

프로젝트 보고서

〈ROS 및 CoppeliaSim 연구〉

프로젝트
프로젝트 기간
이 름

2021학년도 하계 학부연구생
2021.07.01 ~ 2021.08.31
신건희

1. ROS Course 수강

1. Abstract	2
2. ROS 용어	2
3. Topic/Publisher/Subscriber	2~5
4. Service/Server/Client	5~10

2. Pioneer p3dx Potential Field Script in Robot Simulator 'CoppeliaSim'

1. Abstract	11
2. Pioneer p3dx script	11~15
3. Hokuyo Sensor Script	15~16
4. Parameter 조작	16~23
5. Conclusion	23~24

1. ROS Course 수강

1. Abstract

2021학년도 하계 학부연구생 프로그램에서 가장 먼저 개인 노트북에 Ubuntu와 ROS를 설치하고 ROS 강의를 수강했다. Dual Booting으로 Ubuntu를 설치하는 과정에서 노트북이 포맷되고 ROS 설정 오류가 발생하는 등 많은 시행착오가 있었다. ROS 강의는 표윤석 박사가 제공하는 유튜브 강의를 수강했다. ROS 강의와 함께 지도 교수님의 지시에 따라 Robot Simulator CoppeliaSim에 대한 공부를 병행했다.

2. ROS 용어

- node: 최소 단위의 실행 가능한 프로세스, 각 노드는 메시지 통신으로 데이터를 주고받는다.
- Package: 하나 이상의 노드, 노드 실행을 위한 정보 등을 묶어 놓은 것.
- Message: 메시지를 통해 노드 간의 데이터를 주고받게 된다. Integer, Floating, Point, Boolean 변수 형태
- Topic, Publisher, Subscriber: Topic에 대해 1:1의 통신도 가능하며 목적에 따라 1: N, N: N, N:1 통신도 가능
- Service, Service Server, Service Client
- Action, Action Server, Action Client

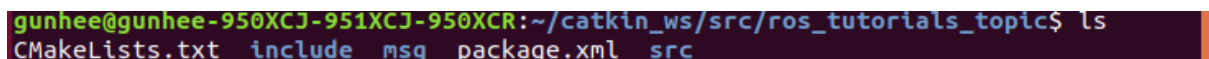
3. Topic/Publisher/Subscriber

1. 패키지 생성

- ROS에서 단 방향 통신을 할 때 'TOPIC' 메시지 통신을 한다. 이때 송신 측을 'Publisher', 수신 측을 'Subscriber'라 한다.

```
$cd~/catkin_ws/src
```

```
$ catkin_create_pkg ros_tutorials_topic message_generation std_msgs roscpp
```



```
gunhee@gunhee-950XCJ-951XCJ-950XCR:~/catkin_ws/src/ros_tutorials_topic$ ls
CMakeLists.txt  include  msg      package.xml  src
```

<FIG.1 생성한 패키지 내의 list 확인>

2. 패키지 설정 파일(package.xml)수정

- ROS의 필수 설정 파일 중 하나인 package.xml은 패키지 정보를 담은 XML파일로 패키지 이름, 저작자, 라이선스, 의존성 패키지 등이 기술되어 있다.

```

<?xml version="1.0"?>
<package format="2">
  <name>ros_tutorials_topic</name>
  <version>0.0.0</version>
  <description>The ros_tutorials_topic package</description>

  <!-- One maintainer tag required, multiple allowed, one person per tag -->
  <!-- Example:  -->
  <!-- <maintainer email="jane.doe@example.com">Jane Doe</maintainer> -->
  <maintainer email="gunhee@todo.todo">gunhee</maintainer>

  <!-- One license tag required, multiple allowed, one license per tag -->
  <!-- Commonly used license strings: -->
  <!--   BSD, MIT, Boost Software License, GPLv2, GPLv3, LGPLv2.1, LGPLv3 -->
  <license>TODO</license>

  <buildtool_depend>catkin</buildtool_depend>
  <depend>roscpp</depend>
  <depend>std_msgs</depend>
  <depend>message_generation</depend>
  <export></export>
</package>

```

<FIG.2 package.xml>

3. Build 설정 파일(CMakeLists.txt) 수정

1. catkin_make시 필요한 의존성 패키지로 message_generation, std_msgs, roscpp가 있으며 해당 패키지들이 존재하지 않으면 Build 도중 에러가 발생한다.
2. 메시지 선언: MsgTutorial.msg
3. 의존성 메시지 설정, std_msgs가 설치되어 있지 않다면 빌드 도중에 에러가 발생한다.

```

cmake_minimum_required(VERSION 3.0.2)
project(ros_tutorials_topic)

① find_package(catkin REQUIRED COMPONENTS
  message_generation
  roscpp
  std_msgs
)

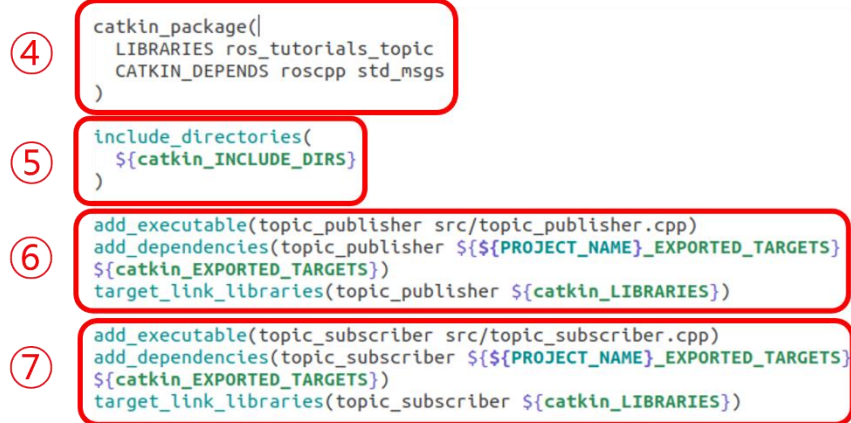
② add_message_files(FILES MsgTutorial.msg)

③ generate_messages(DEPENDENCIES std_msgs)

```

<FIG.3 CMakeList.txt>

4. Catkin 패키지 옵션으로 라이브러리, Catkin_make 의존성 패키지를 기술한다.
5. Directories 설정하는 구간이다.
6. Topic_Publisher 노드에 대한 빌드 옵션, 실행 파일, 타겟 링크 라이브러리, 추가 의존성 패키지 등을 설정한다.
7. 6번과 같이 Topic_subscriber 노드에 대한 빌드 옵션을 설정한다.



<FIG.4 CMakeList.txt>

4. Message File 작성

- 앞서 CMakeList.txt 파일에서 다음과 같은 옵션을 넣었다.
add_message_files(FILES MsgTutorial.msg)
- Message File을 생성한 뒤 time stamp, int32 data를 입력한다. time과 int32는 메시지 형식을 의미하며 stamp와 데이터는 메시지 이름을 의미한다

\$ mkdir msg

\$ cd msg

\$ gedit MsgTutorial.msg

5. Publisher 노드 작성

- 앞서 CMakeList.txt 파일에서 다음과 같은 옵션을 넣었다.
add_executable(topic_publisher src/topic_publisher.cpp)
- gedit 명령어를 통해 topic_publisher.cpp를 사진과 같이 수정한다.

\$. ~/catkin_ws/devel/setup.bash

\$ roscd ros_tutorials_topic/src

\$ gedit topic_publisher.cpp

```

#include "ros/ros.h"
#include "ros_tutorials_topic/MsgTutorial.h"

int main(int argc, char **argv)
{
  ros::init(argc, argv, "topic_publisher");
  ros::NodeHandle nh;
  ros::Publisher ros_tutorial_pub = nh.advertise<ros_tutorials_topic::MsgTutorial>("ros_tutorial_msg",
    100);

  ros::Rate loop_rate(10);
  ros_tutorials_topic::MsgTutorial msg;
  int count = 0;

  while (ros::ok())
  {
    msg.stamp = ros::Time::now();
    msg.data = count;

    ROS_INFO("send msg= %d", msg.stamp.sec);
    ROS_INFO("send msg= %d", msg.stamp.nsec);
    ROS_INFO("send msg= %d", msg.data);

    ros_tutorial_pub.publish(msg);
    loop_rate.sleep();

    ++count;
  }
  return 0;
}

```

<FIG.5 topic_publisher.cpp>

6. Subscriber 노드 작성

- 앞서 CMakeList.txt 파일에서 다음과 같은 옵션을 넣었다.
add_executable(topic_subscriber src/topic_subscriber.cpp)
- gedit 명령어를 통해 topic_subscriber.cpp를 사진과 같이 수정한다.

```
$ . ~/catkin_ws/devel/setup.bash
```

```
$ roscd ros_tutorials_topic/src
```

```
$ gedit topic_subscriber.cpp
```

```
#include "ros/ros.h"
#include "ros_tutorials_topic/MsgTutorial.h"

void msgCallback(const ros_tutorials_topic::MsgTutorial::ConstPtr& msg)
{
    ROS_INFO("received msg= %d", msg->stamp.sec);
    ROS_INFO("received msg= %d", msg->stamp.nsec);
    ROS_INFO("receivedmsg= %d", msg->data);
}

int main(int argc, char **argv)
{
    ros::init(argc, argv, "topic_subscriber");
    ros::NodeHandle nh;

    ros::Subscriber ros_tutorial_sub= nh.subscribe("ros_tutorial_msg", 100, msgCallback);

    ros::spin();

    return 0;
}
```

<FIG.6 topic_subscriber.cpp>

7. 패키지 빌드

- catkin_make를 통해 패키지를 빌드한다.

```
$ cd~/catkin_ws
```

```
$ catkin_make
```

```

gunhee@gunhee-950C91-951C91-950C91:~/catkin_ws$ /usr/src/ros_tutorials/Topics cd ~/catkin_ws
gunhee@gunhee-950C91-951C91-950C91:~/catkin_ws$ catkin_make

base path: /home/gunhee/catkin_ws
Source space: /home/gunhee/catkin_ws/src
Build space: /home/gunhee/catkin_ws/build
Devel space: /home/gunhee/catkin_ws/devel
Install space: /home/gunhee/catkin_ws/install

Running command: "make check_build_system" in "/home/gunhee/catkin_ws/build"
make
--
Using CATKIN_DEVEL_PREFIX: /home/gunhee/catkin_ws/devel
Using CMAKE_PREFIX_PATH: /usr/src/melodic
This workspace overlays: /opt/ros/melodic
Found PythonInterp: /usr/bin/python2 (found suitable version "2.7.17", minimum required is "2")
Using Python setup file: /usr/bin/python2
Using Debian Python package layout
Using Catkin tool to generate command line dependencies for catkin packages
Using CMake, cmake/3.16.3
Call enable_testing()
Using CATKIN_TEST_RESULTS_DIR: /home/gunhee/catkin_ws/build/test_results
Found gtest sources under: /usr/src/googletest; gtests will be built
Found gmock sources under: /usr/src/googletest; gmock will be built
CMake Deprecation Warning at /usr/src/googletest/CMakeLists.txt:1 (cmake_minimum_required):
Compatibility with CMake <2.8.12 will be removed from a future version of
CMake.

Update the VERSION argument <min value> or use a ...>max suffix to tell
CMake that the project does not need compatibility with older versions.

CMake Deprecation Warning at /usr/src/googletest/gmock/CMakeLists.txt:4 (cmake_minimum_required):
Compatibility with CMake <2.8.12 will be removed from a future version of
CMake.

Update the VERSION argument <min value> or use a ...>max suffix to tell
CMake that the project does not need compatibility with older versions.

CMake Deprecation Warning at /usr/src/googletest/gmock/CMakeLists.txt:40 (cmake_minimum_required):
Compatibility with CMake <2.8.12 will be removed from a future version of
CMake.

Update the VERSION argument <min value> or use a ...>max suffix to tell
CMake that the project does not need compatibility with older versions.

CMake Deprecation Warning at /usr/src/googletest/gmock/CMakeLists.txt:40 (cmake_minimum_required):
Compatibility with CMake <2.8.12 will be removed from a future version of
CMake.

Update the VERSION argument <min value> or use a ...>max suffix to tell
CMake that the project does not need compatibility with older versions.

Found PythonInterp: /usr/bin/python2 (found version "2.7.17")
Using Python nose2ests: /usr/bin/nose2ests-2.7
catkin -b -D
BUILD_SHARED_LIBS is on
BUILD_SHARED_LIBS is on
gunhee@gunhee-950C91-951C91-950C91:~/catkin_ws$

```

<FIG.7 catkin_make 과정>

8. Publisher 실행

- ROS 노드 실행 명령어인 rosrn을 이용하여, ros_tutorials_topic 실행.
- 생성한 패키지에서 topic_publisher 노드를 구동하는 명령어

\$ rosrn ros_tutorials_topic topic_publisher

```
gunhee@gunhee-950XCJ-951XCJ-950XCR:~$ . ~/catkin_ws/devel/setup.bash
gunhee@gunhee-950XCJ-951XCJ-950XCR:~$ rosrn ros_tutorials_topic topic_publisher
[ INFO] [1628662076.921279134]: send msg= 1628662076
[ INFO] [1628662076.921905086]: send msg= 921247633
[ INFO] [1628662076.921928313]: send msg= 0
[ INFO] [1628662077.021433851]: send msg= 1628662077
[ INFO] [1628662077.021537979]: send msg= 21384507
[ INFO] [1628662077.021583641]: send msg= 1
[ INFO] [1628662077.121454560]: send msg= 1628662077
[ INFO] [1628662077.121549879]: send msg= 121389017
[ INFO] [1628662077.121601924]: send msg= 2
[ INFO] [1628662077.221423741]: send msg= 1628662077
[ INFO] [1628662077.221505830]: send msg= 221376839
[ INFO] [1628662077.221550568]: send msg= 3
[ INFO] [1628662077.321492443]: send msg= 1628662077
[ INFO] [1628662077.321585745]: send msg= 321424888
[ INFO] [1628662077.321620041]: send msg= 4
[ INFO] [1628662077.421455289]: send msg= 1628662077
[ INFO] [1628662077.421552919]: send msg= 421393192
```

<FIG.8 topic_publisher 노드 구동>

9. Subscriber 실행

- ROS 노드 실행 명령어인 rosrn을 이용하여, ros_tutorial_topic 실행.
- 생성한 패키지에서 topic_subscriber 노드를 구동하는 명령어

\$ rosrn ros_tutorials_topic topic_subscriber

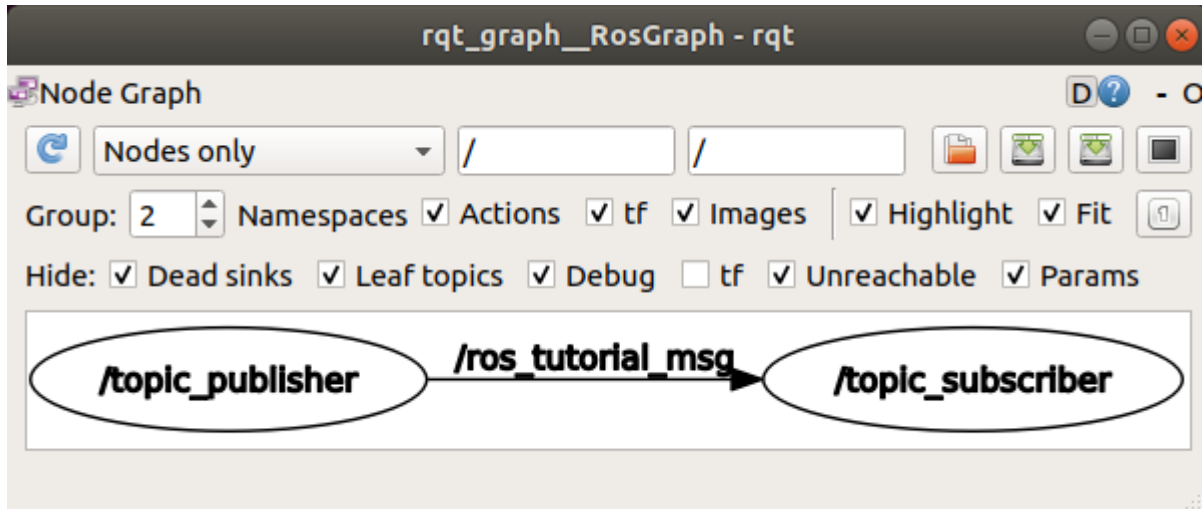
```
gunhee@gunhee-950XCJ-951XCJ-950XCR:~$ rosrn ros_tutorials_topic topic_subscriber
[ INFO] [1628662293.822222013]: recieve msg= 1628662293
[ INFO] [1628662293.825125183]: recieve msg= 821380909
[ INFO] [1628662293.825185353]: recievemsg= 2169
[ INFO] [1628662293.922078109]: recieve msg= 1628662293
[ INFO] [1628662293.922161083]: recieve msg= 921385932
[ INFO] [1628662293.922189440]: recievemsg= 2170
[ INFO] [1628662294.022062455]: recieve msg= 1628662294
[ INFO] [1628662294.022157863]: recieve msg= 21385028
[ INFO] [1628662294.022198197]: recievemsg= 2171
[ INFO] [1628662294.122032962]: recieve msg= 1628662294
[ INFO] [1628662294.122117053]: recieve msg= 121381619
[ INFO] [1628662294.122169771]: recievemsg= 2172
[ INFO] [1628662294.222272235]: recieve msg= 1628662294
[ INFO] [1628662294.222364854]: recieve msg= 221375592
[ INFO] [1628662294.222415822]: recievemsg= 2173
[ INFO] [1628662294.322119407]: recieve msg= 1628662294
[ INFO] [1628662294.322204806]: recieve msg= 321382608
```

<FIG.9 topic_subscriber 노드 구동>

10. rosrn을 통해 실행된 노드 통신 상태 확인

- Rqt_graph를 통해 통신 상태를 확인할 수 있다.

\$rqt_graph



<FIG.10 rqt_graph>

4. Service/Server/Client

1. 패키지 생성

- ROS에서 양방향 통신을 할 때 Service 메시지 통신을 사용한다. 이때 요청이 있을 때만 응답하는 Service Server와 응답 받는 Service Client가 있다.

```
$ cd~/catkin_ws/src
```

```
$ catkin_create_pkg ros_tutorials_service message_generation std_msgs roscpp
```

```
gunhee@gunhee-950XCJ-951XCJ-950XCR:~$ cd ~/catkin_ws/src
gunhee@gunhee-950XCJ-951XCJ-950XCR:~/catkin_ws/src$ catkin_create_pkg ros_tutorials_service message_generation std_msgs roscpp
WARNING: Packages with messages or services should depend on both message_generation and message_runtime
Created file ros_tutorials_service/CMakeLists.txt
Created file ros_tutorials_service/package.xml
Created folder ros_tutorials_service/include/ros_tutorials_service
Created folder ros_tutorials_service/src
Successfully created files in /home/gunhee/catkin_ws/src/ros_tutorials_service.
Please adjust the values in package.xml.
gunhee@gunhee-950XCJ-951XCJ-950XCR:~/catkin_ws/src$ cd ros_tutorials_service
gunhee@gunhee-950XCJ-951XCJ-950XCR:~/catkin_ws/src/ros_tutorials_service$ ls
CMakeLists.txt include package.xml src
```

<FIG.11 Service Tutorial Package 생성>

2. Build 설정 파일(CMakeLists.txt) 수정

1. Catkin_make시 필요한 의존성 패키지로 message_generation, std_msgs, roscpp가 있으며 해당 패키지들이 존재하지 않으면 Build 도중 에러가 발생한다.
2. Service File 추가: SrvTutorial.srv
3. 의존성 메시지 설정, std_msgs가 설치되어 있지 않다면 빌드 도중에 에러가 발생한다.

```

cmake_minimum_required(VERSION 3.0.2)
project(ros_tutorials_service)

① find_package(catkin REQUIRED COMPONENTS
  message_generation
  roscpp
  std_msgs
)

② add_service_files(
  FILES SrvTutorial.srv
)

③ generate_messages(
  DEPENDENCIES
  std_msgs
)

```

<FIG.12 CMakeList.txt>

4. Catkin 패키지 옵션으로 라이브러리 catkin_make 의존성, 시스템 의존 패키지를 기술한다.
5. Directories 설정하는 구간이다.
6. Service Server 노드에 대한 빌드 옵션, 실행 파일, 타겟 링크 라이브러리, 추가 의존성 패키지 등을 설정한다.
7. 6번과 같이 Service Client 노드에 대한 빌드 옵션을 설정한다.

```

④ catkin_package(
  LIBRARIES ros_tutorials_service
  CATKIN_DEPENDS roscpp std_msgs
)

⑤ include_directories(
  ${catkin_INCLUDE_DIRS}
)

⑥ add_executable(service_server src/service_server.cpp)
  add_dependencies(service_server ${${PROJECT_NAME}_EXPORTED_TARGETS}
    ${catkin_EXPORTED_TARGETS})
  target_link_libraries(service_server ${catkin_LIBRARIES})

⑦ add_executable(service_client src/service_client.cpp)
  add_dependencies(service_client ${${PROJECT_NAME}_EXPORTED_TARGETS}
    ${catkin_EXPORTED_TARGETS})
  target_link_libraries(service_client ${catkin_LIBRARIES})

```

<FIG.13 CMakeList.txt>

3. Message File 작성

- 앞서 CMakeList.txt 파일에서 다음과 같은 옵션을 넣었다.
add_message_files(FILES SrvTutorial.srv)
- Message File을 생성한 뒤 int64 a, int64 b, ---, int64 result를 입력한다. Int64는 메시지 형식, a,b는 서비스 요청, result는 서비스 응답, ---는 요청과 응답을 구분하는 구분자를 의미한다

\$ mkdir msg

\$ cd msg

\$ gedit SrvTutorial.srv

4. Publisher 노트 작성

- 앞서 CMakeLixt.txt 파일에서 다음과 같은 옵션을 넣었다.
add_executable(service_server src/service_server.cpp)
- gedit 명령어를 통해 topic_publisher.cpp를 사진과 같이 수정한다.

```
$ . ~/catkin_ws/devel/setup.bash
```

```
$ roscd ros_tutorials_service/src
```

```
$ gedit topic_service_server.cpp
```

```
#include "ros/ros.h"
#include "ros_tutorials_service/SrvTutorial.h"

bool calculation(ros_tutorials_service::SrvTutorial::Request &req,
ros_tutorials_service::SrvTutorial::Response &res)
{ res.result= req.a+ req.b;
  ROS_INFO("request: x=%ld, y=%ld", (long int)req.a, (long int)req.b);
  ROS_INFO("sending back response: %ld", (long int)res.result);
  return true;
}

int main(int argc, char **argv)
{
  ros::init(argc, argv, "service_server");
  ros::NodeHandle nh;
  ros::ServiceServer ros_tutorials_service_server= nh.advertiseService("ros_tutorial_srv", calculation);
  ROS_INFO("ready srv server!");
  ros::spin();
  return 0;
}
```

<FIG.14 service_server.cpp>

5. Subscriber 노트 작성

- 앞서 CMakeLixt.txt 파일에서 다음과 같은 옵션을 넣었다.
add_executable(service_client src/service_client.cpp)
- gedit 명령어를 통해 service_client.cpp 를 사진과 같이 수정한다.

```
$ . ~/catkin_ws/devel/setup.bash
```

```
$ roscd ros_tutorials_service /src
```

```
$ gedit topic_service_client.cpp
```

```
#include "ros/ros.h"
#include "ros_tutorials_service/SrvTutorial.h"
#include <cstdlib>

int main(int argc, char **argv)
{ros::init(argc, argv, "service_client");
  if (argc!= 3)
  {ROS_INFO("cmd: rosrn ros_tutorials_service service_client arg0 arg1");
   ROS_INFO("arg0: double number, arg1: double number");return 1;}
  ros::NodeHandle nh;

  ros::ServiceClient ros_tutorials_service_client= nh.serviceClient<ros_tutorials_service::SrvTutorial>("ros_tutorial_srv");

  ros_tutorials_service::SrvTutorial srv;
  srv.request.a= atoll(argv[1]);
  srv.request.b= atoll(argv[2]);
  if (ros_tutorials_service_client.call(srv))
  {
    ROS_INFO("send srv, srv.Request.a and b: %ld, %ld", (long int)srv.request.a, (long int)srv.request.b);
    ROS_INFO("receive srv, srv.Response.result: %ld", (long int)srv.response.result);
  }
  else{ROS_ERROR("Failed to call service ros_tutorial_srv");return 1;}return 0;}
}
```

<FIG.15 service_client.cpp>

6. 패키지 빌드

- catkin_make를 통해 패키지를 빌드한다.

```
$ catkin_make
```

```

gunhee@gunhee-950XC-951XC-950XC:~/catkin_ws/src/ros_tutorials$ cd ~/catkin_ws
$ pwd
/home/gunhee/catkin_ws
$ echo path: /home/gunhee/catkin_ws
path: /home/gunhee/catkin_ws
$ echo source space: /home/gunhee/catkin_ws/src
source space: /home/gunhee/catkin_ws/src
$ echo build space: /home/gunhee/catkin_ws/build
build space: /home/gunhee/catkin_ws/build
$ echo devel space: /home/gunhee/catkin_ws/devel
devel space: /home/gunhee/catkin_ws/devel
$ echo install space: /home/gunhee/catkin_ws/install
install space: /home/gunhee/catkin_ws/install
$ echo
$ echo ## Running command: "make cmake_check_build_system" in "/home/gunhee/catkin_ws/build"
$ make
-- Using CATKIN_DEVEL_PREFIX: /home/gunhee/catkin_ws/devel
-- Using CMAKE_PREFIX_PATH: /opt/ros/melodic
-- This workspace overlays: /opt/ros/melodic
-- Found PythonInterp: /usr/bin/python2 (found suitable version "2.7.17", minimum required is "2")
-- Using Python setup file /usr/bin/python2
-- Using Debian Python package layout
-- Using Exec: /usr/bin/epm
-- Using CATKIN_ENABLE_TESTING: ON
-- Call enable_testing()
-- Using CATKIN_TEST_RESULTS_DIR: /home/gunhee/catkin_ws/build/test_results
-- Found gtest sources under /usr/src/googletest: gtests will be built
-- Found gmock sources under /usr/src/googletest: gmock will be built
CMake Deprecation Warning at /usr/src/googletest/CMakeLists.txt:1 (cmake_minimum_required):
Compatibility with CMake <2.8.12 will be removed from a future version of
CMake.

Update the VERSION argument <min> value or use a ...>max> suffix to tell
CMake that the project does not need compatibility with older versions.

CMake Deprecation Warning at /usr/src/googletest/googletest/CMakeLists.txt:4 (cmake_minimum_required):
Compatibility with CMake <2.8.12 will be removed from a future version of
CMake.

Update the VERSION argument <min> value or use a ...>max> suffix to tell
CMake that the project does not need compatibility with older versions.

CMake Deprecation Warning at /usr/src/googletest/googletest/CMakeLists.txt:48 (cmake_minimum_required):
Compatibility with CMake <2.8.12 will be removed from a future version of
CMake.

Update the VERSION argument <min> value or use a ...>max> suffix to tell
CMake that the project does not need compatibility with older versions.

-- Found PythonInterp: /usr/bin/python2 (found version "2.7.17")
-- Using PythonInterp: /usr/bin/python2
-- catkin 0.7.29
-- BUILD_SHARED_LIBS is on
-- BUILD_SHARED_LIBS is on
$ echo
$ echo ## Processing catkin package: 'ros_tutorials'
$ echo
$ echo == add_subdirectory(/ros_tutorials) ==
$ echo
$ echo Using these message generators: genmsg;gennodejs;genpy
$ echo
$ echo ros_tutorials: 1 messages, 0 services
$ echo
$ echo Configuring done
$ echo
$ echo -- Build files have been written to: /home/gunhee/catkin_ws/build
$ echo
$ echo ## Running command: "make -j8 -l8" in "/home/gunhee/catkin_ws/build"
$ make
[ 0%] Build target roscpp_msgs_generate_messages_nodejs
[ 0%] Build target roscpp_generate_messages_py
[ 0%] Build target roscpp_msgs_generate_messages_py
[ 0%] Build target roscpp_msgs_generate_messages_egl
[ 0%] Build target roscpp_msgs_generate_messages_lisp
[ 0%] Build target std_msgs_generate_messages_egl
[ 0%] Build target roscpp_msgs_generate_messages_cpp
[ 0%] Build target std_msgs_generate_messages_egl
[ 0%] Build target geometry_msgs_generate_messages_lisp
[ 0%] Build target std_msgs_generate_messages_cpp
[ 0%] Build target geometry_msgs_generate_messages_cpp
[ 0%] Build target roscpp_generate_messages_cpp
[ 0%] Build target std_msgs_generate_messages_lisp
[ 0%] Build target geometry_msgs_generate_messages_nodejs
[ 0%] Build target roscpp_generate_messages_lisp
[ 0%] Build target geometry_msgs_generate_messages_py
[ 0%] Build target std_msgs_generate_messages_py
[ 0%] Build target roscpp_generate_messages_nodejs
[ 0%] Build target geometry_msgs_generate_messages_cpp
[ 0%] Build target roscpp_generate_messages_egl
[ 0%] Build target roscpp_generate_messages_egl
Consolidate compiler generated dependencies of target rosbabelnode
[ 0%] Build target _ros_tutorials_topic_generate_messages_check_deps_MyTutorial
[ 15%] Build target _ros_tutorials_topic_generate_messages_py
[ 30%] Build target _ros_tutorials_topic_generate_messages_lisp
[ 40%] Build target _ros_tutorials_topic_generate_messages_egl
[ 40%] Build target _ros_tutorials_topic_generate_messages_cpp
[ 53%] Build target _ros_tutorials_topic_generate_messages_nodejs
[ 60%] Build target rosbabelnode
Consolidate compiler generated dependencies of target topic_publisher
Consolidate compiler generated dependencies of target topic_subscriber
[ 60%] Build target _ros_tutorials_topic_generate_messages
[ 100%] Build target _catkin_ws
$ echo
$ echo gunhee@gunhee-950XC-951XC-950XC:~/catkin_ws$

```

<FIG.16 catkin_make 과정>

7. Service Server 실행

- Service Server는 요청이 있기 전까지는 응답이 없다. 그러므로 다음 명령어를 실행하면 Service Server는 서비스 요청을 기다린다.

```
$roslaunch rostutorials_service service_server
```

```
gunhee@gunhee-950XCJ-951XCJ-950XCR:~$ . ~/catkin_ws/devel/setup.bash
gunhee@gunhee-950XCJ-951XCJ-950XCR:~$ rosrn ros_tutorials_service service_ser
[ INFO] [1628664217.884896196]: ready srv server!
[ INFO] [1628664285.543648324]: request: x=2, y=3
[ INFO] [1628664285.543675035]: sending back response: 5
[ INFO] [1628664344.684271619]: request: x=10, y=5
[ INFO] [1628664344.684386422]: sending back response: 15
[ INFO] [1628664350.636802125]: request: x=15, y=5
[ INFO] [1628664350.636892653]: sending back response: 20
```

<FIG.17 rosrun service_server>

8. Service Client 실행

- Service Server를 실행했으면 이어서 다음 명령어로 Service Client를 실행한다.

```
$roslaunch ros_tutorials_service service_client 2 3
```

- Service Client에서 2,3을 요청하고 Server는 두 상수의 합인 5를 Return했다.

```
gunhee@gunhee-950XCJ-951XCJ-950XCR:~$ . ~/catkin_ws/devel/setup.bash
gunhee@gunhee-950XCJ-951XCJ-950XCR:~$ rosrn ros_tutorials_service service_client
t 2 3
[ INFO ] [1628664285.543837317]: send srv, srv.Request.a and b: 2, 3
[ INFO ] [1628664285.544833789]: receive srv, srv.Response.result: 5
```

<FIG.18 rosrn service_client>

2. Pioneer p3dx Potential Field Script in CoppeliaSim

1. Abstract

Leopoldo Armsesto의 CoppeliaSim 강의를 수강하면서 Pioneer p3dx Potential Field Script를 분석했다. 인력과 척력을 이용하여 장애물을 피해 목표 지점에 도달하는 Script로 코드 중간에 To Do Section이 있었다. 개념을 익히고 팀원들과 함께 노력하여 코드를 채워 넣으려 했지만 한계에 부딪혔고 유료로 솔루션을 구매했다. 솔루션을 바탕으로 변수를 변경하며 최적의 값을 찾으려고 노력했다.

2. Pioneer p3dx script

1. Function sysCall_init()

- `sim.getObjectHandle('Object Name')` : scene hierarchy상의 객체의 이름을 적어준다. 이를 통해 객체를 변수로 지정할 수 있다. return 값은 handle of object(objectHandle)를 반환한다.
- `sim.getObjectPosition(objectHandle, -1)` : 객체의 위치를 검색해준다. `sim.getObjectHandle`를 통해 반환된 objectHandle 값을 입력하고, -1을 입력하여 절대좌표의 위치를 기준으로 객체의 위치를 검색한다.
- 각 변수들에 대한 값을 설정해준다.

```
function sysCall_init()
    left_wheel=sim.getObjectHandle('Pioneer_p3dx_leftMotor')
    right_wheel=sim.getObjectHandle('Pioneer_p3dx_rightMotor')
    robot_pose=sim.getObjectHandle('robot_pose')
    goal=sim.getObjectHandle('goal')
    goalP=sim.getObjectPosition(goal, -1)

    wheel_radius=0.195/2
    b=0.1655
    vref=0.35
    e=0.24
    epsilon_q=1
    epsilon_c=2
    epsilon_r=0.005
    Qsafe=0.1
    Qinf=1
end
```

2. Function updateRobotPose()

- Pioneer p3dx의 최근 위치를 알려준다.
- `sim.getOrientation(objectHandle, -1)` : 객체의 방향을 알려준다. -1을 입력하여 절대좌표를 기준으로 객체의 방향을 검색한다.

- position[1]은 X좌표, position[2]은 Y좌표, orientation[3]은 방향을 나타낸다. 이 값들을 pose 라는 변수로 지정한다.

```
function updateRobotPose()
    local pose
    position=sim.getObjectPosition(robot_pose, -1)
    orientation=sim.getObjectOrientation(robot_pose, -1)
    pose = {position[1], position[2], orientation[3]}
    return pose
end
```

3. Function getLaserPoints()

- sim.callScriptFunction('function', sim.scripttype_script type) : 다른 객체의 script를 가져오는 함수이다. 가져올 function을 적고 sim.scripttype뒤에 script의 type을 적어준다. 위의 getMeasuredData@fastHokuyo을 통해 Laser의 x, y, z point 값들을 불러와 준다.
- Laserpoint는 장애물을 뜻하며 laserPts 배열에 저장된다.

```
function getLaserPoints()
    local laserScan
    local laserPts={}
    local j = 1
    laserScan=sim.callScriptFunction('getMeasuredData@fastHokuyo',sim.scrip
    ttype_childscript)
    for i=1, #laserScan,3 do
        laserPts[j]={laserScan[i], laserScan[i+1]}
        j=j+1
    end
    return laserPts
end
```

4. Function sysCall_sensing()

- getLaserpoints()을 통해 반환된 값을 변수 laserPoints로 지정해준다.

```
function sysCall_sensing()
    laserPoints=getLaserPoints()
end
```

5. Function attractiveForce()

- $d_g = \sqrt{(x-x_g)^2 + (y-y_g)^2}$ 의 식을 이용하여 Pioneer p3dx과 목표 지점 사이의 거리

를 나타낸다.

$$\nabla U_{attractive}(q, q_g) = \begin{cases} [\epsilon_q d_g (x - x_g) & \epsilon_q d_g (y - y_g)]^T & \text{if } d_g < Q_g \\ \left[\frac{Q_g \epsilon_q (x - x_g)}{d_g} & \frac{Q_g \epsilon_q (y - y_g)}{d_g} \right]^T & \text{if } d_g \geq Q_g \end{cases}$$

- 위의 수식을 바탕으로 Attractive Force에 대한 코드를 작성한다.
- d_g 는 코드에서 d 로 표현됐으며 Q_g 는 QC 로 표현됐다.

```
function attractiveForce(pose,goalP,epsilon_q,epsilon_c)
    local d,deltaX,deltaY,aForce,QC
    QC=epsilon_q/epsilon_c
    deltaX=pose[1]-goalP[1]
    deltaY=pose[2]-goalP[2]
    d=math.sqrt(deltaX^2+deltaY^2)
    if d<QC then
        aForce={-epsilon_q*deltaX,-epsilon_q*deltaY}
    else
        aForce={-epsilon_c*deltaX/d,-epsilon_c*deltaY/d}
    end
    return aForce
end
```

6. Function repulsiveForce()

- 센서를 통해 알아낸 장애물의 좌표를 통해 Pioneer p3dx과 장애물 사이의 거리

$d_i = \sqrt{l_{x,i}^2 + l_{y,i}^2}$ 의 식을 이용하여 알아낼 수 있다.

$$\nabla U_{repulsive}(l) = \sum_i \begin{cases} \begin{bmatrix} \epsilon_\gamma l_{x,i}/d_i^3 \\ \epsilon_\gamma l_{y,i}/d_i^3 \end{bmatrix} & \text{if } Q_{min} \leq d_i \leq Q_{inf} \\ \begin{bmatrix} 0 \\ 0 \end{bmatrix} & \text{otherwise} \end{cases}$$

- 위의 수식을 바탕으로 Repulsive Force에 대한 코드를 작성한다.
- Q_{min} 는 코드에서 Q_{safe} 로 표현했다.

```
function repulsiveForce(pose,laserPoints,Qsafe,Qinf,epsilon_r)
    local d,deltaX,deltaY,Fx,Fy,F
    Fx=0
    Fy=0
    for i=1,#laserPoint,1 do
```

```

        deltaX=laserPoints[i][1]
        deltaY=laserPoints[i][2]
        d=math.sqrt(deltaX^2+deltaY^2)
        if Qsafe<d and d<=Qinf then
            Fx=Fx-epsilon_r*deltaX/d^3
            Fy=Fy-epsilon_r*deltaY/d^3
        end
    end
    F={math.cos(pose[3])*Fx-math.sin(pose[3])*Fy,math.sin(pose[3])*Fx+
    math.cos(pose[3])*Fy}
    return F
end

```

7. Function deepestDecentPF(pose,aForce,rForce)

- $u = J(q) \eta'_{ref}(q, q_d, l)$ $u=[W_l \ W_r]^T$
- $$J(q) = \frac{1}{R} \begin{bmatrix} \cos \theta + \frac{b}{e} \sin \theta & \sin \theta - \frac{b}{e} \cos \theta \\ \cos \theta - \frac{b}{e} \sin \theta & \sin \theta + \frac{b}{e} \cos \theta \end{bmatrix}$$
- $$\eta'_{ref}(q, q_d, l) = v_{ref} \begin{bmatrix} F_x \\ F_u \end{bmatrix} = v_{ref} (F_{attractive} + \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} F_{repulsive})$$
- 위의 수식을 바탕으로 deepestDecentPF()의 코드를 작성한다.

```

function deepestDecentPF(pose,aForce,rForce)
    local F,Fx,Fy,vxc,vyc,wL,wR
    Fx=aForce[1]+rForce[1]
    Fy=aForce[2]+rForce[2]
    F=math.sqrt(Fx^2+Fy^2)
    vxc=vref*Fx
    vyc=vref*Fy
    wL=(1/(e*wheel_radius))*((e*math.cos(pose[3])+b*math.sin(pose[3]))*vxc+
    (e*math.sin(pose[3])-b*math.cos(pose[3]))*vyc)
    wR=(1/(e*wheel_radius))*((e*math.cos(pose[3])-
    b*math.sin(pose[3]))*vxc+(e*math.sin(pose[3])+b*math.cos(pose[3]))*vyc)
    return wL,wR
end

```

8. Function PotentialForce()

- $U(q, q_g, q_0) = U_{attractive}(q, q_g) + U_{repulsive}(q, q_0)$
- q : Robot configuration, q_g : Goal configuration, q_0 : Obstacle configuration
- Potential Field는 Attractive Force와 Repulsive Force의 힘으로 설명되고, Pioneer p3dx의 움

직임은 function deepestDecentPF()으로부터 설정할 수 있다. 이를 통한 Potential Field는 다음과 같이 script를 작성할 수 있다.

```
function PotentialFields(pose,goalP,laserPoints)
    local aForce,rForce,wL,wR
    aForce=attractiveForce(pose,goalP,epsilon_q,epsilon_c)
    rForce=repulsiveForce(pose,laserPoints,Qsafe,Qinf,epsilon_r)
    wL,wR=deepestDecentPF(pose,aForce,rForce)
    return wL,wR
end
```

9. Function sysCall_actuation()

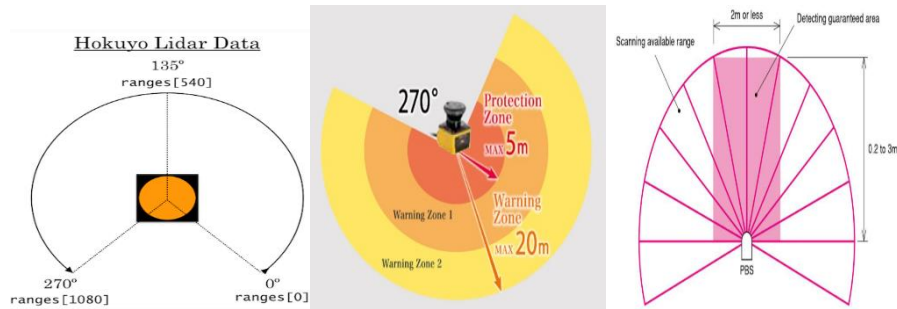
- function PotentialFields()을 통해 계산된 오른쪽과 왼쪽 바퀴의 속도를 계산할 수 있다.
- sim.setJointTargetVelocity(objectHandle, velocity) : objectHandle를 지정하고 속도를 입력함으로써, 객체의 속도를 입력한다.

```
function sysCall_actuation()
    local pose
    pose=updateRobotPose()
    if laserPoints then
        wL,wR=PotentialFields(pose,goalP,laserPoints)
        sim.setJointTargetVelocity(left_wheel,wL)
        sim.setJointTargetVelocity(right_wheel,wR)
    end
end
```

3. Hokuyo Sensor Script

1. Hokuyo Sensor의 Scan Angle

- Hokuyo Sensor의 경우 데이터를 가져오는 범위가 좌우 각각 270도이다. 이 때문에 부채꼴 모양의 스캐닝 범위 모양을 띠는 것을 알 수 있다. 가운데에서 측정하는 곳이 장애물이 있음을 측정하는 범위이고, 나머지 범위에 대해서는 전체적인 world의 데이터를 측정한다.



<FIG.19 Scan Angle of Hokuyo Sensor>

2. Sensor Joint

- Joint1과 Joint2의 값은 x, y축의 원점을 기준으로 왼쪽 방향으로 (= -45), 오른쪽방향으로 (= 45) 임의의 값을 설정하면 왼쪽 방향 시야(= -45) + 오른쪽 방향 시야(= 45) = 총 시야 범위 값 (= 90) 이 된다.
- `sim.setJointPosition(jointhandle(Joint 이름), -scanningangle/4(float 위치))`
- Potential Field Script에서 스캔 각도를 반으로 줄이면 Pioneer 3-DX가 목표 지점에 도착하지 못한다. 장애물을 인식할 때마다 방향을 바꾸어 좌우 바퀴가 돌아가는 원리로 작성되었기 때문에 장애물을 스캔 각도에서 인식하지 못해 목표 지점에 도착하지 못한다.

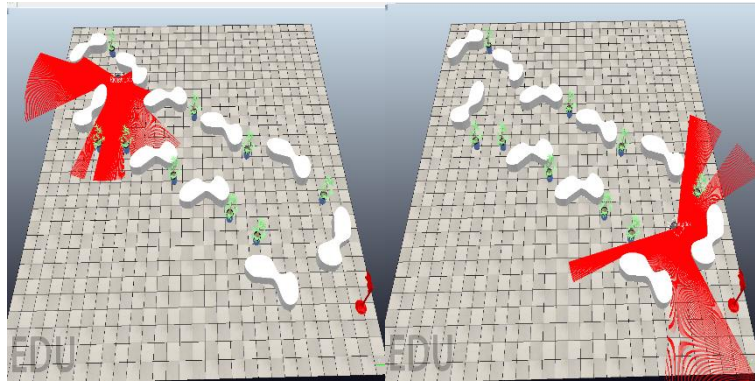
3. Hokuyo Sensor와 Potential Filed Script 연결

- `Sim, callScriptFunction('getMeasureDate@fastHokuyo', sim.scripttype_childscript)`
- Hokuyo Sensor Script 마지막에 사용되는 `getmeasureDate()` 함수에서 값을 받아 `sim.callScriptFunction`의 `@fastHokuyo`로 데이터를 보내 API에서 인식한다.

4. Potential Field Script Parameter 조작

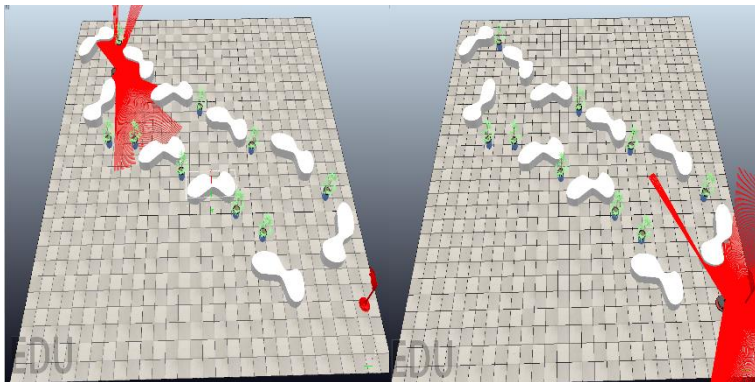
1. epsilon_r

- `epsilon_r`은 Repulsive Force와 관련된 변수로, 로봇이 장애물로부터 얼마만큼 영향을 받을지에 대한 변수이다. `epsilon_r`이 크면 장애물의 영향을 많이 받게 되고, 작으면 장애물과 충돌하게 된다.
- `epsilon_r=0.01`일 때 첫 구간과 중간 구간에서는 로봇과 목표 지점까지의 최적의 경로로 움직이는 것이 아닌, 왔다 갔다 하는 등 불필요한 경로를 지나갔다. 마지막 구간에서는 장애물 앞에서 방향을 회전하지 못하고 정지했다.



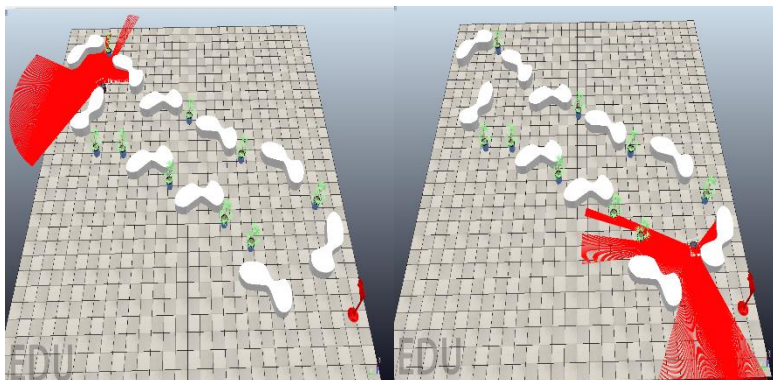
<FIG.20 epsilon_r=0.01>

- epsilon_r=0.005일 때는 첫 구간과 중간 구간을 이상 없이 지나갔고 불필요한 경로를 지나가지 않았다. 마지막 구간에서 방향을 회전하는데 시간이 걸렸지만 결국엔 목적지에 도착했다.



<FIG.21 epsilon_r=0.005>

- epsilon_r=0.001일 때는 첫 구간과 중간 구간을 이상 없이 지나갔다. 하지만 다른 경우와 다르게 장애물과의 거리가 가까운 채로 이동했다. 마지막 구간에서는 방향을 회전하는데 epsilon_r=0.005일때보다 오래 걸렸지만 목적지에 도착했다.



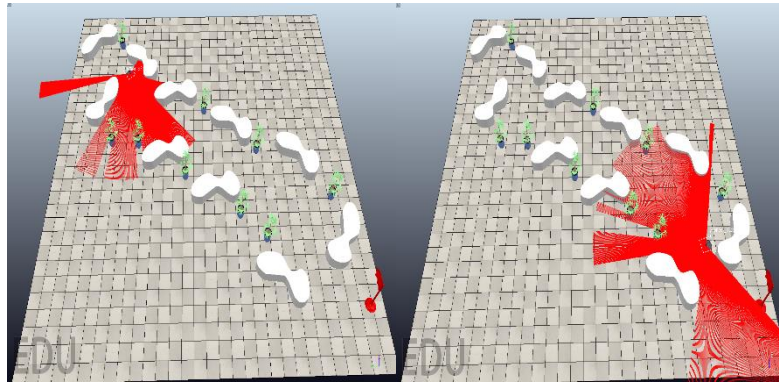
<FIG.22 epsilon_r=0.001>

- epsilon_r=0.01일 때는 epsilon_r의 값이 너무 커서 불필요한 경로를 지나갔고, 로봇이 이동하는데 많은 시간이 걸렸다. 그리고 목표 지점에 도착하지 못한 채 정지하였다.

- $\epsilon_r=0.005$ 와 $\epsilon_r=0.001$ 일 때는 $\epsilon_r=0.01$ 일 때와 다르게 무사히 목표 지점에 도착하였다. $\epsilon_r=0.005$ 일 때 목적지에 더 빠르게 도착하였으므로, ϵ_r 의 값은 0.005가 적합하다.

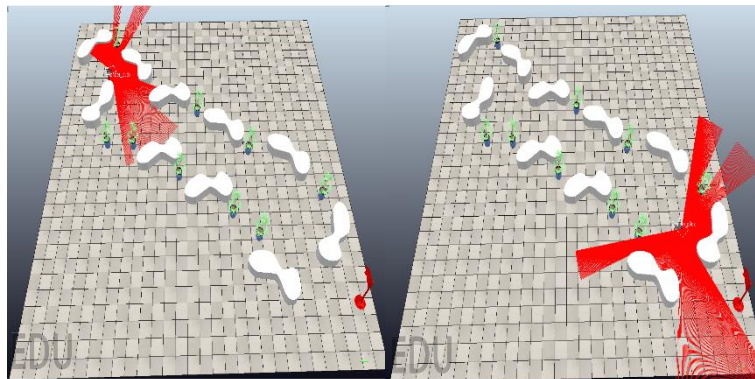
2. ϵ_c , ϵ_q

- ϵ_q 와 ϵ_c 는 Attractive Force와 관련된 변수이다. $QC=\epsilon_q/\epsilon_c$ 를 이용하여, 로봇과 목표 지점의 거리가 QC 보다 작으면 Attractive Force는 ϵ_q 의 영향을 받고 로봇과 목표 지점의 거리가 QC 보다 크면 Attractive Force는 ϵ_c 의 영향을 받는다.
- $\epsilon_q = 1$, $\epsilon_c = 2$ 일 때 첫 구간과 중간구간을 지날 때 불필요한 경로가 없었다. 마지막 구간에서 방향을 회전하는데 어려움이 있었지만 목적지에 도착했다. 첫 구간에서는 빠르게 이동하다가 목표 지점에 가까워질 때 로봇의 속도가 느려졌다.



<FIG.23 $\epsilon_q = 1$, $\epsilon_c = 2$ >

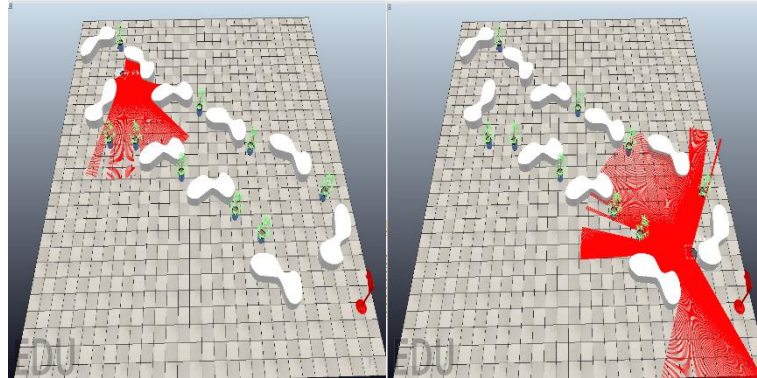
- $\epsilon_q = 2$, $\epsilon_c = 1$ 일 때는 첫 구간과 중간구간을 지날 때 불필요한 경로가 없었다. 마지막 구간에서는 시간이 다소 걸렸지만, 천천히 방향을 회전하다 목적지에 도착했다. 첫 구간에서는 다소 느리게 이동하다가, 목표 지점에 가까워질 때 로봇의 속도가 첫 구간에 비해 빨랐다.



<FIG.24 $\epsilon_q = 2$, $\epsilon_c = 1$ >

- $\epsilon_q = 3$, $\epsilon_c = 3$ 일 때는 다른 경우 보다 속도가 빨랐다. 첫 구간과 중간구간에서는

불필요한 경로가 없었다. 빠른 속도로 이동하다가 마지막 구간에서는 방향을 바꾸지 못하고 장애물에 부딪혔다. 그 뒤로는 움직이지 못했다.

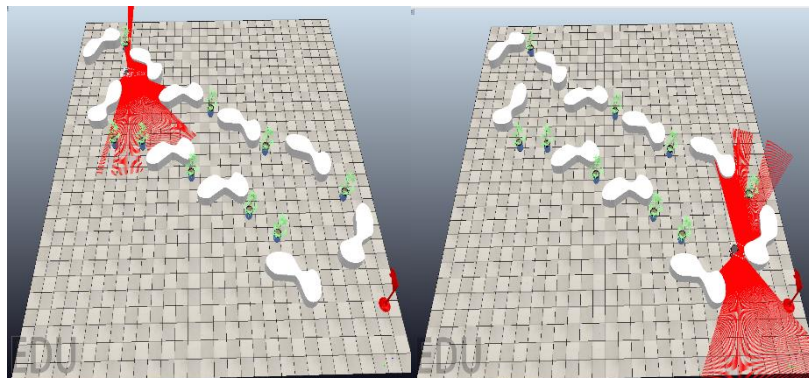


<FIG.25 epsilon_q = 3, epsilon_c = 3>

- epsilon_q = 3, epsilon_c = 3일 때와 같이 epsilon_q와 epsilon_c가 크면 전체적으로 로봇의 속도가 빨랐고, 두 변수가 작으면 로봇의 속도가 느렸다. epsilon_q=1, epsilon_c=2일 때 첫 구간과 중간구간에서는 epsilon_c의 영향을 받아 epsilon_q의 영향을 받는 마지막 구간보다 속도가 빨랐다. epsilon_q=2, epsilon_c=1일 때는 첫 구간에서는 속도가 느렸지만, 중간구간과 마지막 구간에서는 epsilon_q의 영향으로 빨라졌다. 장애물 사이의 거리가 좁은 마지막 구간에서는 속도를 줄여 진입하는 것이 바람직해 보이므로 epsilon_q은 1, epsilon_c은 2가 적합하다.

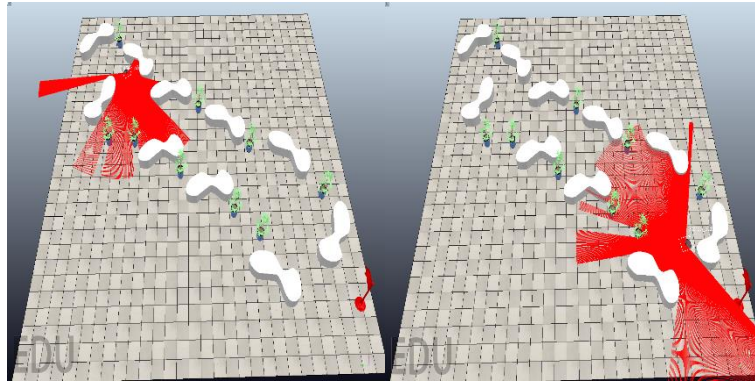
3. Qsafe

- Qsafe는 Repulsive Force와 관련된 변수로, 로봇과 장애물 사이의 안전거리를 나타낸다.
- Qsafe=0.5일 때는 첫 구간과 중간구간을 지날 때 불필요한 움직임 없이 이동하였다. 하지만 마지막 구간에서 방향을 바꾸는 도중에 장애물에 부딪혔고 정지했다.



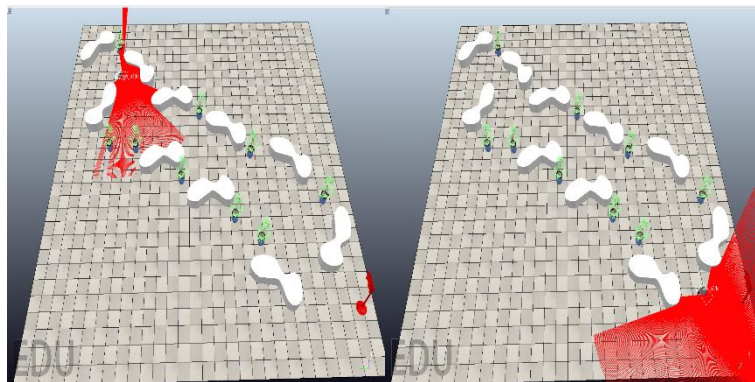
<FIG.26 Qsafe-0.5>

- Qsafe=0.1일 때는 첫 구간과 중간구간을 지날 때 이상 없이 이동했다. 불필요한 경로도 없었고 속도도 적당했다. 마지막 구간에서는 다소 시간이 걸렸지만 부딪힘 없이 목표 지점에 도착했다.



<FIG.27 Qsafe=0.1>

- Qsafe=0.05일 때는 다른 경우와 다르게 장애물과 가까운 거리에서 이동했다. 마지막 구간에서 이상 없이 장애물을 피해 회전했고 빠르게 목표 지점에 도착했다.

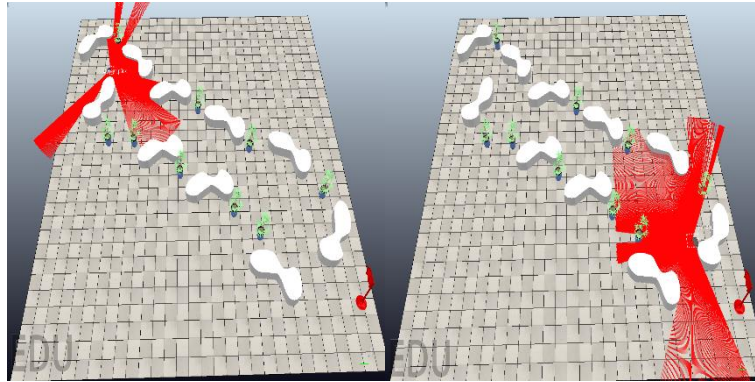


<FIG.28 Qsafe=0.05>

- Qsafe=0.05일 때와 같이 Qsafe가 작으면 장애물과 거리가 가까워져 다소 불안하게 로봇이 이동했고, Qsafe=0.5일 때와 같이 Qsafe가 크면 장애물들 사이의 거리가 좁은 구간을 이동하는 데 어려움이 있었다. 그렇기에 두 경우의 중간 값인 Qsafe=0.1이 적합하다.

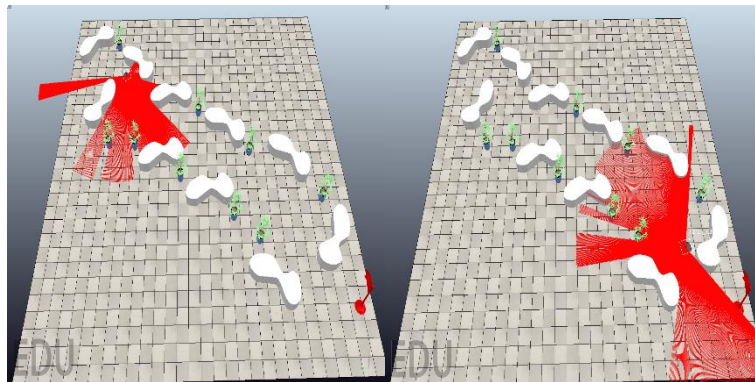
4. Qinf

- Qinf는 Repulsive Force와 관련된 변수로, 장애물의 영향을 받기 시작하는 거리를 나타낸다.
- Qinf=2일 때는 첫 구간과 중간 구간은 이상 없이 지나갔다. 하지만 마지막 구간에서 장애물에 부딪혀 정지했다.



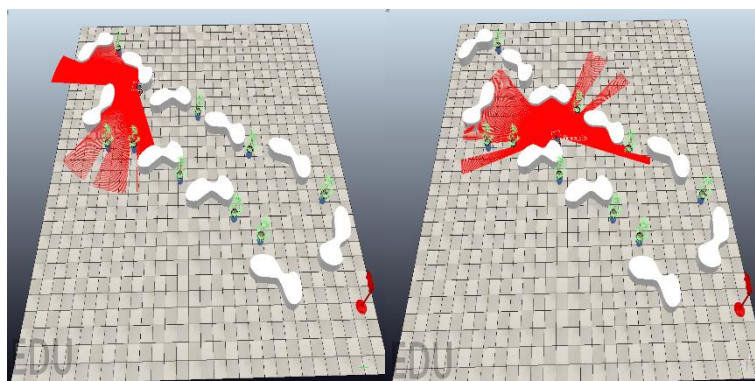
<FIG.29 Qinf=2>

- Qinf=1일 때는 첫 구간과 중간 구간은 이상 없이 지나갔다. 불필요한 경로도 없었고 속도도 적당했다. 마지막 구간에서는 다소 시간이 걸렸지만 부딪힘 없이 목표 지점에 도착했다.



<FIG.30 Qinf=1>

- Qinf=0.5일 때는 다른 경우와 다르게 장애물과의 간격이 가까운 채로 이동했다. 첫 구간에서부터 경로를 제대로 확인하지 못하였고 속도 조절을 하지 못했다. 결국 중간 구간에서 장애물에 부딪혀 정지했다.



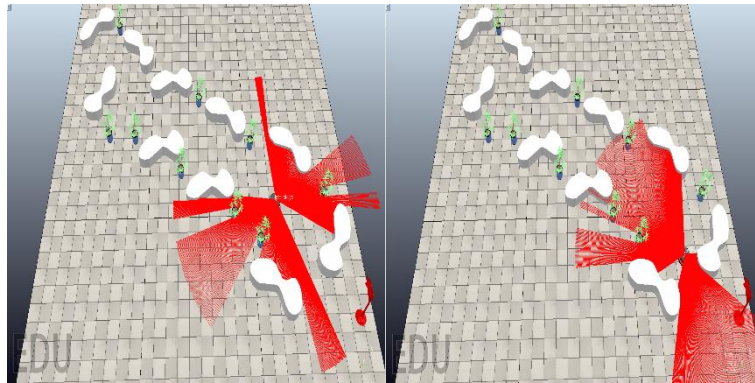
<FIG.31 Qinf=0.5>

- Qinf 값이 작으면 장애물과 거리가 가까워져 장애물에 부딪히기 전에 즉각적인 반응이 어려워지고, Qinf가 크면 장애물들 사이의 거리가 좁은 구간을 이동하는 데 어려움이 있다. 그렇기에

Qinf=1이 적합하다.

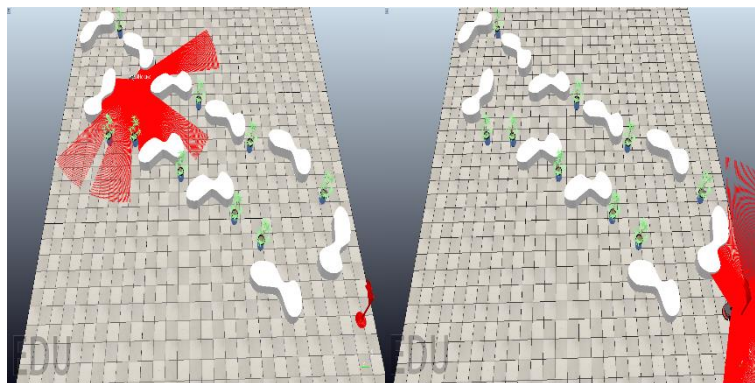
5. Scanning angle

- Scanning angle은 Hokuyo Sensor의 장애물 인식 앵글의 각도를 조절하는 변수를 나타낸다.
- Scanning angle/3일 때는 Sensing 시야각이 양쪽으로 펼쳐진 부채꼴 모양으로 생성되고 첫 구간부터 마지막 구간까지 장애물의 인식과 방향 변경이 잘 이루어져 목적지에 도달했다.



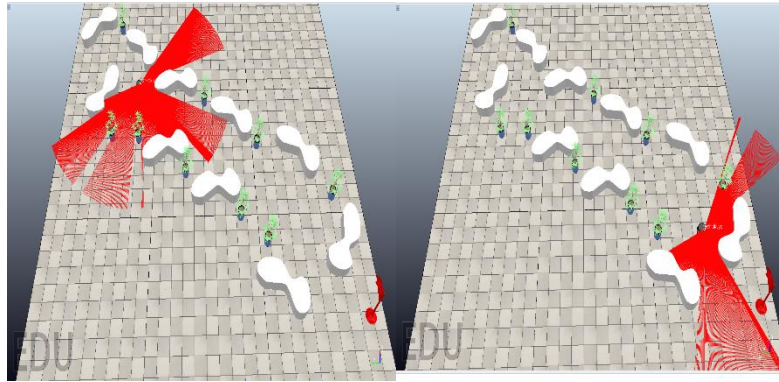
<FIG.32 Scanning angle/3>

- Scanning angle/4일 때는 Sensing 시야각이 실제 Hokuyo Sensor의 범위와 같은 540도로 펼쳐져서 장애물을 인식했다. 첫 구간부터 마지막 구간까지 장애물의 인식과 방향 변경이 빠르게 이루어져 변수 중 가장 빠른 시간 내에 목표 지점에 도착했다



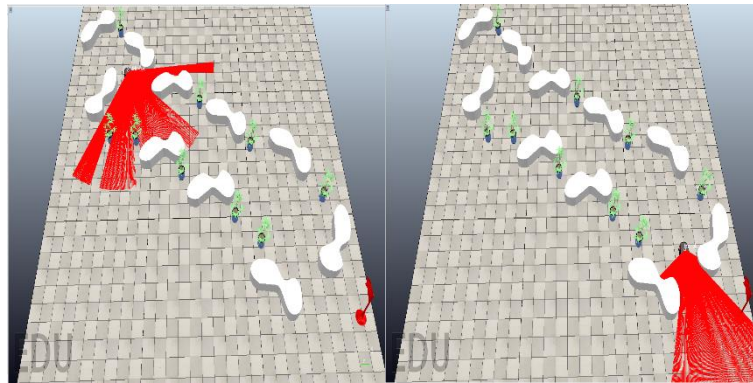
<FIG.33 Scanning angle/4>

- Scanning angle/5일 때는 Scanning angle/4일 때와 비슷한 시야각이 생성됐지만, 장애물 사이의 거리가 좁은 구간에서 방향을 틀지 못하고 그 자리에 정지했다.



<FIG.34 Scanning angle/5>

- Scanning angle/10일 때는 시야각이 좁은 범위에서 생성됐다. 장애물을 인식하는 범위를 p3dx의 정면 시야를 중심으로 이전의 경우보다 촘촘하고 좁게 형성되었다. 첫 구간부터 마지막 구간까지 장애물을 잘 인식했지만, 목표 지점에 도달하는 시간은 다른 경우에 비해 길었다.



<FIG.35 Scanning angle/10>

- Scanning angle/3 경우 양쪽으로 장애물을 파악하여 방향을 잘 회전했지만, 장애물이 좁은 간격으로 연속적으로 있는 구간에서는 방향을 회전하지 못했다. Scanning angle/5 경우는 로봇의 앞쪽에 Sensing 시야가 장애물 간격이 좁을 때 방향을 회전하지 못하고 정지했고, 인식범위가 고르게 분포하지 못한 것으로 보였다. Scanning angle/10의 경우에는 인식 범위가 촘촘하되 좁은 범위로 시야각이 형성되어서 도착 시간이 길어지는 것으로 보였다. 따라서 넓은 센싱 시야각과 고른 인식범위를 가진 Scanning angle/4일때가 가장 최적의 값이라고 판단된다.

5. Conclusion

1. Parameters 조정 결과 최적의 Parameters 값은 다음과 같다

$\epsilon_q = 1$

$\epsilon_c = 2$

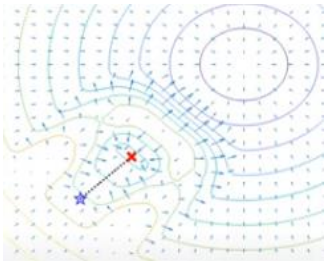
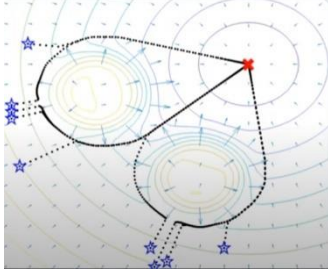
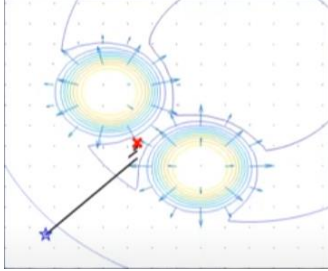
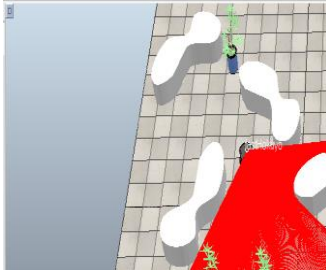
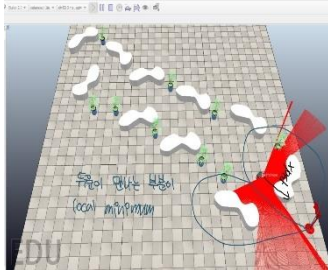
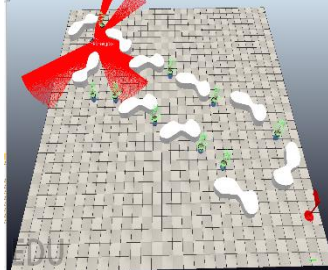
$\epsilon = 0.005$

$Q_{safe} = 0.1$

$Q_{inf} = 1$

Scanning Angle = 4분할

2. Potential Field에는 몇 가지 문제점들이 존재한다. 대표적인 문제들은 다음과 같다.

문제점	U-Shape	Local Minimum	Too Influent Objects
이론			
문제 발생			
원인	CoppeliaSim 프로그램에서 Hokuyo Sensor의 센싱 시야가 U자형이기 때문에 Local Minimum에 갇힐 수 있는 가능성이 존재	장애물에 의해 생기는 Local Force의 접점, 즉 Local Minimum(최솟값)을 통과하는 Path Planning을 세우지 못하면 로봇이 Force에 갇혀 움직이지 못하게 됨	장애물 자체에서 발생하는 Force가 로봇의 Total Force보다 커서 로봇이 장애물을 pass 할 수 없음
해결 방안	Hokuyo 센서를 제외한 다른 shape를 가진 센서를 추가	로봇이 가지는 Force 값을 변경하거나 센싱 시야각의 분배를 다르게 해서 해결	로봇의 Total Force를 높이기 위한 설정 값을 증가시켜 해결