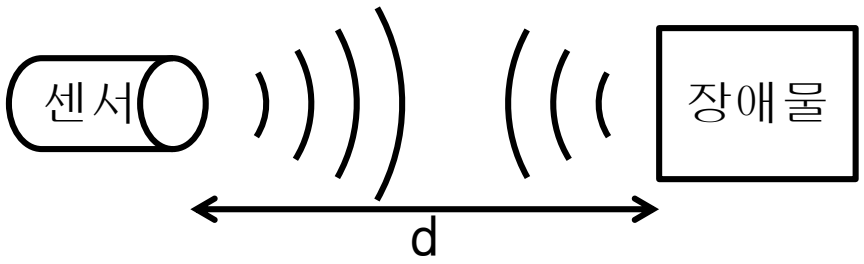
Lecture 5. 2D 라이다

Outline

- 거리 센서
- 2D 라이다를 이용한 장애물 회피

거리 센서 (Distance Sensor)

- 주변 물체와의 거리를 측정하는 센서
  - 초음파 센서
    - 소리(초음파)를 발사한 뒤 돌아올 때까지 시간 측정

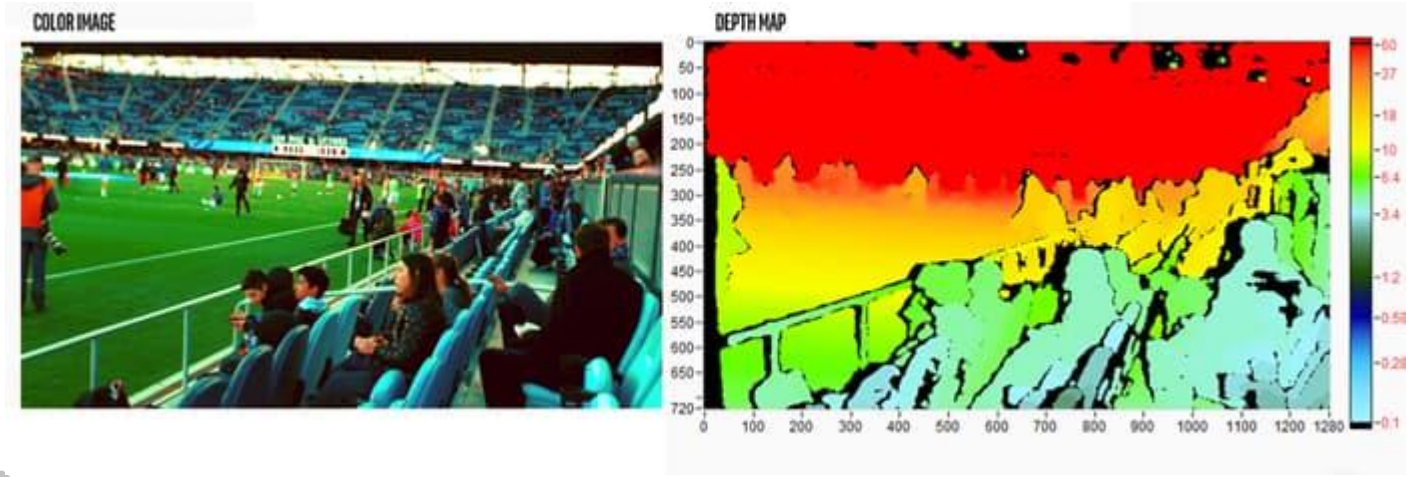


$$\Delta t = \frac{2d}{340m/s}$$

  - 레이다(Radar)
    - 전자파를 발사한 뒤 돌아오는 반사파를 분석하여 거리를 측정

거리 센서 (Distance Sensor)

- 주변 물체와의 거리를 측정하는 센서
  - 뎁스 카메라 (Depth Camera)
    - 특수한 무늬의 빛을 투사하여 비춰지는 모양을 분석
      - Microsoft Kinect, Intel RealSense SR305
    - Stereo 카메라를 이용하여 두 이미지의 차이를 분석
      - Intel RealSense D435i
    - 빛이 오고 가는 시간을 분석 (Time of Flight)
      - Microsoft Azure Kinect

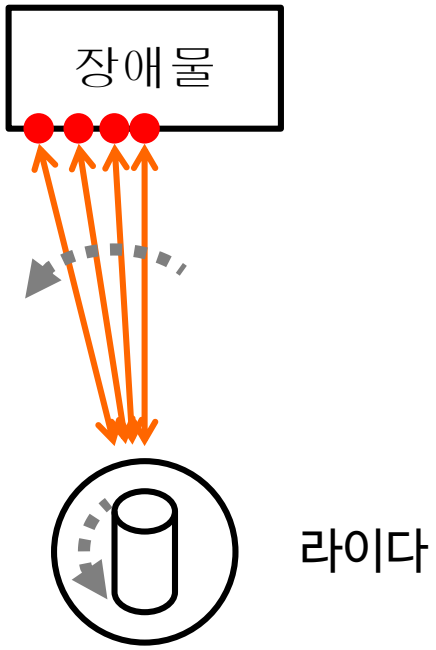


거리 센서 (Distance Sensor)

- 주변 물체와의 거리를 측정하는 센서
  - Laser Scanner (LiDAR, 라이다)
    - 2D LiDAR: SICK, RPLiDar, …
    - 3D LiDAR: Velodyne, …
    - Intel RealSense L515

2D 라이다

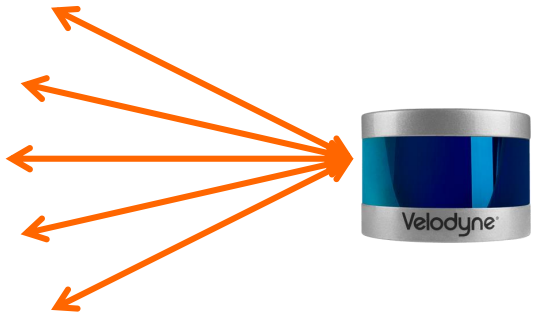
- 2D 라이다
  - 레이저를 회전하며 발사
  - 돌아오는 반사광을 분석
  - 각 각도에 대해 장애물까지의 거리 출력
  - 종류
    - 측정 범위: 180도, 360도, …
    - 각도 해상도: 1도 간격, 2도 간격, …
    - 측정 시작 각도: 0도, -90도, …
      - 정면이 0도



터틀봇의 라이다
- 360도
- 1도 간격
- 0도부터 측정 시작

3D 라이다

- 3D 라이다
  - 기본적으로는 2D 라이다가 상하로 기울어지며 측정
  - 채널: 상하 방향의 해상도
    - 16 채널, 32 채널, 64 채널 …



Outline

- 깊이 센서
- 2D 라이다를 이용한 장애물 회피

2D 라이다 데이터 자료형

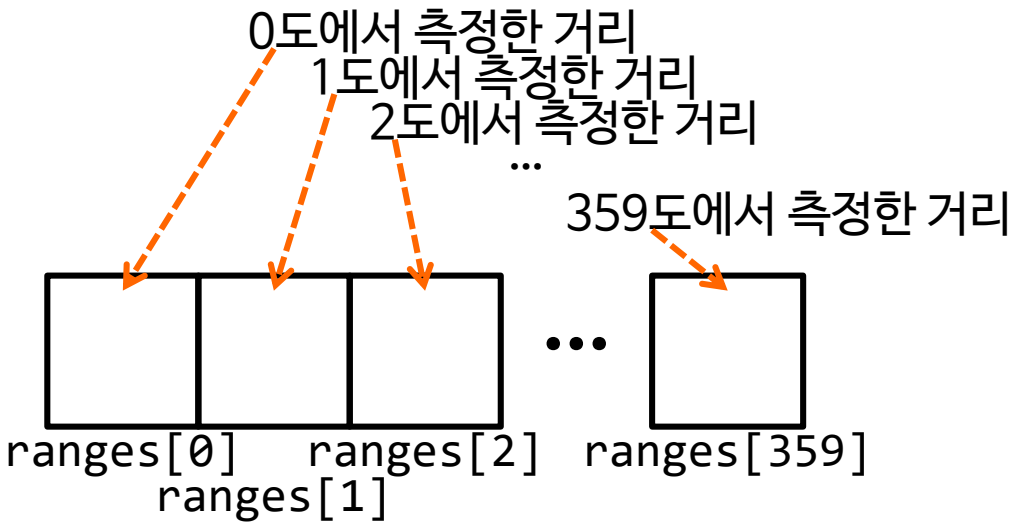
- LaserScan 메시지 자료형:

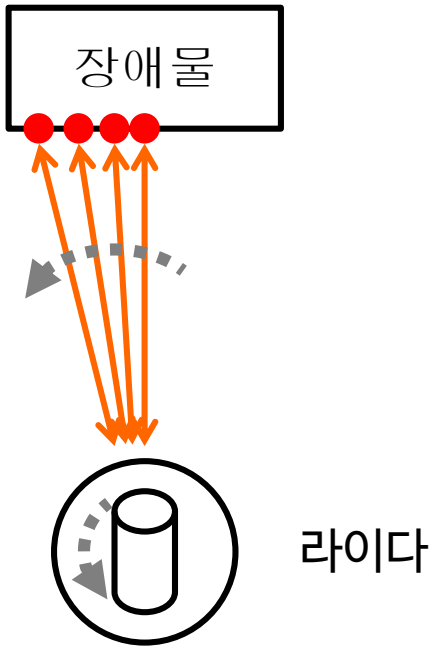
```
Header header            # timestamp in the header is the acquisition time of
                         # the first ray in the scan.
                         #
                         # in frame frame_id, angles are measured around
                         # the positive Z axis (counterclockwise, if Z is up)
                         # with zero angle being forward along the x axis

float32 angle_min        # start angle of the scan [rad]
float32 angle_max        # end angle of the scan [rad]
float32 angle_increment  # angular distance between measurements [rad]

float32 time_increment   # time between measurements [seconds] - if your scanner
                         # is moving, this will be used in interpolating position
                         # of 3d points
float32 scan_time        # time between scans [seconds]

float32 range_min        # minimum range value [m]
float32 range_max        # maximum range value [m]

float32[] ranges         # range data [m] (Note: values < range_min or
                         # > range_max should be discarded)
float32[] intensities    # intensity data [device-specific units].  If your
                         # device does not provide intensities, please leave
                         # the array empty.
```

각 각도에 대해 장애물까지의 거리

터틀봇 발행 scan 메시지

- 터틀봇의 라이다
  - 360도
  - 1도 간격
  - 0도부터 측정 시작





2D 라이다를 이용한 장애물 회피

- 전방에 장애물이 있으면 회전

```
#!/usr/bin/env python3
import rospy
from geometry_msgs.msg import Twist
from sensor_msgs.msg import LaserScan

def scan_cb(msg):
    global  range_ahead
    range_ahead = msg.ranges[0]
    print "range ahead: %0.1f" %  range_ahead

range_ahead = 0;
rospy.init_node('go_scan')
cmd_pub = rospy.Publisher('cmd_vel', Twist, queue_size = 1)
scan_sub = rospy.Subscriber('scan', LaserScan, scan_cb)

rate = rospy.Rate(10)
cmd = Twist()
```

함수 외부에서 이용하기 위해 외부 변수로 선언

정면(각도 0도) 방향의 장애물까지의 거리(ranges[0])를 range_ahead 변수에 저장

callback 함수

토픽 이름

자료형은 LaserScan

2D 라이다를 이용한 장애물 회피

- 전방에 장애물이 있으면 회전

```
while not rospy.is_shutdown():
    if range_ahead < 0.8:
        cmd.linear.x = 0
        cmd.angular.z = 0.2
    else:
        cmd.linear.x = 0.2
        cmd.angular.z = 0
    cmd_pub.publish(cmd)
    rate.sleep()
```

정면 장애물까지의 거리가 0.8m 미만이면

회전해서 장애물 회피

그렇지 않으면

전진

노드 만들기

- wanderbot 패키지 사용

```
user@hostname$ cd ~/my_ws/src/wanderbot/scripts
```

- 노드 소스 파일 작성

```
user@hostname$ gedit go_scan.py
```

- 실행 파일 지정

```
user@hostname$ chmod +x go_scan.py
```

노드 컴파일 및 실행

- 다른 터미널에서 TurtleBot3 월드 실행

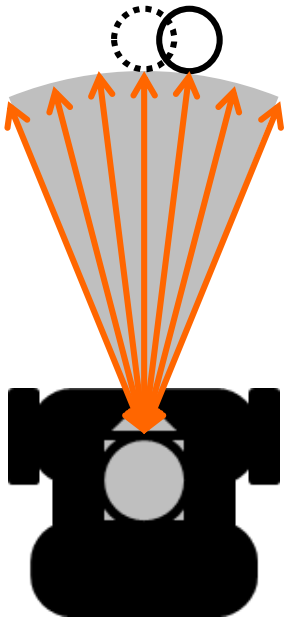
```
user@hostname$ cd ~/my_ws
user@hostname$ source devel/setup.bash
user@hostname$ roslaunch turtlebot3_gazebo turtlebot3_world.launch
```

- 실행

```
user@hostname$ cd ~/my_ws
user@hostname$ source devel/setup.bash
user@hostname$ rosrun wanderbot go_scan.py
```

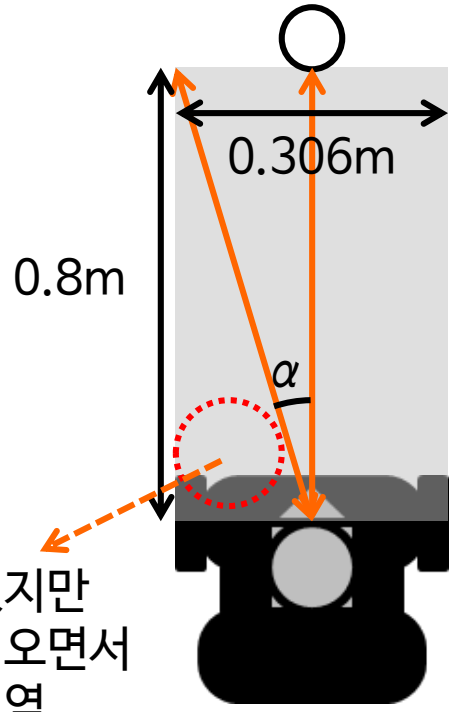
전방 장애물 확인

- 이전 예제 문제점: 전방 0도만 확인
- 충돌 예상 범위 안의 모든 방향에 대해 확인해야 함



전방 장애물 확인

- 장애물 확인 범위 계산하기
  - 거리가 0.8m 미만인 장애물을 회피한다면



사각 지대가 있지만
이전에 진행해 오면서
탐지 가능한 영역

$$\tan\alpha = \frac{0.306/2}{0.8}$$

$$\alpha = \tan^{-1}\frac{0.306/2}{0.8}$$

$$= 10.62°$$

약간의 여유를 넣어서
±15도 범위를 탐지

2D 라이다를 이용한 장애물 회피

- 15도 범위 안에 가장 가까이 있는 장애물 탐지

```
#!/usr/bin/env python3
import rospy
from geometry_msgs.msg import Twist
from sensor_msgs.msg import LaserScan

def scan_cb(msg):
    global  range_ahead
    range_ahead = 3.0
    for i in range(16):
        if msg.ranges[i] < range_ahead:
            range_ahead = msg.ranges[i]

    for i in range(359,344,-1):
        if msg.ranges[i] < range_ahead:
            range_ahead = msg.ranges[i]

    print "range ahead: %0.1f" %  range_ahead
```

0-15도 범위 탐지

345-359도 범위 탐지

2D 라이다 데이터 자료형

- LaserScan 메시지 자료형:

```
Header header            # timestamp in the header is the acquisition time of
                         # the first ray in the scan.
                         #
                         # in frame frame_id, angles are measured around
                         # the positive Z axis (counterclockwise, if Z is up)
                         # with zero angle being forward along the x axis

float32 angle_min        # start angle of the scan [rad]
float32 angle_max        # end angle of the scan [rad]
float32 angle_increment  # angular distance between measurements [rad]

float32 time_increment   # time between measurements [seconds] - if your scanner
                         # is moving, this will be used in interpolating position
                         # of 3d points
float32 scan_time        # time between scans [seconds]

float32 range_min        # minimum range value [m]
float32 range_max        # maximum range value [m]

float32[] ranges         # range data [m] (Note: values < range_min or
                         # > range_max should be discarded)
float32[] intensities    # intensity data [device-specific units].  If your
                         # device does not provide intensities, please leave
                         # the array empty.
```

정상적인 거리 값의 범위

2D 라이다를 이용한 장애물 회피

- 잘못 측정된 값 버리기

```
#!/usr/bin/env python3
import rospy
from geometry_msgs.msg import Twist
from sensor_msgs.msg import LaserScan

def scan_cb(msg):
    global  range_ahead
    range_ahead = 3.0
    for i in range(16):
        if msg.ranges[i] >= msg.range_min:
            if msg.ranges[i] < range_ahead:
                range_ahead = msg.ranges[i]
    for i in range(359,344,-1):
        if msg.ranges[i] >= msg.range_min:
            if msg.ranges[i] < range_ahead:
                range_ahead = msg.ranges[i]
    print "range ahead: %0.1f" %  range_ahead
```

2D 라이다 데이터 자료형

- LaserScan 메시지 자료형:

```
Header header            # timestamp in the header is the acquisition time of
                         # the first ray in the scan.
                         #
                         # in frame frame_id, angles are measured around
                         # the positive Z axis (counterclockwise, if Z is up)
                         # with zero angle being forward along the x axis

float32 angle_min        # start angle of the scan [rad]
float32 angle_max        # end angle of the scan [rad]
float32 angle_increment  # angular distance between measurements [rad]

float32 time_increment   # time between measurements [seconds] - if your scanner
                         # is moving, this will be used in interpolating position
                         # of 3d points
float32 scan_time        # time between scans [seconds]

float32 range_min        # minimum range value [m]
float32 range_max        # maximum range value [m]

float32[] ranges         # range data [m] (Note: values < range_min or
                         # > range_max should be discarded)
float32[] intensities    # intensity data [device-specific units].  If your
                         # device does not provide intensities, please leave
                         # the array empty.
```

각도 범위와 해상도