2019 GIST ROBOTICS FINAL PROJECT

TEAM 5

20155002 Kang rae-Young

20181105 Yoon Ju-Seon

20165012 Gong ji-kang

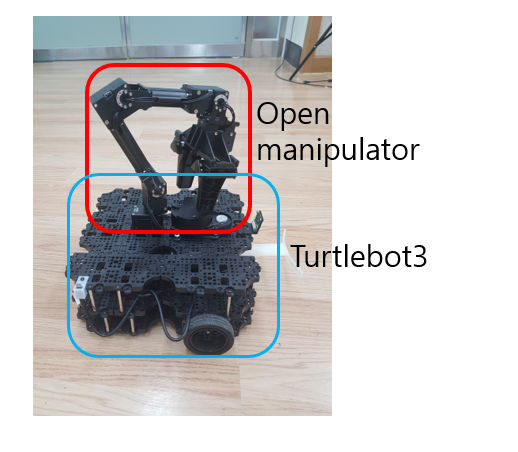
**Grasping Scattered Object using Turtlebot3 and Vision System**

1. **Overview**

* Objective : Make autonomous pick trash and place them in trash bin

|  |  |
| --- | --- |
|  |  |
| Bottle | Cup |
|  |  |
| Block | Trashbin |

* Platform : Turtlebot3 with open-manipulator



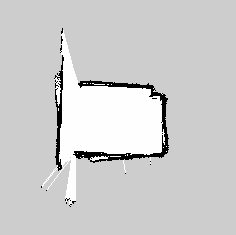
* Used Sensors : 2D\_LIDAR, webcam, raspicam, uwb sensor

|  |  |
| --- | --- |
|  |  |
| Webcam | Realsense |
|  |  |
| UWB | LiDar |

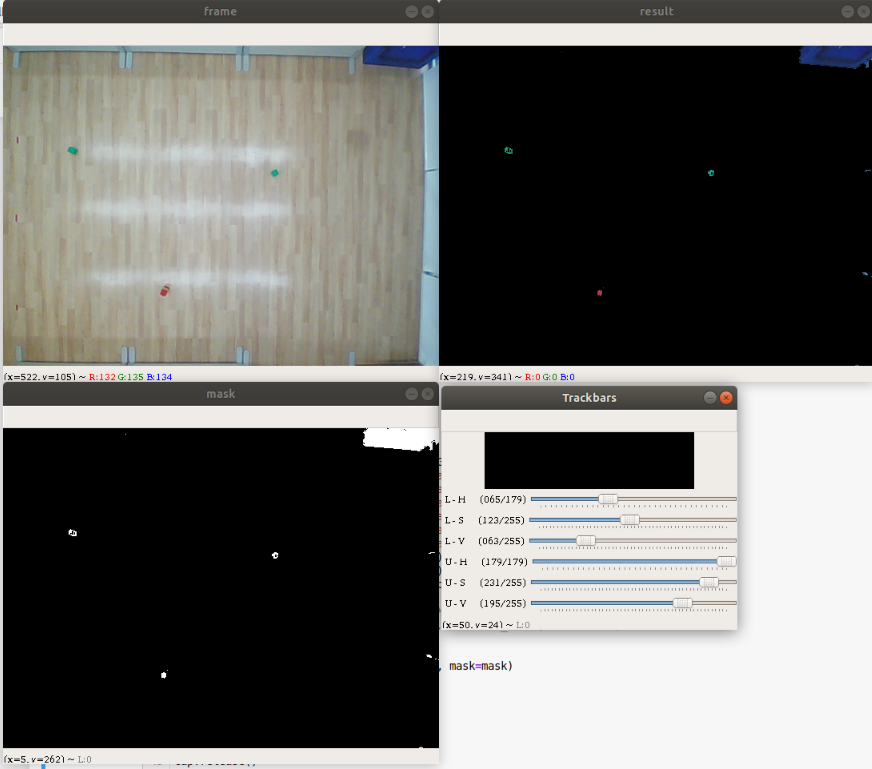
* Environment : ROS-kinetic, opencv3
* Used open source : turtlebot3 with open\_manipulation packages, turtlebot3 packages, cv\_bridge... etc

1. **Methodology**

* Initialize position of turtlebot : Using uwb sensor
* Get map of competition room : SLAM tool for turtlebot3



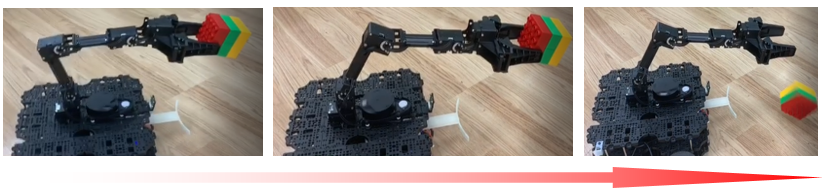
* Navigation : Navigation tool for turtlebot3
* Get trash position : image processing(color detection) , ceiling camera



* Set the position of turtlebot3 to pick up trash : image processing(color detection), raspicam, extra equipment



* Place trash : saved joint value state for pick up, lego, cup etc... and execute them



1. **Result**

* Task1 : How to find initial position and heading of platform
  + using uwb sensor to get initial position of turtlebot
  + commend to turtlebot going forward until get 1000 position of turtlebot
  + calculate linear regression equation to get slope of turtlebot

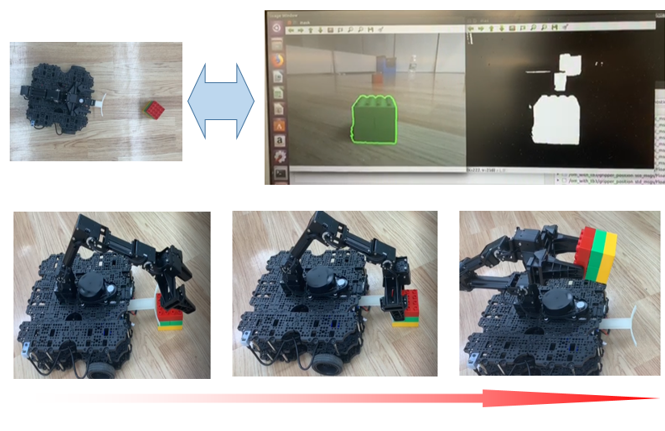
|  |
| --- |
|  |
| Initializing first location |
|  |
| Get the initialize position |

* Task2 : How to Detect 9 trash-objects
  + we detect 3 cup, 3 Lego.
  + get image from ceiling camera
  + find lego by color detect algorithm

|  |  |
| --- | --- |
|  |  |
| Find every object and get the distance | Find the nearest trash in input image |

We detect colors from ceiling cameras and look for objects. Then measure the distance between the turtle bot and the object. Pick up the object by approaching the shortest of the measured distances.

* Task3 : How to Navigate platform to each trash-object
  + use Navigation tool for turtlebot3
  + set goal position : get position of trash from image processing of ceiling camera
  + set goal angle : get direction from turtlebot to trash
  + set goal pos and angle in SLAM\_map coordinate : calculate transform equation by empirical method
* Task4 : How to Pick and drop the trash-object into trash-bin
  + save the trajectory of joint state like PickUp\_state, Place\_state, init\_state etc...
  + only can pick up the trash locate at the right front of turtlebot
  + set the turtlebot direction to locate the trash front of turtlebot : use image processing from raspicam
  + set the trash(lego) orientation : use extra equipment



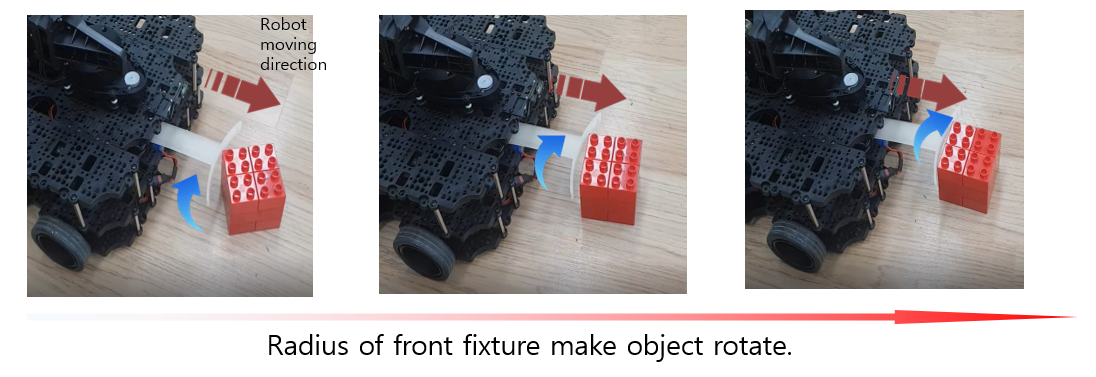
1. **How to Improve Task**
   * Making Sensor Fixtures

|  |  |
| --- | --- |
|  |  |
| UWB Fixture Model | Applying UWB Fixture on Turtlebot |
|  |  |
| Camera Fixture Model | Applying camera fixture on ceiling |

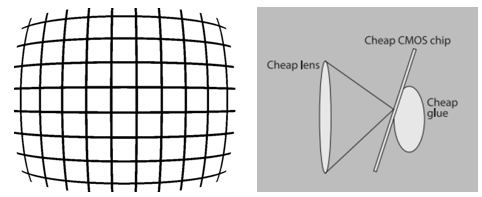
* + Front Support

|  |  |
| --- | --- |
|  |  |

We added support to make the object straight at the front of the robot. The robot arm is hard to catch because the object is placed in arbitrary form in front of the robot. In order to make a certain shape, a certain shape was added by 3d modeling.



* + Camera Calibration

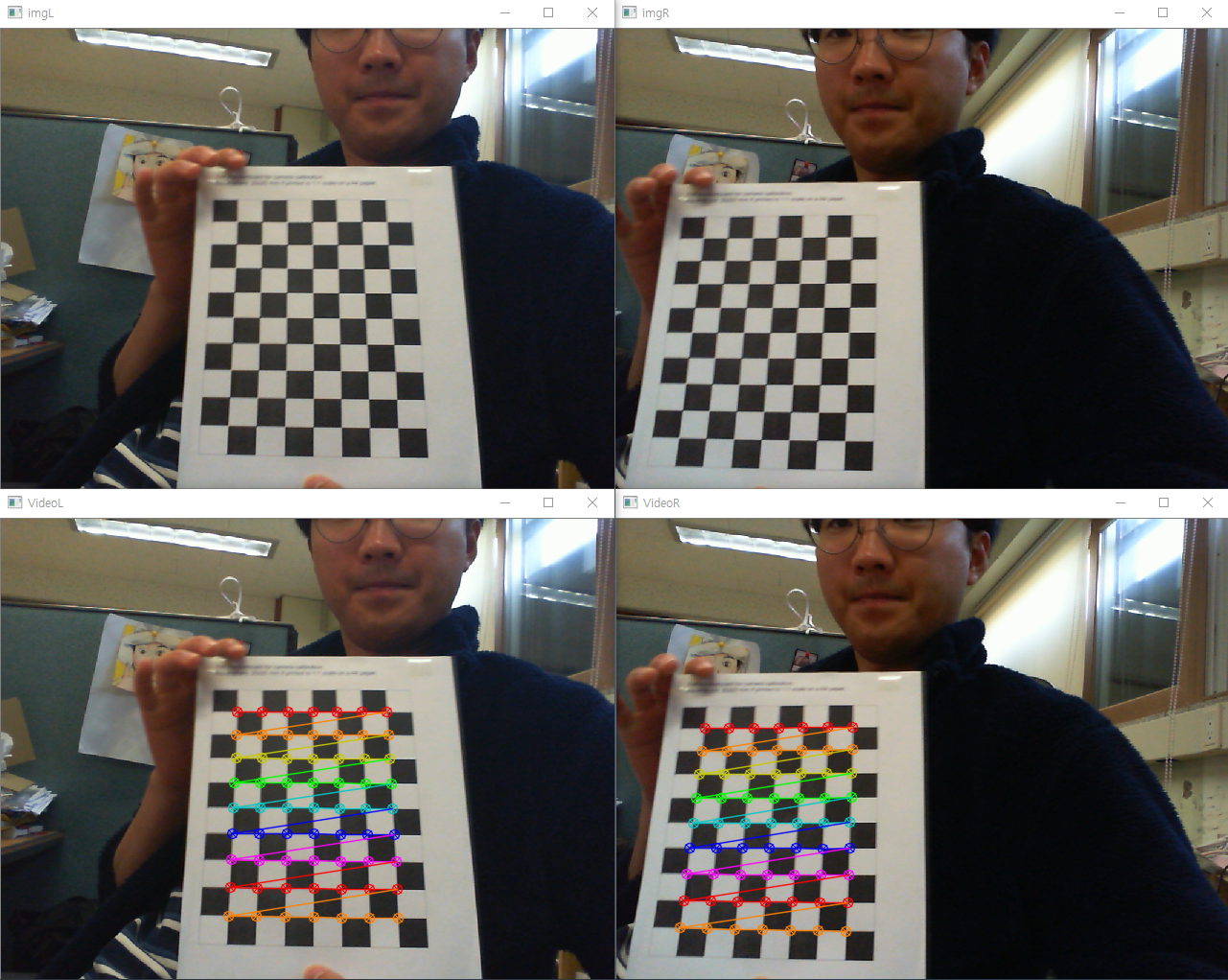


Factors of camera image distortion include focal length and optical center. This is a geometric parameter for where the camera is fixed (installation height, direction). The matrix is ​​expressed as follows.

are optical centers

Using Checker board(7\*9) implement camera calibration.

|  |  |
| --- | --- |
| Opencv Function | Explain |
| cv2.findChessboardCorners(gray, (n,m),None) | Finds the positions of internal corners of the chessboard. (MxN) |
| cv2.cornerSubPix(gray,corners,(11,11),(-1,-1),criteria) | Refines the corner locations.. corners is a parameter that represents input/output. |
| cv2.drawChessboardCorners(gray, (9,6), corners, ret) | Checking the corners on checkerboard, gray is input./output parameter |
| cv2.calibrateCamera | When the position of the vertex is input, the position of the corrected value is derived from the output. |
| cv2.getOptimalNewCameraMatrix | Obtain optimization values ​​from parameters obtained from multiple images |



camera calibration result using 7X9 checkerboard

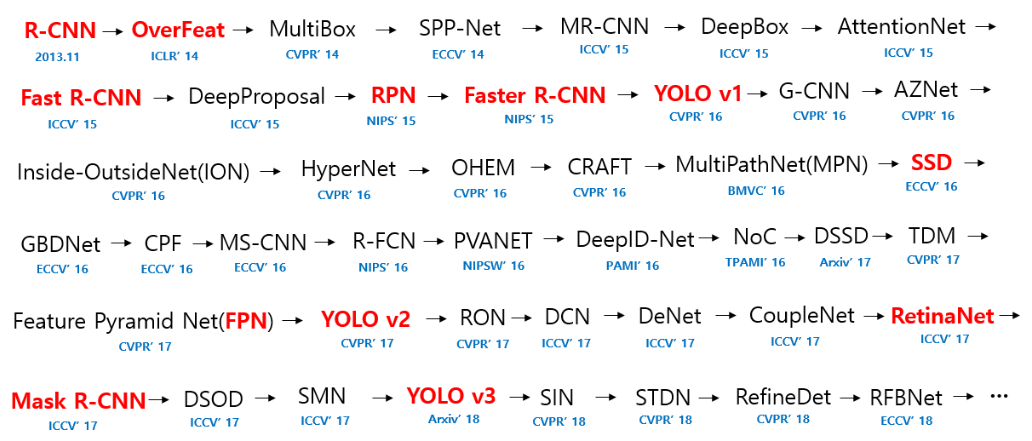
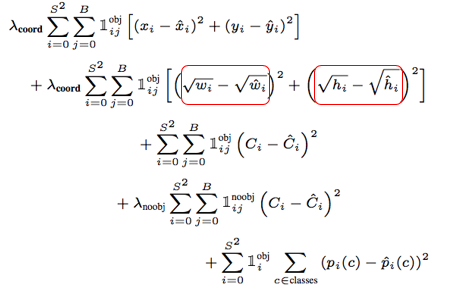
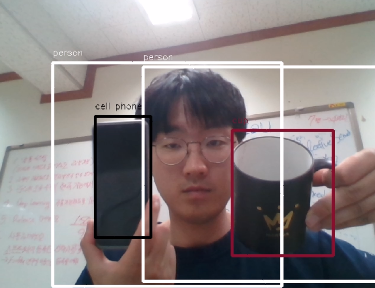
* + Deep Learning based Vision
* Object Detection - YOLO v3

Figure 9 Object Detection 알고리즘 변화도

The above figure is a flow figure showing the evolution of the object detection algorithm. The current object detection algorithm is divided into R-CNN and YOLO. R-CNN finds a region of interest and then performs classification on the classification layer. In the case of YOLO, classifies the bounding box and class simultaneously. It is faster than R-CNN because it searches for coordinates and carries out classification at the same time. (50 ~ 130fps) it has a learning layer based on GoogleNet, and the inference layer extracts one output using non-maximal suppression. However, there is a disadvantage that the type can not be found well if the close region is touched. Because, since the size of the object box in the loss function is reduced (), there is a case that the small object can not be detected well.



The result of implementing object detection with webcam in Coco pre-trained model is as follows.

YOLO Object Detection

* Learning costomized dataset(cup, bottle, turtlebot)

We learned with three objects. The results are as follows.

|  |  |
| --- | --- |
| Parameter | Unit |
| Algorithm | YOLO V3 |
| Total dataset & Label | 3,000 images |
| Epoch | 4,000,000 |
| Batch size | 100 |
| Threshold | 40% |
| Accuracy | 72% |

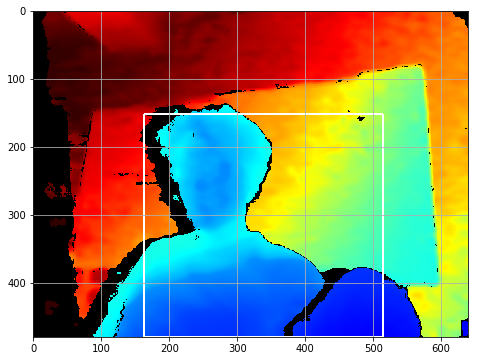
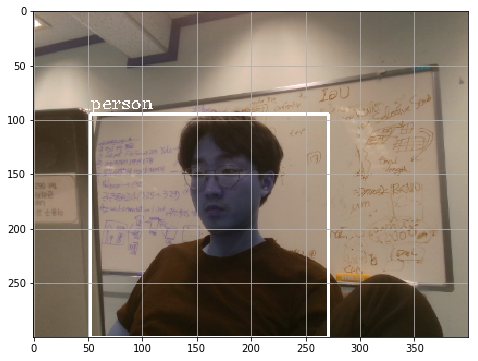
|  |  |
| --- | --- |
|  |  |
| Input = topview of webcam  (fixed camera) | Input = topview of webcam  (fixed pole camera) |
|  |  |
| Input = topview of webcam  (different environment) | Input Training image |

We were able to achieve 70% accuracy despite running 4,000,000 epochs. If you do this, you can see the same result as above. It could not be used because the accuracy was low.

* + Realsense2 Depth Implementation
* YOLO V3 + Depth Map with given realsense camera

We used a Tensorflow version of the YOLO algorithm, DarkFlow, to link the stereo camera.

The bounding box is hit on the target and the depth of the corresponding point is determined by defeating the depth map. Since the depth of each box is different, the depth is extracted by calculating the center point or the mean value. In addition, there is a method of extracting only the target instance in the box and obtaining the depth and averaging.

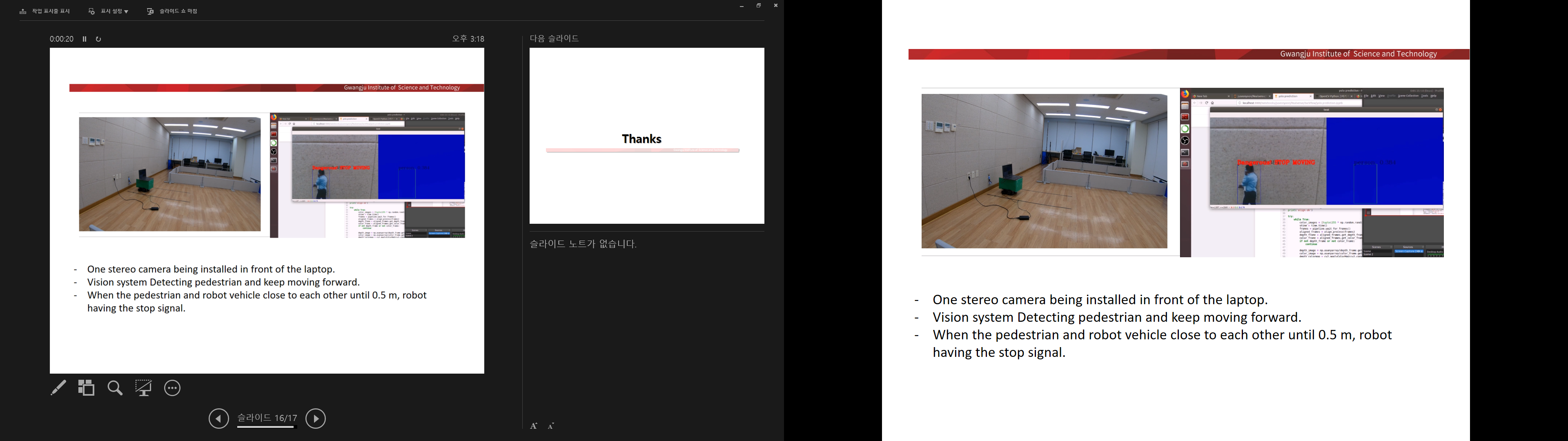


Detection (L) combined with depth map and extract depth(R)



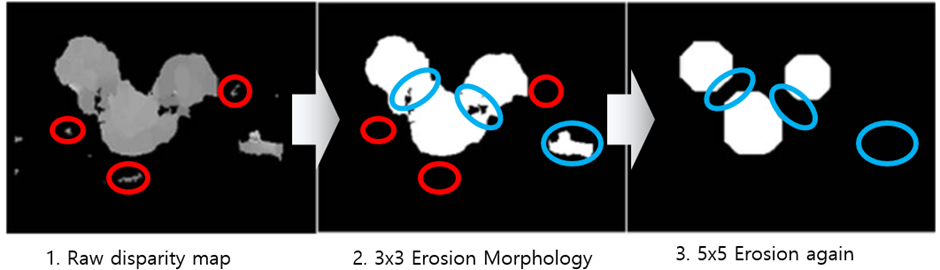
* Turtlebot + Object Detection + Depth Map

We have implemented an example that stops turtlebot when the distance of the detected object nears by using Bluetooth communication. Data transmission is transmitted from Turbo to remote PC via Bluetooth communication using Robot Operation System (ROS). The stopping distance was set within 0.5 m and the image of the person was set as an object. The Turtlebot moves about 1 km / h.



However, when using a camera on the ceiling, the field of view was too narrow to be used. When using turtlebot, only the relative distance of the object could be detected.

* + Color detection morphology
* What is morphology



Morphology is a technique used to sharpen a contour by processing a binary image. If you create a depth map using two raw images that are not preprocessed, it is difficult to determine the outline of the object. We use dilate to expand and tie the pixels, and erosion to reduce the expanded object to a proper outline.

|  |
| --- |
|  |
| Before morphology (white cup is very small) |
|  |
| After morphology (white cup is larger than before) |

1. **How to Run the Code**

**This is specified on the GITHUB**

[remote PC] [Terminal\_1] roscore

roscore

[turtlebot] [Terminal\_1] ROS\_NAMESPACE=om\_with\_tb3 roslaunch turtlebot3\_bringup turtlebot3\_robot.launch multi\_robot\_name:=om\_with\_tb3 set\_lidar\_frame\_id:=om\_with\_tb3/base\_scan

to bring up sensor data from turtlebot

[turtlebot] [Terminal\_2] ROS\_NAMESPACE=om\_with\_tb3 roslaunch turtlebot3\_bringup turtlebot3\_rpicamera.launch

to bring up camera image from raspicam

[turtlebot] [Terminal\_3] roslaunch healthcare\_robotics\_uwb healthcare\_robotics\_uwb\_loc.launch

to bring up uwb data from turtlebot

[remote PC] [Terminal\_2] ROS\_NAMESPACE=om\_with\_tb3 roslaunch open\_manipulator\_with\_tb3\_tools om\_with\_tb3\_robot.launch

to bring up sensor data from turtlebot

[remote PC] [Terminal\_3] rosrun image\_transport republish compressed in:=/om\_with\_tb3/raspicam\_node/image raw out:=/image\_raw

to transport compressed image to raw image

[remote PC] [Terminal\_4] roslaunch open\_manipulator\_with\_tb3\_tools manipulation.launch open\_rviz:=false

to activate manipulation move group

[remote PC] [Terminal\_5] roslaunch healthcare\_robotics\_uwb\_sub uwb\_sub\_py.launch

by using uwb sensor initialize the position and orientation of turtlebot and launch navigation tool

get uwb sensor data : subscribe uwb topic

commend to turtlebot : publish move command(twist) to /cmd topic

use linear regression equation to calculate slope

subprocess to execute navigation tool

[remote PC] [Terminal\_6] python getTrashPos.py

service server for get nearest trash position from ceiling camera image

service server that wait a sign from main node that find the trash and send position and direction of trash

find trash by color detection on hsv channel

only find legos... and turtlebot

find nearest trash from turtlebot

calculate them to trasform coordinate from image\_pixel to map\_coordinate

send postion and direction info by response of service

[remote PC] [Terminal\_7] python cv\_test\_node.py

service server for setting turtlebot direction to middle of trash by using raspicam

service server that wait a sign from main node and send them a type of trash

detect lego use color detection

commend to turtlebot rotate until lego locate middle of image from raspicam

In this algorithm there is no recognize code...

assume that trash is lego, and send type of trash = "lego" to main node by response of service

[remote PC] [Terminal\_8]rosrun kong\_test kong\_test\_node.py

service server for manipulation which execute saved trajectory for each trash

save the each trashes' pickup joints state(except joint1)

when trash type come from main node set trash joint state to that one and execute

open gripper

pick up trash state

close gripper

holding state

when "place" come from main node, execute

close gripper

place state

open gripper

holding state

[remote PC] [Terminal\_9]rosrun robotics\_main robotics\_main\_node.py

main node

service client for find\_trash\_pos, set\_trash\_mid\_locate, execute\_manipulation servers...

also commend turtlebot to move forward by publish to /cmd topic

overview : repeat a~f until there is no trash.

find trash position

move to near trash (to move trash on the front of turtlebot)

set the trash locate on the middle

go forward to use extra equipment

pickup trash

move to trash-bin and place trash