЗУРАГ 1.8: ROS өгөгдөл солилцох процесс

#### 1.4.4 ROS Хэрэглүүрүүд

##### Rviz(ROS Visualizer)

Rviz нь topic болон параметрүүдээс авах 2D болон 3D утгуудыг дүрслэх боломжтой ROS ийн 3D дүрслэгч програм. Rviz нь роботийн загвар, 3D (tf) өгөгдөл, цэгүүд , лазер ,дүрсэн өгөгдөл болон бусад олон мэдрүүрүүдийн утгуудыг дүрслэхэд тусалдаг.

##### rqt\_plot

rqt\_plot програм нь ROS topic ийн хэлбэрт байгаа скаляр утгуудын график дүрслэлийг харуулах хэрэглүүр.

##### rqt\_graph

The rqt\_graph ([http://wiki.ros.org/rqt\\_graph](http://wiki.ros.org/rqt_graph)) ROS GUI хэрэглүүр нь ROS node -үүдийн хоорондох холбоосын графийг дүрслэх боломжтой.

#### 1.4.5 ROS simulator

Нээлттэй эхтэй ROS той холбогддог робот симуляторын нэг нь Газебо (). Газебо нь олон төрлийн роботийн модел болон мэдрүүрүүдийг холбох боломжтой динамик робот симулятор юм. Мэдрүүрийн утгууд ROS руу topics, parameters, services ээр дамжин хүрэх боломжтой. Ихэнх роботикс симуляторууд нь хаалттай эхтэй бөгөөд ашиглахад өндөр төлбөртэй байдаг учир Газебо симуляторийг ашиглаж болно.



ЗУРАГ 1.9: Газебо симулятор

---

---

## БҮЛЭГ 2

---

Төслийн хэсэг

Бататган суралцах аргыг бодит физик робот дээр турших нь цаг хугацаа болон эдийн засгийн хувьд хүндрэлтэй. Робот суралцаж эхлэх үед санамсаргүй үйлдлүүд хийх учир өөрийгөө болон бусад зүйлүүдийг гэмтээх эрсдэлтэй учир бид компьютерийн симуляцийн орчинд сургалтаа явуулан туршилт хийсэн. Туршилтыг хийхдээ робот үйлдлийн систем ашиглан Газебо симуляторын орчинд роботийн модел, мэдруүр, актуаторийн тохиргоог хийн туршилт хийсэн. ROS дээр C, C++, Python, Java зэрэг програмчлалын хэлүүдийг ашиглан удирдлагын програмчлал хийх боломжтой бөгөөд бид энэ төсөлд Python хэлийг ашиглан програмчлал хийсэн. Роботийн төлвийг түүн дээр урагш 180°байрласан 5 зай хэмжигч мэдруүрийн утгууд, өнцөг болон шугаман хурдны хослуудаар тодорхойлж өгсөн.

## 2.1 Симуляц дахь туршилт

Нейрон сүлжээг сургахын тулд уг сүлжээний алдааны функцийн параметруудийн тухайн уламжлал бодох шаардлагатай ба үүнийг Tensorflow дээр суурилсан Keras сан ашигласан ба б цөмтэй Intel Xeon CPU тэй NVIDIA GeForce GTX Titan X GPU тэй компьютер дээр 5 дахин хурдаасан Газебо симулятор ашигласан.

### 2.1.1 Робот үйлдлийн систем дээрх роботийн симуляцийн тохиргоо

Роботийг симуляцийн орчинд ажиллуулахын тулд эхлээд түүний 3D моделийг гаргах шаардлагатай. ROS нь моделийг загварчлах стандарт meta-package тай бөгөөд үүнд urdf, kdl\_parser, robot\_state\_publisher, collada\_urdf гэх мэт package ууд орно. Эдгээр package ууд нь бодит техник хангамжийг характеристикуудыг агуулсан роботийн 3D моделийг гаргахад туслана. Бид роботийн загвар, мэдруүр, орчнийг тодорхойлж өгөхдөө URDF(Unified Robot Description Format) форматийг ашигладаг бөгөөд үүнд холбоос(link), ye(joint) уудыг тодорхойлж өгнө (<http://wiki.ros.org/urdf/XML>). Комплекс роботийн 3D загварыг urdf форматаар тодорхойлж өгөх нь төвөгтэй учир AutoCAD, Solidworks болон Blender гэх мэт CAD програмийг ашиглах боломжтой.

Роботийн 3D загварыг тодорхойлж өгсөний дараа мэдруүр болон актуаторуудийн интерфейсийн тохиргоог хийж өгөх шаардлагатай. Үүнийг <gazebo>

```

<gazebo reference="hokuyo_link">
  <material>Gazebo/Blue</material>
  <turnGravityOff>false</turnGravityOff>
  <sensor type="ray" name="head_hokuyo_sensor">
    <pose>${hokuyo_size/2} 0 0 0 0 0</pose>
    <visualize>true</visualize>
    <update_rate>200</update_rate>
    <ray>
      <scan>
        <horizontal>
          <samples>5</samples>
          <resolution>1</resolution>
          <min_angle>-1.57</min_angle>
          <max_angle>1.57</max_angle>
        </horizontal>
      </scan>
      <range>
        <min>0</min>
        <max>2</max>
        <resolution>0.1</resolution>
      </range>
    </ray>
    <plugin name="gazebo_ros_head_hokuyo_controller" filename="libgazebo_ros_laser.so">
      <topicName>/scan</topicName>
      <frameName>hokuyo_link</frameName>
    </plugin>
  </sensor>
</gazebo>

```

ЗУРАГ 2.1: Hokuyo лазерийн газебо симуляторт холбох холболт

</gazebo> таг дотор тодорхойлж өгнө. Зай хэмжих лазерийг Hokuyo лазерийн драйверийг зайд хэмжихийн тулд оруулсан. Роботийн удирдлагыг хийхдээ дифференциал удирдлагын драйверийн тохиргоог баруун болон зүүн дугуйнд тохируулж өгсөн. Зураг 2.1 д зай хэмжих лазер мэдрүүрийн симуляц дээрх холболтыг харуулав. Энд лазерийн цацрагийн тоо, хэмжих өнцгийн дээд доод хязгаар, утгыг илгээх топик, хэмжих зайны дээд доод хязгаар, нарийвчлал, мэдрүүрийн бие болох линк зэргийг тодорхойлж өгнө.

Зураг 2.2 д роботийн дифференциал удирдлагын тохиргоог харуулав. Энд роботийн хоёр дугуй, моторийн хурдатгал, эргэх хүчний момент(torque), коммандийн утгыг илгээх топик зэргийг тодорхойлж өгнө.

Хэрэв роботийн удирдлага нь олон node тэй бол бүгдийг тусад нь терминал дээр нэг нэгээр нь ажиллуулах нь хүндрэлтэй. launch өргөтгөлтэй файл дотор <launch></launch> таг дотор package болон node үүдийг тодорхойлж өгснөөр бүх node үүдийг нэг коммандаар ажиллуулж болно.

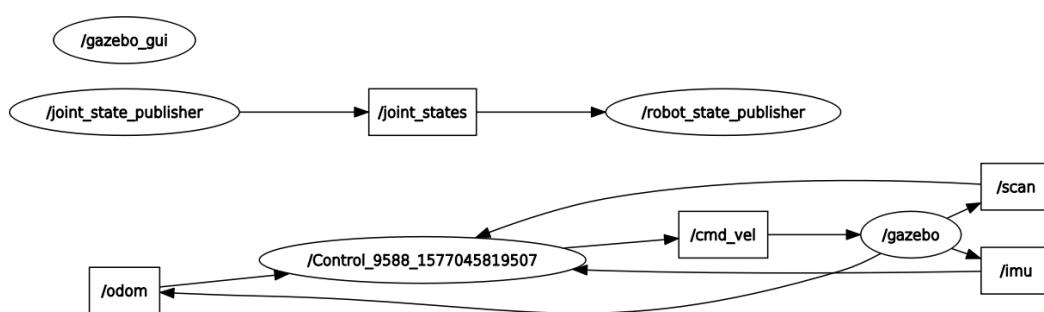
```

<gazebo>
  <plugin name="differential_drive_controller" filename="libgazebo_ros_diff_drive.so">
    <legacyMode>true</legacyMode>
    <rosDebugLevel>Debug</rosDebugLevel>
    <publishWheelTF>false</publishWheelTF>
    <robotNamespace>/</robotNamespace>
    <publishTf>1</publishTf>
    <publishWheelJointState>false</publishWheelJointState>
    <alwaysOn>true</alwaysOn>
    <updateRate>100.0</updateRate>
    <leftJoint>front_left_wheel_joint</leftJoint>
    <rightJoint>front_right_wheel_joint</rightJoint>
    <wheelSeparation>${2*base_radius}</wheelSeparation>
    <wheelDiameter>${2*wheel_radius}</wheelDiameter>
    <broadcastTF>1</broadcastTF>
    <wheelTorque>30</wheelTorque>
    <wheelAcceleration>1.8</wheelAcceleration>
    <commandTopic>cmd_vel</commandTopic>
    <odometryFrame>odom</odometryFrame>
    <odometryTopic>odom</odometryTopic>
    <robotBaseFrame>base_footprint</robotBaseFrame>

  </plugin>
</gazebo>

```

ЗУРАГ 2.2: Роботийг дифференциал удирдлагын драйвераар  
удирдах симуляторийн холболт



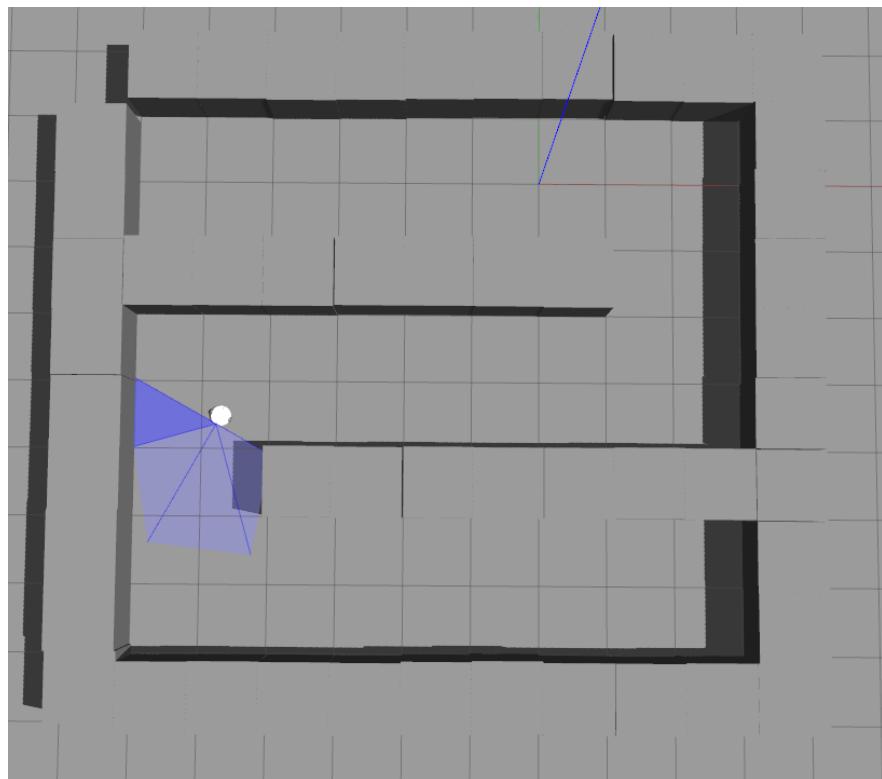
ЗУРАГ 2.3: Симуляцийн орчин дахь ROS тооцооллын граф

### 2.1.2 Q-Learning

Таван лазерийн цацраг бүрийн хэмжих хязгаар нь 2м ба уг утгуудыг тоймлон 0,1,2 гэсэн 3 боломжит ялгаатай утгатай ба шугаман болон өнцөг хурд тус бүр эерэг, сөрөг, 0 гэсэн 3 ялгаатай боломжит утгатай байхаар тус тус тохируулсан. Ингэснээр нийт боломжит ялгаатай төлвүүдийн тоо нь  $|S| = 3^7 = 2187$  байна. Нийт боломжит үйлдлүүдийг шугаман болон өнцөг хурдны ([1,0], [0.7,0.5], [0.7,-0.5], [0,1], [0,-1], [-0.8,0]) хослолуудаар тодорхойлсон. Нийт тэмдэглэх ёстой Q хүснэгтийн хэмжээ нь 2187x6 болно. Роботийн авах шагналыг шугаман хурд болон зайд хэмжигчийн утгуудаас хамааруулан 2.1 тэгшитгэлээр зохиож өгсөн.

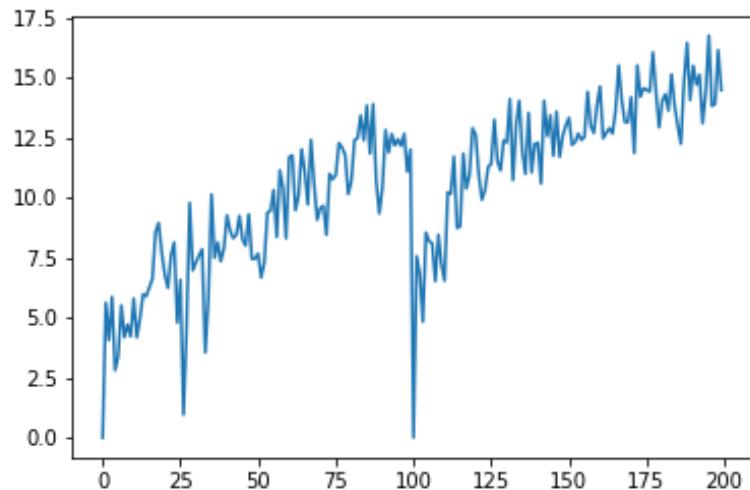
$$R = \max(v, 0) + 0.5 \cdot (sensor\_0 + sensor\_4) + sensor\_1 + sensor\_3 + 2 \cdot sensor\_2 \quad (2.1)$$

`sensor_x` нь баруун талаас эхлэн зайдын утгууд болно. Захын 2 мэдрэгчийн утга шагналд голын зайд мэдрэгчийн утгаас бага нөлөө үзүүлэхээр зохиож өгсөн.

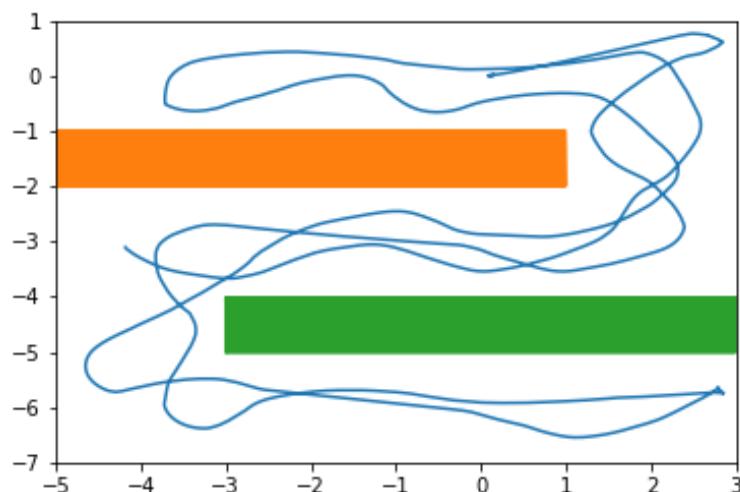


ЗУРАГ 2.4: Роботийн симуляцийн орчин

Зураг 2.5 -д сургалтын явцад 100 алхам тутамд авч буйд дундаж шагналыг харуулав.



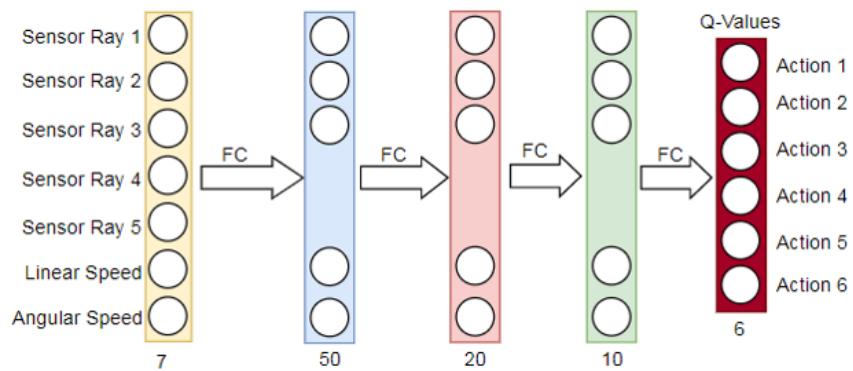
### Зураг 2.5: Дундаж шагнал



ЗУРАГ 2.6: Сургасны дараах роботийн явсан зам

### 2.1.3 DQN

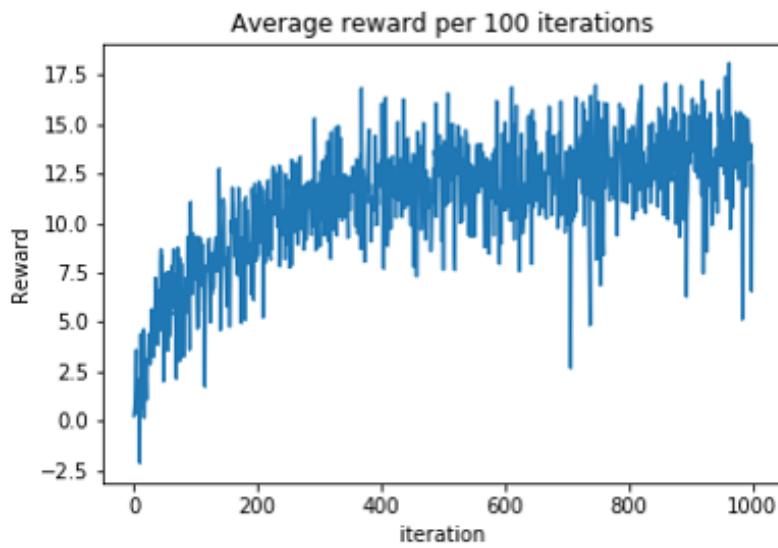
Мэдрүүрүүдийн утгууд, шугаман болон өнцөг хурдуудын нийт ялгаатай төлвүүд төгсгөлгүй учир бүх боломжит төлвүүдийг тэмдэглэж авах нь практикт боломжгүй. Олон давхаргат гүн нейрон сүлжээ нь таван лазерийн хэмжсэн зайны утгууд, шугаман болон өнцөг хурдны утгуудыг оролтод аван бид нарт үйлдэл бүрт харгалзах  $Q$  утгуудыг тооцоолон өгөх боломжтой. Зураг 2.7 д  $Q$  утгыг таамаглах нейрон сүлжээний архитектурийг харуулав. Дээр тодорхойлж өгсөн архитектуртай нейрон сүлжээний загвар нь 1696 параметртай бөгөөд sigmoid шугаман бус идэвхжүүлэгч функц ашигласан. Replay buffer



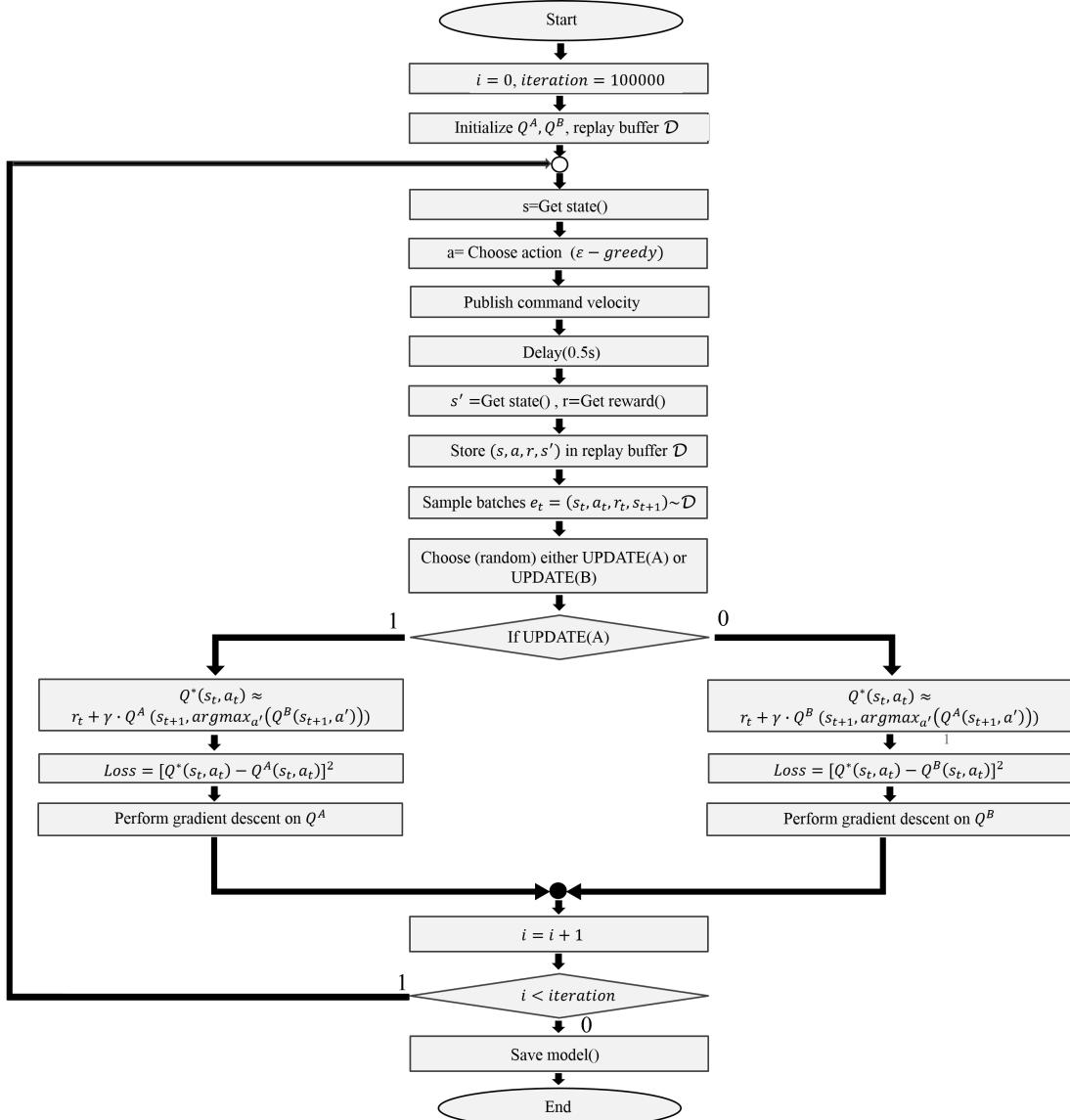
ЗУРАГ 2.7: Q network архитектур

ийн хэмжээг 4000 байхаар өгсөн ба нэг үйлдэл хийх буюу хурдны коммандыг илгээснээс 0.5 секундын дараа шинэ төлвийг симулятороос уншин авч буффер руу нэмсэн ба нэг удаа  $(s, a, s', R)$  хосыг буфферээс уншин авч Q утгыг таамаглах нейрон сүлжээг 10 удаа (epoch=10) сургасан бөгөөд энэ процессийг нийт 100000 удаа давталт хийсэн. Онолын хэсэгт дурьдсанаар дараагийн төлвийн үнэлгээг хэт өндөр утгаар тооцох хазайлтаас (Overestimation bias) сэргийлэхийн тулд хоёр үл хамаарах Double DQN ашиглан туршилт хийсэн. Компьютер дээр нийт 12 цаг туршилт хийсэн.

Зураг 2.8 -д сургалтын явцад 100 алхам тутамд авч буй дундаж шагналыг харуулав. Агент суралцаж эхлэх үед авч буй дундаж шагнал нь бага байсан боловч давталт хийх явцад дундаж шагнал нь ихэсч байна.



ЗУРАГ 2.8: Дундаж шагнал

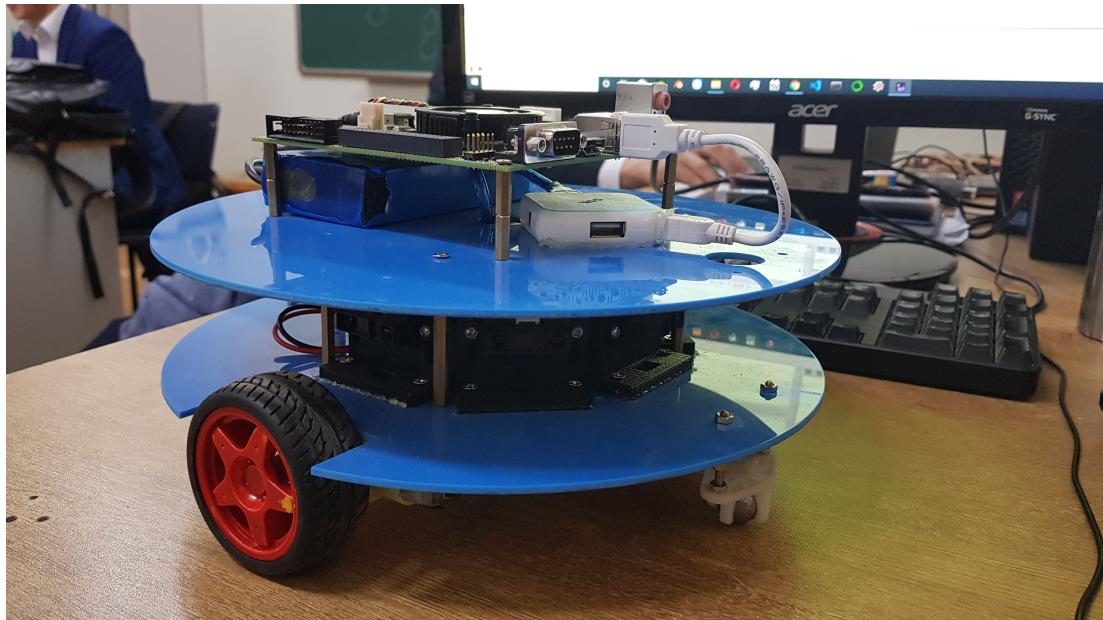


ЗУРАГ 2.9: Double-DQN сургах алгоритм

## 2.2 Бодит туршилт

### 2.2.1 Техник хангамж

Симуляцийн орчин дахь роботийн характеристикийг бодит роботтой ижил байлгахын тулд зай хэмжих лазер мэдрүүрийн оронд үнэ өртөг багатай хэт улаан туяаны зайд хэмжих мэдрүүр ашиглан зайд хэмжин туршсан.



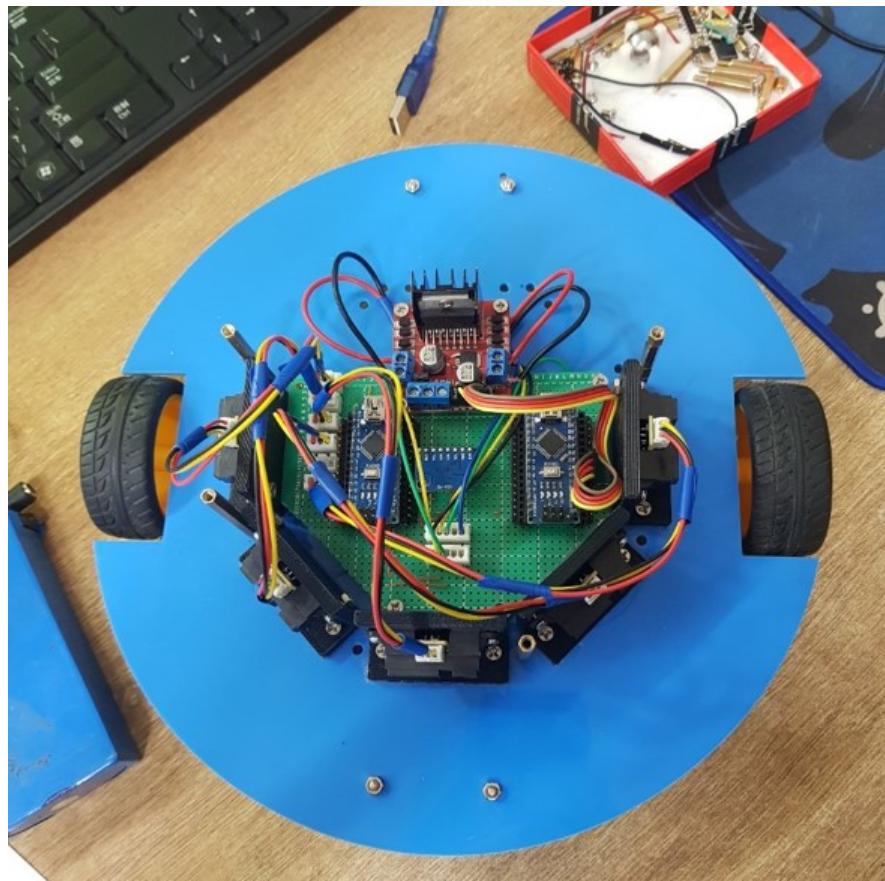
ЗУРАГ 2.10: Бодит туршилтын робот

### Зайд хэмжих хэт улаан туяаны мэдрүүр

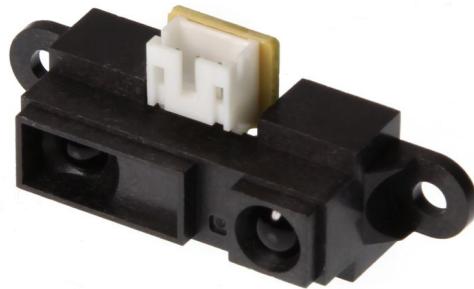
Хэт улаан туяаны Sharp 2Y0A21F98 мэдрүүр нь аналог гаралттай. Уг мэдрүүр нь хамгийн ойрдоо 5 см зайд хэмжих тул уг мэдрүүрүүдийг роботийн ирмэгээс 5 см дотогш зайд байрлуулсан. Мэдрүүрийн хэмжих хамгийн хол зайд туршилтаас 60 см гэж тооцоолсон. Ардуино микроконтроллер нь 10 битийн ADC тэй ба  $V_{ref} = 5V$ . Туршилтаар ATX ын өгөгдөл шуугиантай байсан учир moving-average шүүлтүүр ашигласан. Ингэхдээ хамгийн сүүлийн 10 өгөгдлийг эрэмблэн голын 6 элементийн дундаж утгыг цааш тооцоонд ашигласан.

Зураг 2.14 -д ATX болон зайн хамаарлыг харуулав. Эндээс зайд болон ATX ын утгын хамаарлын тэгшитгэл нь:

$$distance(m) = 53.09 * \cdot adc\_value^{-1.013}. \quad (2.2)$$



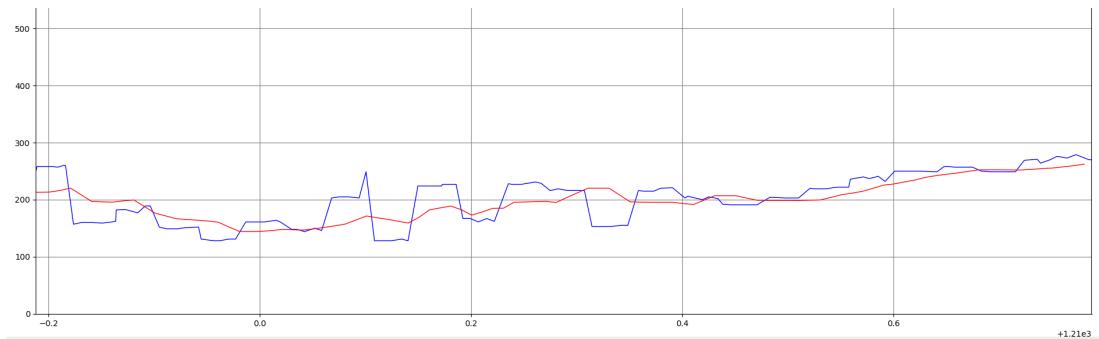
ЗУРАГ 2.11: Бодит туршилтын робот



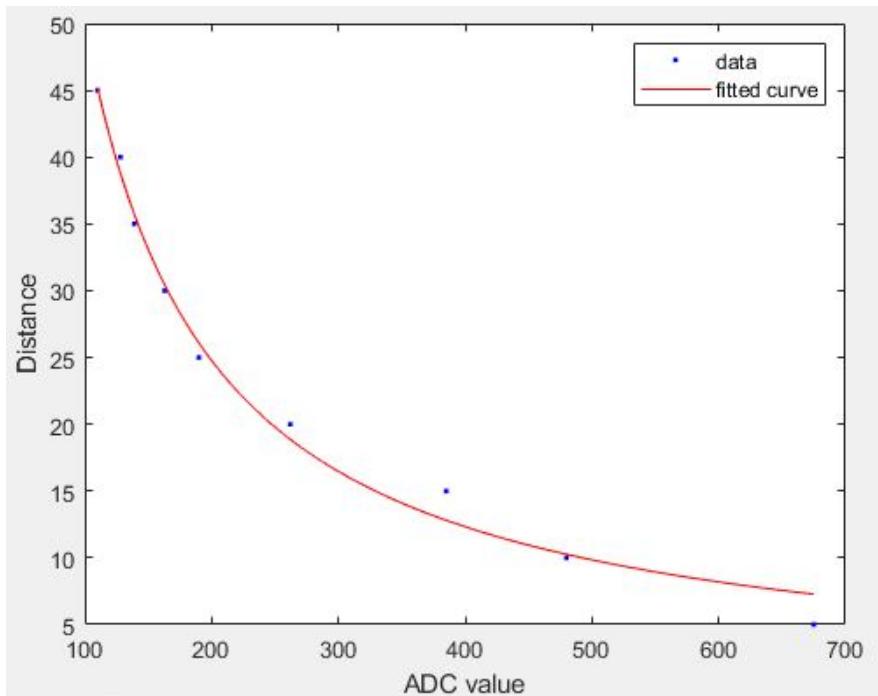
ЗУРАГ 2.12: SHARP GP2Y0A21YK0F зайд хэмжих мэдрүүр

### Тооцоолол хийх компьютер

Симуляцийн орчинд сургасан DQN загвараа бодит мобайл роботийн удирдлагад ашиглахын тулд NVIDIA Jetson TK1 эмбэддэд компьютер дээр ажиллуулан роботийн удирдлагыг гүйцэтгэсэн. TK1 боард дээр Линукс төрлийн Ubuntu 14.04 үйлдлийн системийг дэмждэг бөгөөд ROS Indigo суулгасан.



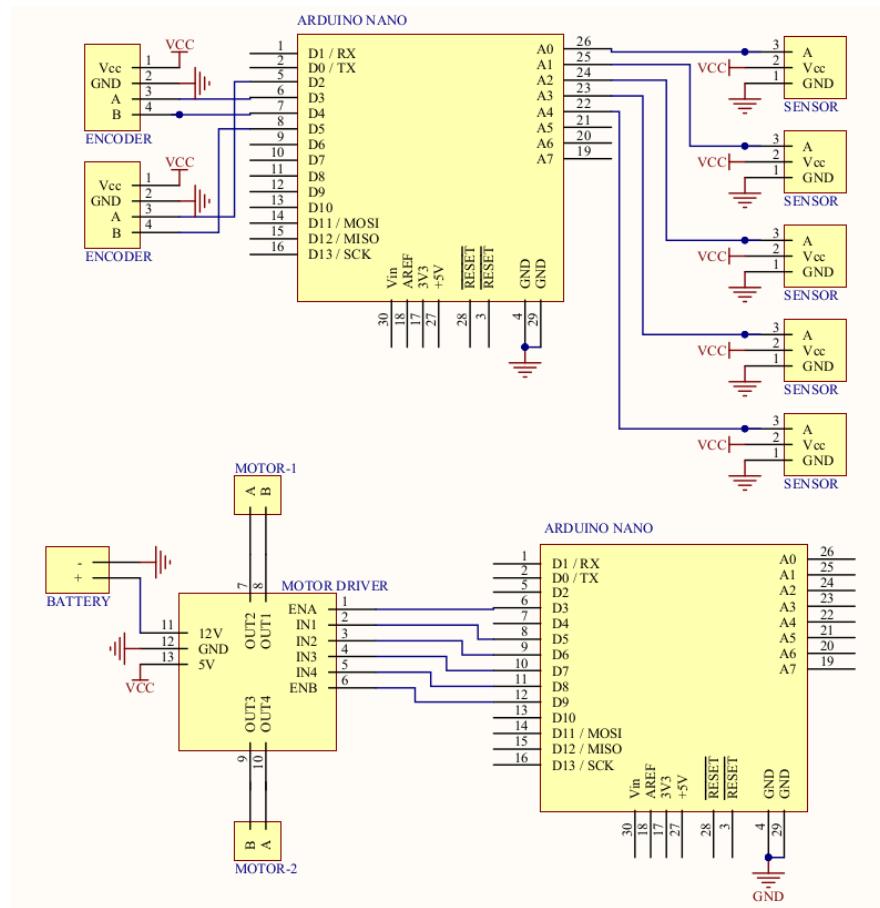
ЗУРАГ 2.13: Мэдрүүрээс авсан өгөгдөл болон шүүлтүүрээр шүүсэн утга



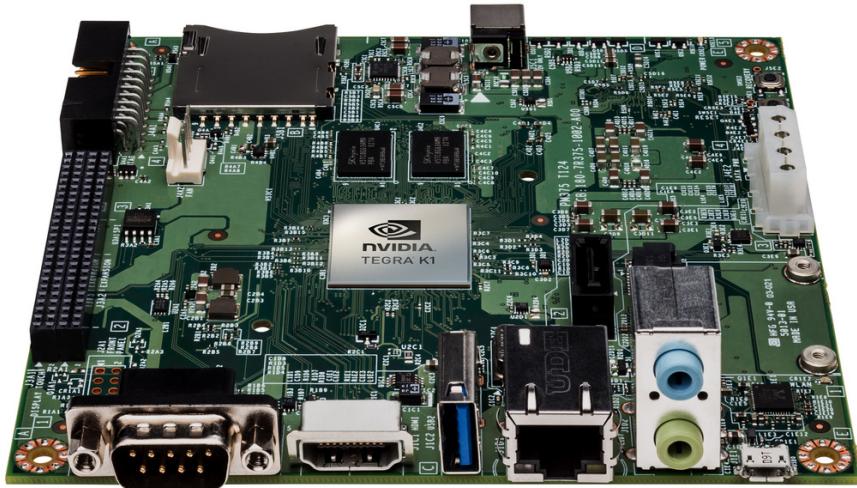
ЗУРАГ 2.14: Мэдрүүрээс объект хүртэлх зайд болон ATX ын утгын хамаарал

ATX NVIDIA Jetson TK1 боард дээр байхгүй учир IR мэдрүүрийн утгыг уншиж боловсруулахын тулд "Arduino Nano" микроконтроллер ашиглан уншин авч TK1 руу UART цуваа интерфейсээр дамжуулахаар тохируулсан. Моторийн хурдыг хэмжих энкодерийн утгыг унших болон тогтмол гүйдлийн моторийн хурдыг удирдах импульсийн өргөний модуляцлагдсан дохио болон чиглэлийг удирдах дохиог мөн "Arduino Nano" микроконтроллер ашиглан хийсэн. ROS нь ардуйно микроконтроллерийг дэмжих **rosserial\_arduino** package тай тул Arduino микроконтроллероос мэдрүүрийн өгөгдөл унших болон моторийн коммандийг илгээх үйлдлийг шууд ROS ашиглан хийсэн .

- 4 цөмтэй ARM Cortex-A15 CPU



ЗУРАГ 2.15: Зарчмын схем



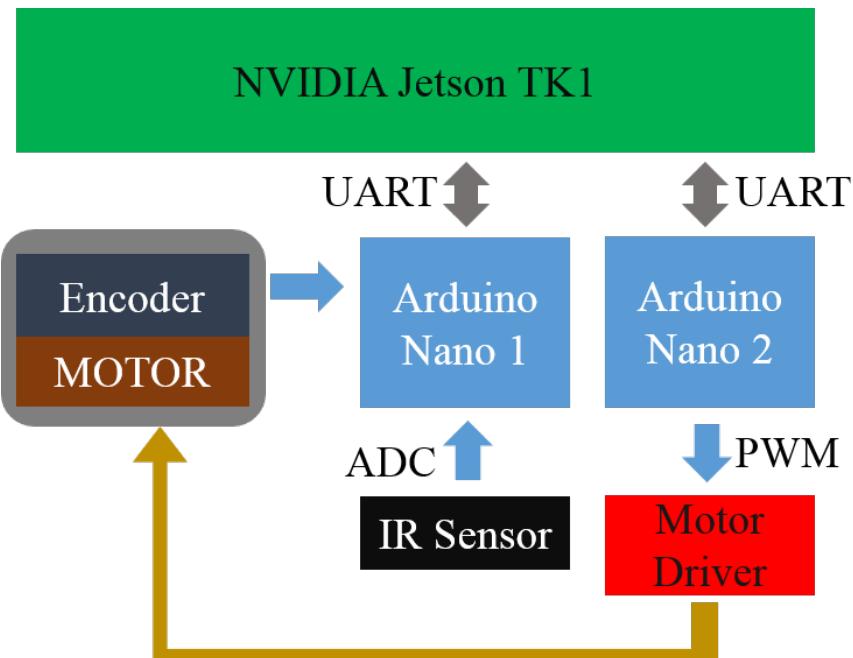
ЗУРАГ 2.16: NVIDIA Jetson TK1

- 64 битийн өргөн 2GB шуурхай санах ой
- 192 CUDA цөмтэй NVIDIA Kepler GPU
- 16 GB 4.51 eMMC memory

- 1 Half mini-PCIE slot 1
- USB 3.0 порт
- 1 HDMI порт
- 1 RS232 цуваа порт
- 1 RTL8111GS Realtek GigE LAN
- Full size SD/MMC connector
- 1 ALC5639 Realtek Audio codec with Mic in and Line out
- 1 SATA data port
- SPI 4MByte boot flash

Дараах дохионуудыг нэмэлт портуудаар хүлээн авах боломжтой

- DP/LVDS
- Touch SPI 1x4 + 1x1 CSI-2
- GPIOs
- UART
- HSIC
- I2C

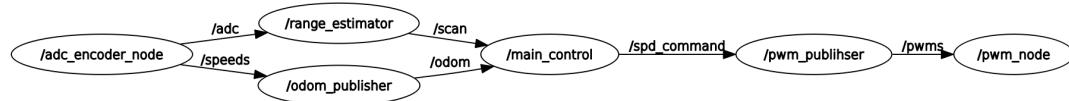


ЗУРАГ 2.17: Бодит роботийн бүтцийн схем

### 2.2.2 Програм хангамж

Робот дээрх хоёр ардуйно нано микроконтроллэрийн нэг нь IR мэдрүүрийн утгыг ATX аар дамжуулан авах болон дугуйн хурдыг хэмжин ROS руу илгээх үйлдлийг хийнэ. Нөгөө микроконтроллёр нь TK1 компьютерийн боловсруулсан моторийн ИӨМ дохио болон чиглэлийг ROS оос уншиж аван мотор драйверийг удирдана. Зураг 2.18-д бодит роботийн програмийн node үүдийн хоорондын өгөгдөл солилцох графийг харуулав. Роботийн удирдлага нь дараах програм буюу node үүдээс тогтоно.

- range\_estimator : Энэ нь ардуйно микроконтроллёр буюу /adc\_encoder\_node ийн /adc топик руу илгээж буй (publish) ATX ын утгыг уншиж аван (subscribe) шүүж зайны утга болгон хувиргаж симуляцийн орчинд ашигласан лазер мэдрүүрийн өгөгдлийн форматад оруулан /scan topic руу илгээнэ.
- odom\_publisher : Ардуйно микроконтроллэрийн дугуйны энкодероос авсан утгаас хурд бодон /speeds топик руу илгээнэ(publish). Уг хурдны утгуудыг уншиж аван subscribe симуляцийн орчин дахь одометрийн өгөгдлийн форматад оруулан /odom топик руу илгээнэ(publish).
- pwm\_publisher : Үндсэн удирдлагын програм (/main\_control node) шугаман болон өнцөг хурдны командийг /spd\_command топик руу илгээх ба тус утгыг уншиж аван (subscribe) түүнд харгалзах баруун болон



ЗУРАГ 2.18: Бодит роботийн тооцооллын граф

зүүн моторийг удирдах ИӨМ ийн утгыг бодон **/pwms** топик руу илгээнэ.

- main\_control : Зай хэмжих мэдрүүрийн утга, шугаман болон өнцөг хурдны утгуудаар төлвийг үүсгэн үйлдэл бүрийн Q утгыг симуляцийн орчинд сургасан DQN загварыг ашиглан таамаглах бөгөөд хамгийн их Q утгатай байгаа үйлдлийн коммандийг **/spd\_command** топик руу илгээнэ.
- adc\_encoder\_node: IR зай хэмжих мэдрүүрийн өгөгдлийг ATX аар уншин **/adc** топик руу илгээх болон энкодерийн утгаас 2 дугуйны хурдыг бодон **/speeds** топик руу илгээх ардуйно микроконтроллёр болно.
- pwm\_node: **/pwm\_publisher** node **/pwms** топик руу баруун болон зүүн моторт өгөх ИӨМ утга болон чиглэлийг илгээх бөгөөд энэ node нь уг топик дээр ирсэн утгыг уншин авч моторийг удирдана.

Симуляцийн орчинд роботийн загварыг симуляторт оруулах тохиргооны файлууд, роботийн сургалт хийж буй код болон бодит роботийн удирдлагын эх кодийг <https://github.com/ganzo0623/Obstacle-avoidance-deep-RL> д нээлттэй байрлуулсан.

## 2.3 ДҮГНЭЛТ

Энэхүү төгсөлтийн ажлаар машин сургалтын нэг салбар болох бататган суралцах аргын онолыг судалж дифференциал удирдлагат хөдөлгөөнт робот системийн удирдлагад хэрэгжүүллээ. Аих тавьсан зорилт болох робот өмнө буй саадыг мөргөхгүйгээр урагш явж сурах алгоритмийг Q-сургалт болон Double-DQN хоёр аргаар симуляцийн орчинд амжилттай туршин бодит робот дээр хэрэгжүүллээ.

Цаашид хөгжүүлэх ажил нь үйлдлийн олонлог нь аналог байх үед удирдлагын загварыг гаргах аргыг судлах хэрэгтэй. Энэ ажлаар үйлдлийн олонлог нь дискрет буюу хийж болох үйлдлүүд нь төгсгөлөг байх үед хэрэгжүүлсэн. Магистр болон докторийн ахисан түвшний ажлаар загвар суурилсан арга(Model based method) болон мета сургалтын аргуудыг судлах болно.

---

---

ХАВСРАЛТ А

---

Хавсралтын нэр

```

1 import rospy
2 import numpy as np
3 import sys
4 import time
5 from tqdm import tqdm
6 import random
7
8 from DQN_functions.mobile_control import deep_mobile_control
9 from DQN_functions.ReplayMemory import ReplayMemory
10
11 from keras import Sequential
12 from keras.layers import Dense,Dropout
13 from keras.optimizers import Adam
14 from keras.models import load_model
15 from keras.callbacks.callbacks import EarlyStopping
16
17 ## extra imports to set GPU options
18 import tensorflow as tf
19 from keras import backend as k
20
21 #####
22 # TensorFlow wizardry
23 config = tf.ConfigProto()
24
25 # Don't pre-allocate memory; allocate as-needed
26 config.gpu_options.allow_growth = True
27
28 # Only allow a total of half the GPU memory to be allocated
29 config.gpu_options.per_process_gpu_memory_fraction = 0.5
30
31 # Create a session with the above options specified.
32 k.tensorflow_backend.set_session(tf.Session(config=config))
33
34
35 gamma = 0.9
36 lr = 0.01
37 lr_decay=0.5
38 num_iterations = 100000
39 epochs=10
40 epsilon = 1
41 epsilon_decay = 0.9999
42 min_epsilon=0.1
43 delay_time=0.5
44 vels=[[1,0],[0.5,1],[0.5,-1],[0,1],[0,-1],[-0.8,0]]
45 legal_actions=len(vels)
46 batch_size=64
47 memory_size=4000

```

```

48 reward_temp=0
49 patience=6
50 rospy.init_node('Mobile_Control', anonymous=True)
51 Robot = deep_mobile_control()
52 memory=ReplayMemory(memory_size)
53
54 Robot.reset()
55 Robot.unpause()
56 optimizer=Adam(learning_rate=lr, beta_1=0.9, beta_2=0.999, amsgrad
    =False)
57 #optimizer=RMSprop(lr=0.001,rho=0.9)
58 #optimizer=SGD(learning_rate=0.001,momentum=0.9)
59
60 model1=load_model('model_1.h5')
61 model2=load_model('model_2.h5')
62
63 early_stop=EarlyStopping(monitor='loss', min_delta=0, patience=
    patience, verbose=0, mode='auto', baseline=None,
    restore_best_weights=True)
64 model1.compile(loss='mean_squared_error',optimizer=optimizer)
65 model2.compile(loss='mean_squared_error',optimizer=optimizer)
66
67 def main():
68     try:
69
70         a=Robot.data_laser
71         a=Robot.imu_x
72         a=Robot.odom_data
73         print(a)
74         test_model()
75
76     except:
77         print("Error occurred. Can't subscribe sensor info")
78     store_transition(delay_time)
79     train_model()
80     test_model()
81 def train_model():
82     global lr,reward_temp
83     append_hist('loss_history.txt',1000)
84     append_hist('reward_hist.txt',30)
85
86     process_bar = tqdm(range(num_iterations))
87     for i in process_bar:
88         if(i%1000==0):
89             model1.save('model_1.h5')
90             model2.save('model_2.h5')
91             lr=lr*lr_decay

```

```

92     if(i%100==0):
93         print('eps:',epsilon)
94         append_hist('reward_hist.txt',reward_temp/100)
95         reward_temp=0
96         check_robot()
97         store_transition(delay_time)
98         optimize_model()
99         model1.save('model_1.h5')
100        model2.save('model_2.h5')
101    def perform_action(a):
102        Robot.vel.linear.x = a[0]
103        Robot.vel.angular.z=a[1]
104        Robot.cmd_vel.publish(Robot.vel)
105    def choose_action(s):
106        global epsilon,min_epsilon
107        #epsilon greedy. to choose random actions initially when Q
108        #network is not trained
109        if np.random.random()<epsilon:
110            action_index=np.random.randint(0,legal_actions)
111            a =vels[action_index]
112            if epsilon>min_epsilon:
113                epsilon = epsilon*epsilon_decay
114                print('Took a random action')
115            else:
116                Q = (model1.predict(np.array([s]))+model2.predict(np.array
117                                ([s])))/2
118                a =vels[np.argmax(Q)]
119                print(Q)
120        return a
121    def get_state():
122        s_temp=Robot.data_laser
123        ang_sp=Robot.odom_data.angular.z
124        lin_sp=Robot.odom_data.linear.x
125        s_temp=np.append(s_temp,[ang_sp,lin_sp])
126        return s_temp
127    def test_model():
128        global epsilon
129        epsilon=0.1
130        while (1):
131            check_robot()
132            s=get_state()
133            #print('eps:',epsilon)
134            a=choose_action(s)
135            #print(a)
136            perform_action(a)
137            try:
138                rospy.sleep(0.1)

```

```

137         except:
138             pass
139     def check_robot():
140         if(abs(Robot imu_x)>0.2 or abs(Robot imu_y)>0.2):
141             Robot.reset()
142         if(abs(Robot odom_pose.x)>10 or abs(Robot odom_pose.y)>10):
143             Robot.reset()
144         if(abs(Robot odom_data.angular.z)>10 or abs(Robot odom_data.
145             linear.x)>10):
146             Robot.reset()
147     def append_hist(a,value):
148         with open(a, 'a') as f:
149             f.write("%s\n" % value)
150     def store_transition(delay_time):
151         global reward_temp
152         s=get_state()
153         a=choose_action(s)
154         perform_action(a)
155         try:
156             rospy.sleep(delay_time)
157         except:
158             pass
159         s1=get_state()
160         reward=max(Robot odom_data.linear.x,0)*30+0.5*(s1[0]+s1[4])
161         +(s1[1]+s1[3])+2*s1[2]-10
162         experience=(s,reward,a,s1)
163         memory.append(experience)
164
165     def optimize_model():
166         global reward_temp
167         if memory.__len__(>batch_size:
168             size=batch_size
169         else:
170             size=memory.__len__()
171         batches=memory.sample(size)
172         states= np.array([batch[0] for batch in batches])
173         rewards= np.array([batch[1] for batch in batches])
174         actions= np.array([batch[2] for batch in batches])
175         new_states= np.array([batch[3] for batch in batches])
176         if np.random.random()<0.5:
177             temp_model1=model1
178             temp_model2=model2
179         else:
180             temp_model1=model2
181             temp_model2=model1

```

```

182     #print(actions)
183     Qs =temp_model1.predict(states)
184     next_state_Qs = temp_model1.predict(new_states)
185     next_state_actions=np.argmax(next_state_Qs, axis=1)
186
187     for i in range(len(rewards)):
188         action_index=vels.index(list(actions[i]))
189         next_state_action_index=next_state_actions[i]
190
191         next_state_value=np.squeeze(temp_model2.predict(new_states[
192             i].reshape(1,-1)))[next_state_action_index]
193         Qs[i][action_index]= rewards[i]+gamma*next_state_value
194
195     history=temp_model1.fit(states,Qs,batch_size=len(batches),
196                             callbacks=[early_stop],epochs =epochs )
197     append_hist('loss_history.txt',history.history['loss'][-1])
198
199 if __name__== "__main__":
200     main()

```

Replay buffer класс

```

1 from collections import deque,namedtuple
2 import random
3
4 class ReplayMemory(object):
5
6     def __init__(self, capacity):
7         self.capacity = capacity
8         self.memory = deque(maxlen=capacity)
9
10    def append(self, pair):
11        """Saves a transition."""
12        self.memory.append(pair)
13
14    def sample(self, batch_size):
15        return random.sample(self.memory, batch_size)
16
17    def __len__(self):
18        return len(self.memory)

```

```

1 import rospy
2 import numpy as np
3 from geometry_msgs.msg import Twist
4 from sensor_msgs.msg import LaserScan
5 from std_srvs.srv import Empty
6 from nav_msgs.msg import Odometry
7 from sensor_msgs.msg import Imu

```

```

8 from std_msgs.msg import Int16
9 cmd_vel="/cmd_vel"
10 laser_topic="/scan"
11 odom_topic="/odom"
12 imu_topic = "/imu"
13 speed_command="/spd_command"
14 class mobile_control():
15     def __init__(self):
16         self.cmd_vel = rospy.Publisher(cmd_vel,Twist,queue_size =1)
17         self.subscriber = rospy.Subscriber(laser_topic,LaserScan,
18             callback=self.laser_callback)
19         self.subscriber_odo=rospy.Subscriber(odom_topic,Odometry,
20             callback=self.odom_callback)
21         self.subscriber_imu=rospy.Subscriber(imu_topic, Imu,
22             callback=self.imu_callback)
23         self.reset = rospy.ServiceProxy("/gazebo/reset_simulation",
24             Empty)
25         self.pause = rospy.ServiceProxy("/gazebo/pause_physics",
26             Empty)
27         self.unpause = rospy.ServiceProxy("/gazebo/unpause_physics"
28             ,Empty)
29         self.i =0
30         self.count =0
31         self.vel=Twist()
32         self.vel.linear.x = 0
33         self.vel.linear.y = 0
34         self.vel.linear.z = 0
35         self.vel.angular.x = 0
36         self.vel.angular.y = 0
37         self.vel.angular.z = 0
38     def odom_callback(self,msg1):
39         self.odom_data=msg1.twist.twist
40         self.odom_pose=msg1.pose.pose.position
41
42     def laser_callback(self, msg):
43         self.laser_data=msg.ranges
44         self.data=np.array(self.laser_data)
45         self.laser_mask=np.logical_or(self.data>2, self.data<0)
46         self.range_angels=np.arange(len(self.laser_data))
47         self.i=0
48         self.range_angels2=self.range_angels[self.laser_mask]
        for self.i in (self.range_angels2):
            self.data[self.i]=2
            self.data_laser=np.round(self.data[:])
49     def imu_callback(self, msg_imu):
50         self.imu_x=msg_imu.orientation.x
51         self.imu_y=msg_imu.orientation.y

```

```

49 class deep_mobile_control():
50     def __init__(self):
51         self.cmd_vel = rospy.Publisher(cmd_vel,Twist,queue_size =1)
52         self.subscriber = rospy.Subscriber(laser_topic,LaserScan,
53             callback=self.laser_callback)
54         self.subscriber_odo=rospy.Subscriber(odom_topic,Odometry,
55             callback=self.odom_callback)
56         self.subscriber_imu=rospy.Subscriber(imu_topic, Imu,
57             callback=self imu_callback)
58         self.spd_command_publisher=rospy.Publisher(speed_command,
59             Int16,queue_size=1)
60         self.reset = rospy.ServiceProxy("/gazebo/reset_simulation",
61             Empty)
62         self.pause = rospy.ServiceProxy("/gazebo/pause_physics",
63             Empty)
64         self.unpause = rospy.ServiceProxy("/gazebo/unpause_physics"
65             ,Empty)
66         self.i =0
67         self.count =0
68         self.spd_comd=Int16()
69         self.spd_comd.data=6
70         self.vel=Twist()
71         self.vel.linear.x = 0
72         self.vel.linear.y = 0
73         self.vel.linear.z = 0
74         self.vel.angular.x = 0
75         self.vel.angular.y = 0
76         self.vel.angular.z = 0
77     def odom_callback(self,msg1):
78         self.odom_data=msg1.twist.twist
79         self.odom_pose=msg1.pose.pose.position
80
81     def laser_callback(self, msg):
82         self.laser_data=msg.ranges
83         self.data=np.array(self.laser_data)
84         self.laser_mask=np.logical_or(self.data>2, self.data<0)
85         self.range_angels=np.arange(len(self.laser_data))
86         self.i=0
87         self.range_angels2=self.range_angels[self.laser_mask]
88         for self.i in (self.range_angels2):
89             self.data[self.i]=2
90         self.data_laser=self.data[:]
91     def imu_callback(self, msg_imu):
92         self.imu_x=msg_imu.orientation.x
93         self.imu_y=msg_imu.orientation.y
94
95 class deep_mobile_control_new(deep_mobile_control):
96     def __init__(self):

```

```

89         deep_mobile_control.__init__(self)
90     def laser_callback(self, msg):
91         self.laser_data=msg.ranges
92         self.data=np.array(self.laser_data)
93         self.laser_mask=np.logical_or(self.data>0.6, self.data<0)
94         self.range_angels=np.arange(len(self.laser_data))
95         self.i=0
96         self.range_angels2=self.range_angels[self.laser_mask]
97         for self.i in (self.range_angels2):
98             self.data[self.i]=0.6
99         self.data_laser=self.data[:]

1 #!/usr/bin/env python2
2 import rospy
3 import numpy as np
4 #import time
5 #import random
6 import os,sys
7 #from DQN_functions.mobile_control import deep_mobile_control
8 #from DQN_functions.ReplayMemory import ReplayMemory
9
10 from mobile_control import deep_mobile_control_new
11 from keras.models import load_model
12
13 epsilon =0.1
14 min_epsilon=0.1
15 epsilon_decay=0.999
16 vels=[[1,0],[0.5,1],[0.5,-1],[0,1],[0,-1],[-0.8,0]]
17 gamma=0.9
18 legal_actions=len(vels)
19 rospy.init_node('Mobile_Control',anonymous=True)
20 Robot = deep_mobile_control_new()
21
22 model1=load_model(os.path.dirname(os.path.realpath(sys.argv[0]))+'
23                 '/model_1.h5')
24 model2=load_model(os.path.dirname(os.path.realpath(sys.argv[0]))+'
25                 '/model_2.h5')
26 def main():
27     test_model()
28     #train_model()
29 def perform_action(a):
30     print('action_number:',a)
31     Robot.spd_comd.data=a
32     Robot.spd_command_publisher.publish(Robot.spd_comd)
33 def choose_action(s):
34     global epsilon,min_epsilon
35     #epsilon greedy. to choose random actions initially when Q
36     is all zeros

```

```

34     if np.random.random()<epsilon:
35         action_index=np.random.randint(0,legal_actions)
36         a =action_index
37         if epsilon>min_epsilon:
38             epsilon = epsilon*epsilon_decay
39             print('Took\underline{random}\uunderline{action}')
40     else:
41         Q = (model1.predict(np.array([s]))+model2.predict(np.array
42             ([s])))/2
43
44         a =int(np.argmax(Q))
45         #a=1
46         print(Q)
47     return a
48 def get_state():
49     s_temp=Robot.data_laser
50     ang_sp=Robot.odom_data.angular.z
51     lin_sp=Robot.odom_data.linear.x
52     s_temp=np.append(s_temp,[ang_sp,lin_sp])
53     return s_temp
54 def append_hist(a,value):
55     with open(a, 'a') as f:
56         f.write("%s\n" % value)
57 def test_model():
58     global epsilon
59     epsilon=0.1
60     while not rospy.is_shutdown():
61         s=get_state()
62         a=choose_action(s)
63         perform_action(a)
64         try:
65             rospy.sleep(0.1)
66         except:
67             pass
68 if __name__== "__main__":
69     main()

```

```

1 #!/usr/bin/env python2
2 # -*- coding: utf-8 -*-
3 """
4 Created\underline{on} \underline{Sat} \underline{Nov} \underline{22} \underline{14:37:13} \underline{2019}
5
6 @author: \underline{ganzobat}
7 """
8
9 from rosserial_arduino.msg import Adc # Datatype to read IR sensor
    value from arduino

```

```

10 from sensor_msgs.msg import LaserScan # Dateatype to publish IR
11   sensor ranges
12 from collections import deque # will use it to calculate moving
13   averages
14 import rospy
15 import numpy as np
16
17
18 ir_ranger_adc_topic='/adc'
19 real_range='/scan'
20
21
22 rospy.init_node('range_estimator')
23 n_readings=5
24 laser_frequency=100
25 r=rospy.Rate(laser_frequency)
26 moving_avg=10
27 memory = deque(maxlen=moving_avg)
28 b=5127
29 m=-1.007 # range=b*adc^m
30
31
32 class IRRange_Laser():
33     def __init__(self):
34         self.range=rospy.Publisher(real_range, LaserScan,
35             queue_size=2)
36         self.scan = LaserScan()
37         self.scan.header.frame_id = 'laser_frame'
38         self.scan.angle_min = -1.57
39         self.scan.angle_max = 1.57
40         self.scan.angle_increment = (self.scan.angle_max-self.scan.
41             angle_min) / n_readings
42         self.scan.time_increment = (1.0 / laser_frequency) / (
43             n_readings)
44         self.scan.range_min = 0.0
45         self.scan.range_max = 0.6
46     def publish_ranges(self,ranges):
47         self.scan.header.stamp = rospy.Time.now()
48         self.scan.ranges=ranges
49         self.range.publish(self.scan)
50
51
52 class ADC():
53     def __init__(self):
54         self.ir_adc=rospy.Subscriber(ir_ranger_adc_topic, Adc,
55             callback=self.adc_callback)
56         self.values=[]
57     def adc_callback(self,adc):
58         self.q_temp=[]
59         self.a=adc
60         self.q_temp.append(self.a.adc0)

```

```
51         self.q_temp.append(self.a.adc1)
52         self.q_temp.append(self.a.adc2)
53         self.q_temp.append(self.a.adc3)
54         self.q_temp.append(self.a.adc4)
55         self.values=self.q_temp
56
57 adc=ADC()
58 ranges=IRRange_Laser()
59
60 def main():
61     while not rospy.is_shutdown():
62         try:
63             #adc_values=adc.values
64             memory.append(adc.values)
65             a=np.array(memory)
66             if len(a)<8:
67                 a=b*pow(np.mean(a, axis=0), m)
68             else:
69                 q=np.sort(a, axis=0)
70                 a=0.01*b*pow(np.mean(q[2:8]+1e-5, axis=0), m)
71
72             for i in range(len(a)):
73                 if a[i]>0.6:
74                     a[i]=0.6
75
76             ranges.publish_ranges(a)
77             r.sleep()
78         except:
79             pass
80 if __name__=="__main__":
81     main()
```

# Ном зу́й

- [1] Stanford Artificial Intelligence Laboratory et al. *Robotic Operating System*. Version ROS Melodic Morenia. May 23, 2018. URL: <https://www.ros.org>.
- [2] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998. URL: <http://www.cs.ualberta.ca/~sutton/book/the-book.html>.
- [3] Christopher Watkins and Peter Dayan. “Technical Note: Q-Learning”. In: *Machine Learning* 8 (May 1992), pp. 279–292. DOI: 10.1007/BF00992698.
- [4] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. <http://www.deeplearningbook.org>. Cambridge, MA, USA: MIT Press, 2016.
- [5] Volodymyr Mnih et al. “Playing atari with deep reinforcement learning”. In: *arXiv preprint arXiv:1312.5602* (2013). URL: <https://arxiv.org/pdf/1312.5602.pdf>.
- [6] Hado van Hasselt, Arthur Guez, and David Silver. “Deep Reinforcement Learning with Double Q-learning”. In: (2015). cite arxiv:1509.06461Comment: AAAI 2016. URL: <http://arxiv.org/abs/1509.06461>.
- [7] L. Joseph. *ROS Robotics Projects 2nd Edition*. Packt Publishing, 2017. ISBN: 9781783554720. URL: <https://books.google.mn/books?id=rLkrDwAAQBAJ>.