LIMO 라이다 주행

목 차

1. Lidar 센서를 통한 장애물 인식
2. Lidar 센서를 통한 터널주행

01 1. Lidar 센서를 통한 장애물 인식

- ydlidar 준비

  -. ydlidar_ros_driver 실행

  $ roslaunch ydlidar_ros_driver X2.launch

  -. X2.launch 파일 확인

  : 라이다 좌표 reversion ‘true’

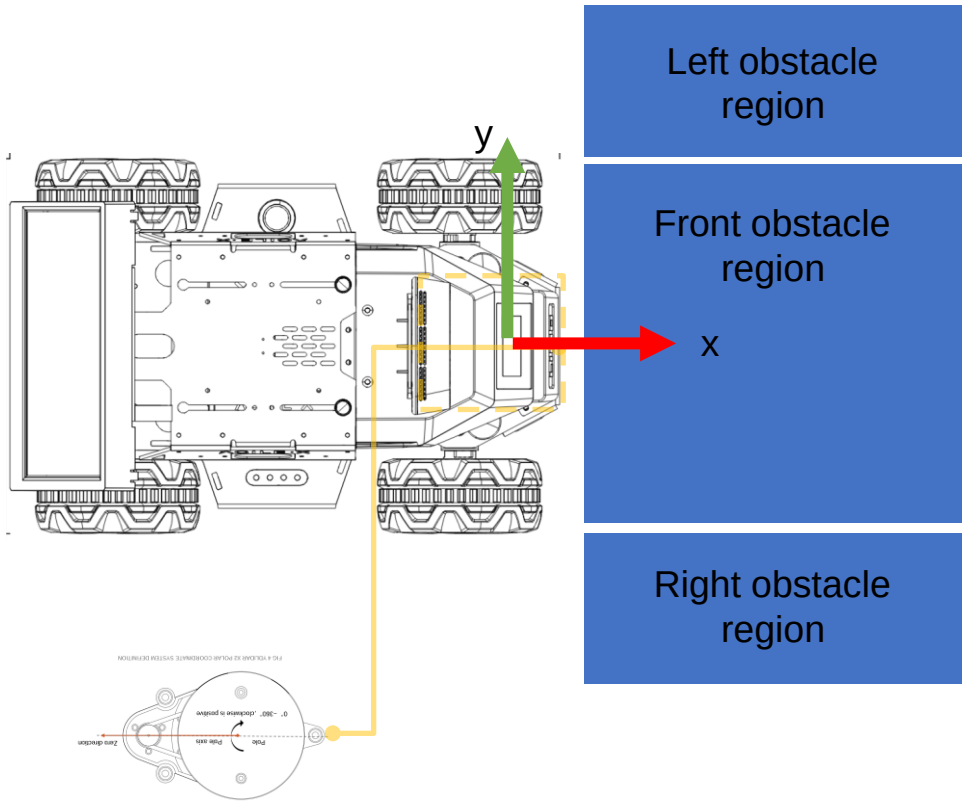
  : 라이다 측정 범위 -100~100

```
<launch>
  <node name="ydlidar_lidar_publisher"  pkg="ydlidar_ros_driver"  type="ydlidar_
respawn="false" >
    <!-- string property -->
    <param name="port"         type="string" value="/dev/ydlidar"/>
    <param name="frame_id"     type="string" value="laser_frame"/>
    <param name="ignore_array"     type="string" value=""/>

    <!-- int property -->
    <param name="baudrate"         type="int" value="115200"/>
    <!-- 0:TYPE_TOF, 1:TYPE_TRIANGLE, 2:TYPE_TOF_NET -->
    <param name="lidar_type"       type="int" value="1"/>
    <!-- 0:YDLIDAR_TYPE_SERIAL, 1:YDLIDAR_TYPE_TCP -->
    <param name="device_type"         type="int" value="0"/>
    <param name="sample_rate"         type="int" value="3"/>
    <param name="abnormal_check_count"         type="int" value="4"/>

    <!-- bool property -->
    <param name="resolution_fixed"    type="bool"   value="true"/>
    <param name="auto_reconnect"    type="bool"   value="true"/>
    <param name="reversion"    type="bool"   value="true"/>
    <param name="inverted"    type="bool"   value="true"/>
    <param name="isSingleChannel"    type="bool"   value="true"/>
    <param name="intensity"    type="bool"   value="false"/>
    <param name="support_motor_dtr"    type="bool"   value="true"/>
    <param name="invalid_range_is_inf"    type="bool"   value="false"/>
    <param name="point_cloud_preservative"    type="bool"   value="false"/>

    <!-- float property -->
    <param name="angle_min"    type="double" value="-90" />
    <param name="angle_max"    type="double" value="90" />
    <param name="range_min"    type="double" value="0.1" />
    <param name="range_max"    type="double" value="12.0" />
    <!-- frequency is invalid, External PWM control speed -->
    <param name="frequency"    type="double" value="10.0"/>
```

01 1. Lidar 센서를 통한 장애물 인식

- 라이다를 통한 장애물 인식 실습

-. 장애물 인식(판단) 알고리즘



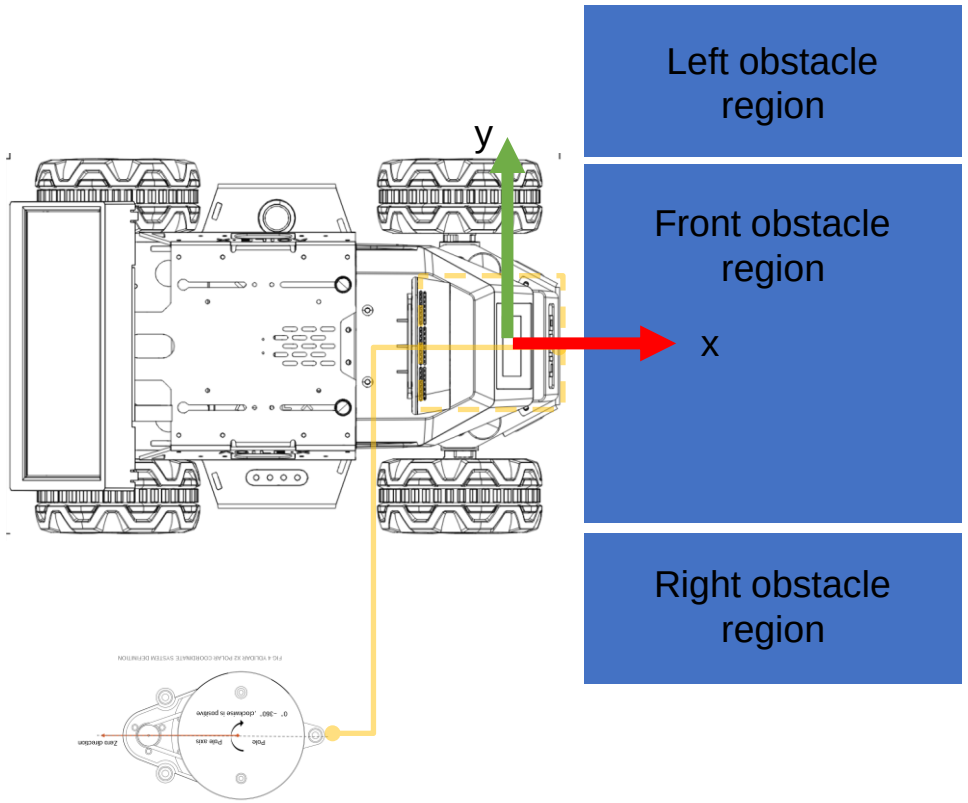

```
def lidar_CB(self, msg):
    if msg != -1:  # 유효한 메시지가 들어왔을 경우:
        self.laser_msg = msg  # LaserScan 메시지를 
        self.laser_flag = True  # LaserScan 메시지 
    else:
        self.laser_flag = False  # 유효하지 않은 메
```

```
def sense(self):  # 감지 함수
    pub_dist90_R_lsit = []
    pub_dist90_L_lsit = []
    pub_dist45_R_lsit = []
    pub_dist45_L_lsit = []
    pub_obs_R_list = []
    pub_obs_L_list = []
    pub_obs_C_list = []
```

01 LIMO 원격접속 및 ROS 환경설정

- 라이다를 통한 장애물 인식 실습

  -. 장애물 인식(판단) 알고리즘

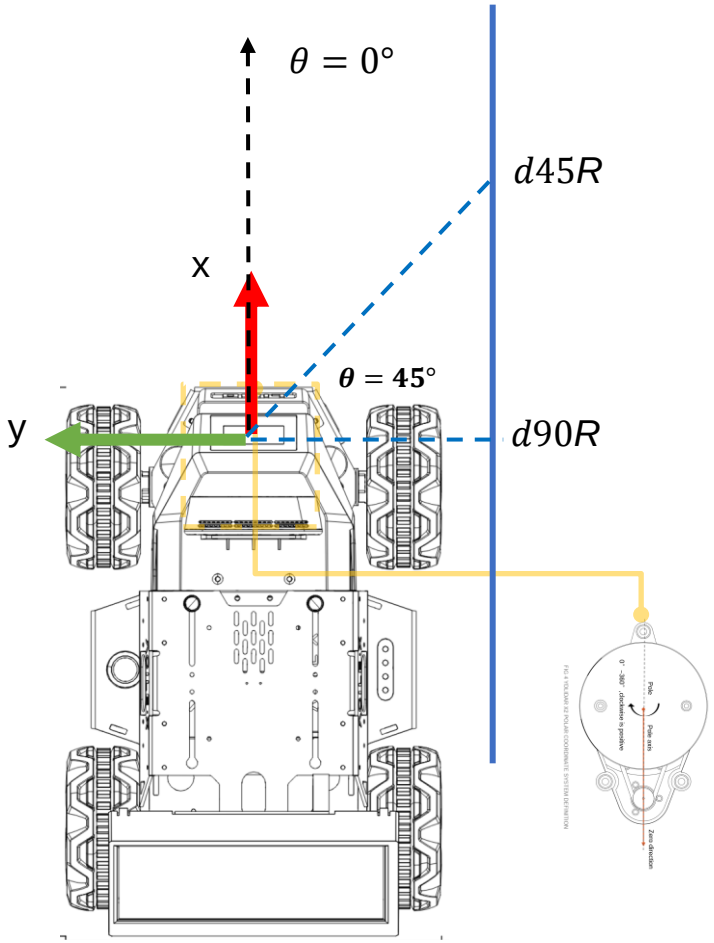



```
60          for i, n in enumerate(self.ranges):
61              ## x, y 좌표로 변환
62              x = n * math.cos(self.degrees[i] * math.pi / 180)
63              y = n * math.sin(self.degrees[i] * math.pi / 180)
64              # right obstacle detection
65              if 0.15 < y < 0.45 and 0 < x < 0.5:
66                  pub_obs_R_list.append(n)
67              # front obstacle detection
68              if -0.15 < y < 0.15 and 0 < x < 0.5:
69                  pub_obs_C_list.append(n)
70              # left obstacle detection
71              if -0.45 < y < -0.15 and 0 < x < 0.5:
72                  pub_obs_L_list.append(n)
```

01 LIMO 원격접속 및 ROS 환경설정

- 라이다를 통한 장애물 인식 실습

-. 장애물 인식(판단) 알고리즘



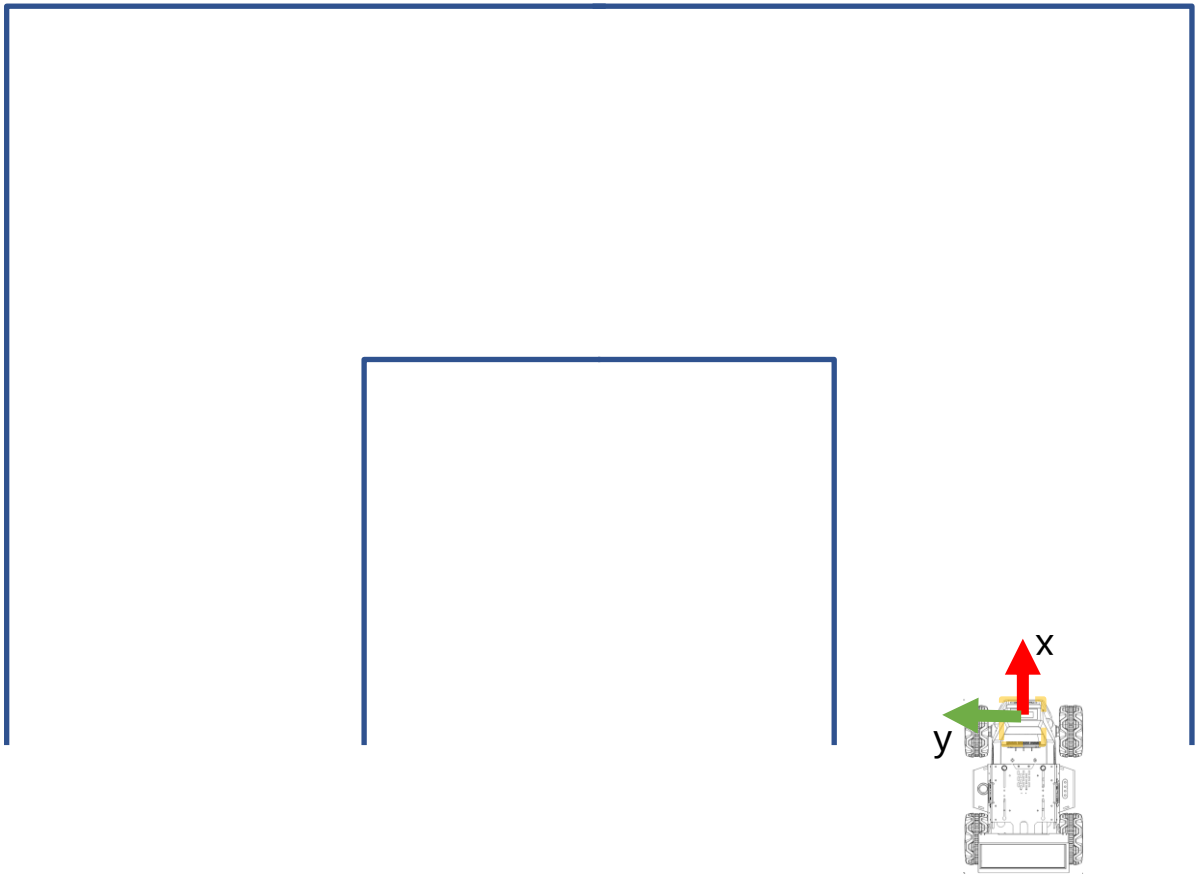

```
60          for i, n in enumerate(self.ranges):
61              ## x, y 좌표로 변환
62              x = n * math.cos(self.degrees[i] * math.pi / 180)
63              y = n * math.sin(self.degrees[i] * math.pi / 180)
64              # right obstacle detection
65              if 0.15 < y < 0.45 and 0 < x < 0.5:
66                  pub_obs_R_list.append(n)
67              # front obstacle detection
68              if -0.15 < y < 0.15 and 0 < x < 0.5:
69                  pub_obs_C_list.append(n)
70              # left obstacle detection
71              if -0.45 < y < -0.15 and 0 < x < 0.5:
72                  pub_obs_L_list.append(n)

74              # 오른쪽 90도 거리 저장
75              if 0 < n < 0.5 and -91 < self.degrees[i] < -89:
76                  pub_dist90_R_lsit.append(n)
77              # 오른쪽 45도 거리 저장
78              if 0 < n < 0.5 and -46 < self.degrees[i] < -44:
79                  pub_dist45_R_lsit.append(n)
```

01 LIMO 원격접속 및 ROS 환경설정

- 라이다를 통한 터널 통과(wall follower)

-. Wall follower 알고리즘

Front	Left	Right	Action
off	off	off	go straight
off	on	on	following the right wall
off	off	on	following the right wall
off	on	off	following the left wall
on	on	off	Turn right
on	on	on	Stop or back
on	off	on	Turn left

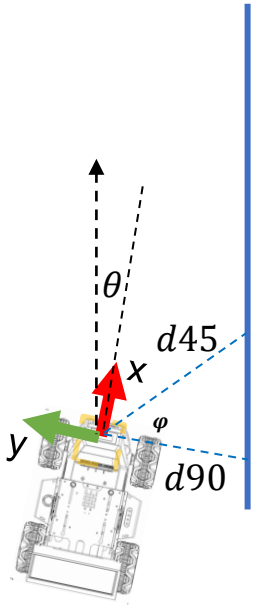


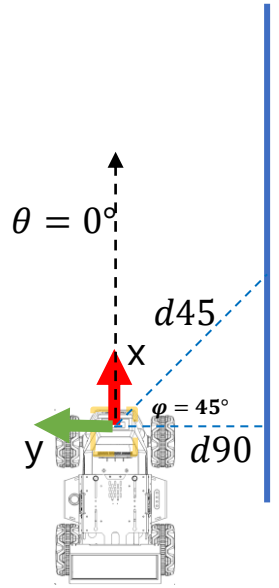
01 LIMO 원격접속 및 ROS 환경설정

- 라이다를 통한 터널 통과(wall follower)

-. Wall follower 알고리즘

Front	Left	Right	Action
off	off	off	go straight
off	on	on	following the right wall
off	off	on	following the right wall
off	on	off	following the left wall
on	on	off	Turn right
on	on	on	Stop or back
on	off	on	Turn left





$$\cos\varphi = \frac{d45}{d90}$$

$\theta = 0\text{deg}$ 이려면, $\varphi = 45\text{deg}$

$$\varphi = \text{acos}\frac{d45}{d90}$$

desired 벽과의 거리

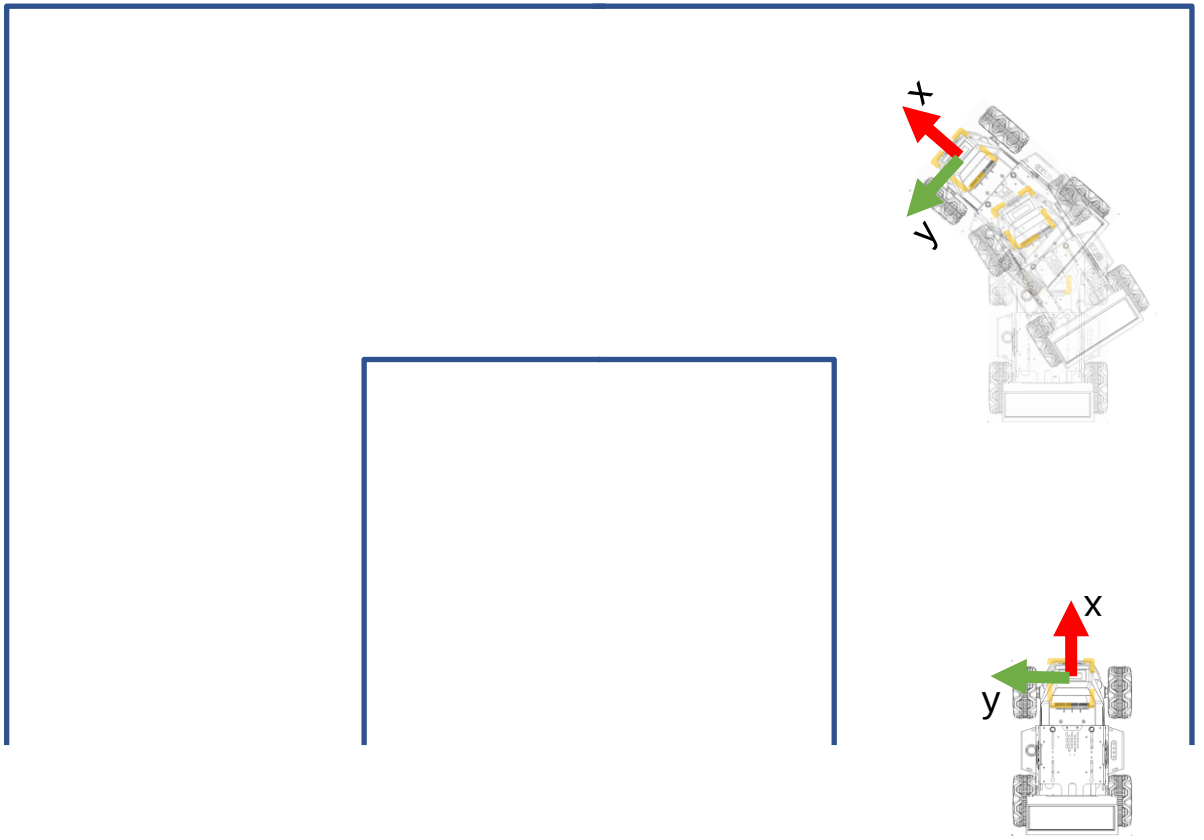
따라서, $e(t) = \left(\frac{\pi}{4} - \varphi\right) + (dist - d90)$

01 LIMO 원격접속 및 ROS 환경설정

- 라이다를 통한 터널 통과(wall follower)

-. Wall follower 알고리즘

Front	Left	Right	Action
off	off	off	go straight
off	on	on	following the right wall
off	off	on	following the right wall
off	on	off	following the left wall
on	on	off	Turn right
on	on	on	Stop or back
on	off	on	Turn left



01 LIMO 원격접속 및 ROS 환경설정

- 라이다를 통한 터널 통과(wall follower)

  -. Wall follower 알고리즘

Front	Left	Right	Action
off	off	off	go straight
off	on	on	following the right wall
off	off	on	following the right wall
off	on	off	following the left wall
on	on	off	Turn right
on	on	on	Stop or back
on	off	on	Turn left

```
64    def control(self):  # 제어 함수
65        if self.obsC and self.obsR and self.obsL: # Stop
66            speed = 0
67            steer = 0
68        elif not self.obsC and self.obsR: # following the right wall
69            speed = 0.2
70            theta = acos(self.dist45R / self.dist90R)*180/pi
71            steer = (45-theta) + self.kp*(self.wall_dist - self.dist90R)
72        elif self.obsC and self.obsR: # turn left
73            speed = 0.1
74            steer = 0.1
75        else:
76            print('조건 없음')
77            speed = 0
78            steer = 0
79
80        self.cmd_msg.linear.x = speed
81        self.cmd_msg.angular.z = steer
82        self.pub.publish(self.cmd_msg)
```