Care-Bot Ve-Ge-Tur, The Night Patrol Utilizing Deep Learning

INDEX

- INTRODUCTION
- CONFIGURATION
- DRIVING SYSTEM
- MULTI-PROCESS
- RECOGNITION SYSTEM
- TESTING
- CONCLUSION

INTRODUCTION

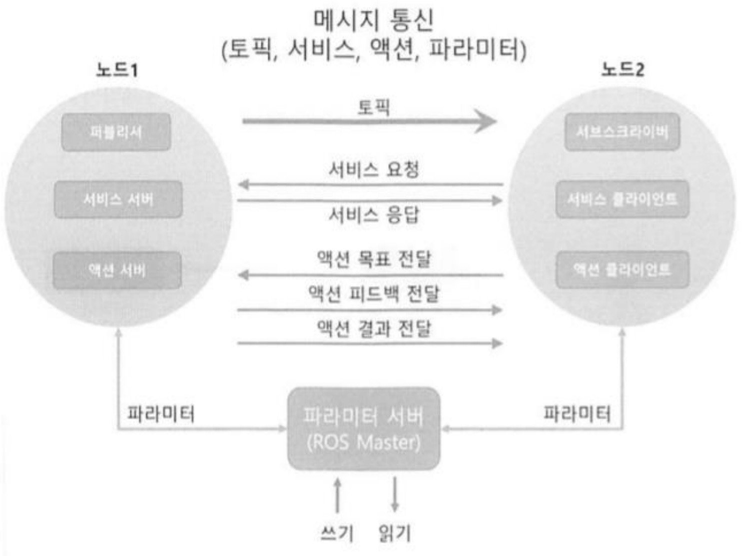
- We were given a turtle-bot to turn any idea that can solve real-world’s problems into service.
- We’ve come up with an idea that we can make service keeping the elderly safe with the given turtle-bot.
- Let the care-bot patrol around the nursing home and keep on watching the elderly to check that they are safe.
- If one is down, we send alarm to nurses.

CONFIGURATION

- Driving System
  - ROS Package
    - SLAM, NAVIGATION, TELEOP
- Recognition System
  - Object Detection based on Faster-RCNN
    - Check whether one is down or not
- Alarm System
  - BlueTooth
    - Ring alarm

- Communication System
  - SSH
    - Communication between remote PC & TurtleBot
  - VNC
    - Show TurtleBot environment on remote PC

DRIVING SYSTEM - ROS



node - A processing unit, units communicate each other by sending messages.

package - A set of nodes or a set of data.

message - Data

Types of messages

publisher --(1~n)--> **topic**(unidirectional) --(1~n)--> subsciber

server <- ---> **service**(bidirectional) < ------> client

server <-- ----> **action**(feedback) <- ----> client

parameter(can assign values from outside, ros master function)

DRIVING SYSTEM

ROS process

1. run master with roscore (master manages the communication between nodes)

1. run subscriber node with rosrun(send info of subscriber to master once when subscriber initializes)

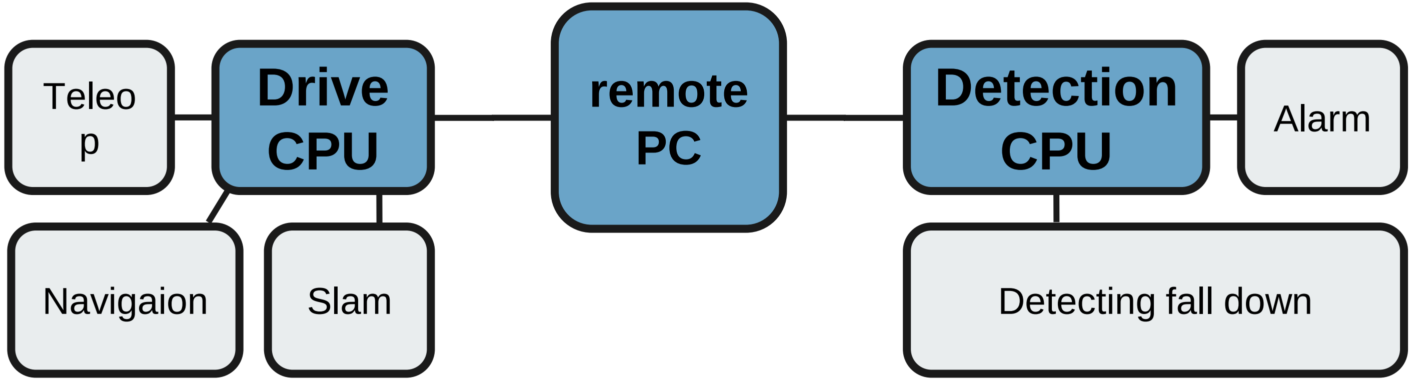
1. run publisher node with rosron(send info of publisher to master once when subscriber initializes)

1. Send topic to master, then after master checking topic, master send it to subscribers

1. After process 4, there is only one communication in the system left

MULTI-PROCESS

- The two CPUs for different usages



MULTI-PROCESS

The purpose of using 2 raspberry PIs and the expected events

- One CPU processing causes overload so the camera frame drops when running object detection

- By combining two Turtlebots into one, the tasks are distributed and operated on two CPUs

- By lowering the CPU's overload, it was possible to increase the frame of the camera during object detection.

MULTI-PROCESS

Instruction sample of ROS pakage

1. **Remote PC**

   run master node

- $ roscore

1. **Turtlebot**

   Turtlebot bringup

- $ roslaunch turtlebot3_bringup turtlebot3_robot.launch

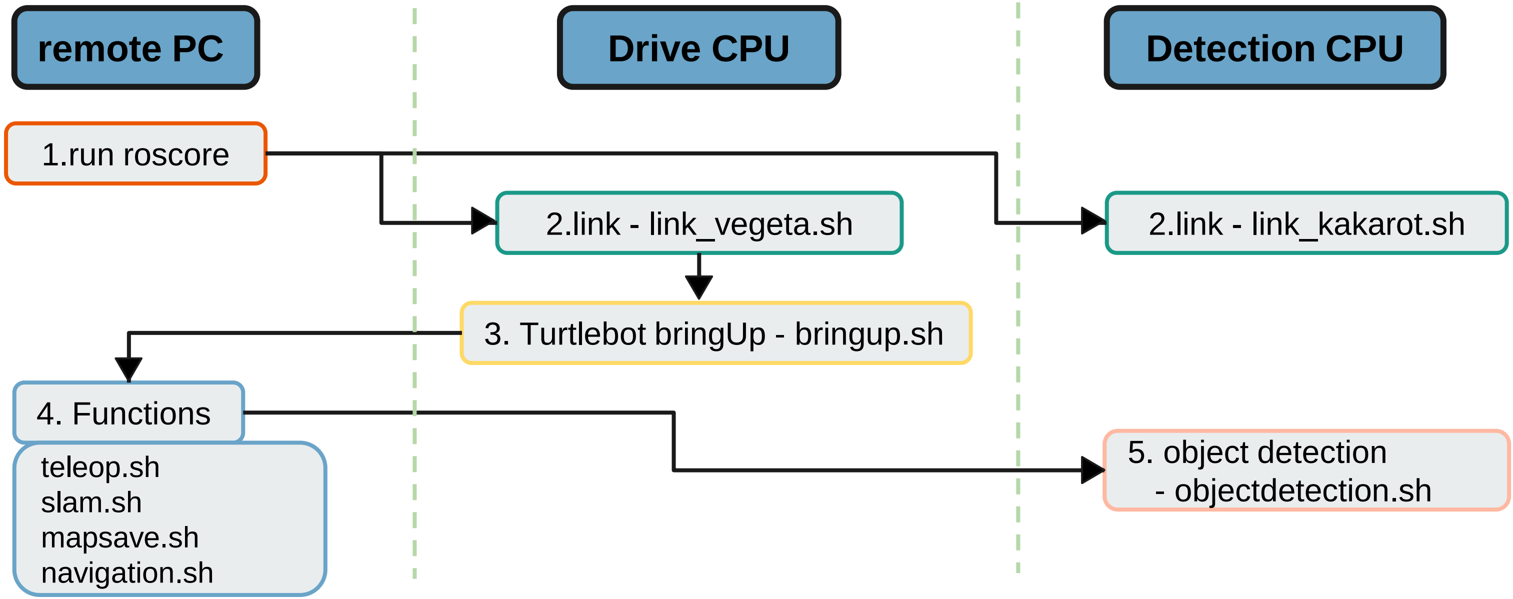
1. **Remote PC**

   run teleop(move by keybord)

- $ export TURTLEBOT3_MODEL=waffle_pi
- $ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch

MULTI-PROCESS

Made sh files for easy use of shell scripts files



remote PC

Drive CPU

Detection CPU

1.run roscore

2.link - link_vegeta.sh

2.link - link_kakarot.sh

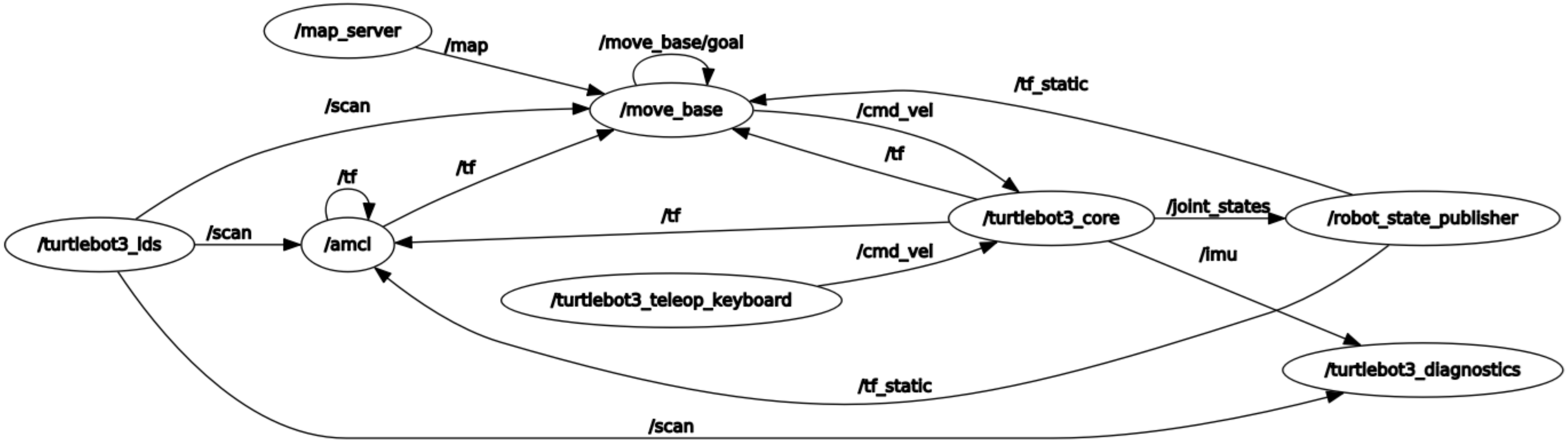
3. Turtlebot bringUp - bringup.sh

4. Functions

teleop.sh
slam.sh
mapsave.sh
navigation.sh

5. object detection
- objectdetection.sh

ROS Graph


/map_server
/map
/move_base/goal
/move_base
/scan
/tf_static
/cmd_vel
/tf
/tf
/tf
/amcl
/tf
/turtlebot3_core
/joint_states
/robot_state_publisher
/turtlebot3_lds
/scan
/cmd_vel
/imu
/turtlebot3_teleop_keyboard
/turtlebot3_diagnostics
/tf_static
/scan

COMMUNICATION SYSTEM

VNC

Shows the turtle-bots view on remote PC and allows PC to control the turtle-bot

turtlebot

Run Server

```
vncserver : 1
x11vnc
```

remote PC

Enter turtle-bot’s IP address to remote PC’s vnc viewer

COMMUNICATION SYSTEM

SSH

- An application that logs in to another computer on the network or executes commands on a remote system.

Input Turtelbot’s IP in terminal of the remote PC to connect.

ex)

- ssh pi@192.168.0.9
- password

HARDWARE

problems

- When creating a build file with catkin_make, there were many errors due to the conflict between cmake_list and pakage xml
- The Turtlebot's country time and that of remote PC's has to be same, since WIFI doesn't work.
- In Virtual environment, remote PC couldn't find the wireless LAN card. Need to connect adapter bridge to connect wireless LAN.
- Unstable Wifi.

RECOGNITION SYSTEM

- Based on FASTER-RCNN architecture
- Have been trained 2 networks in an End-to-End architecture
- One is Region Proposal Network
- The other one is Classification Network

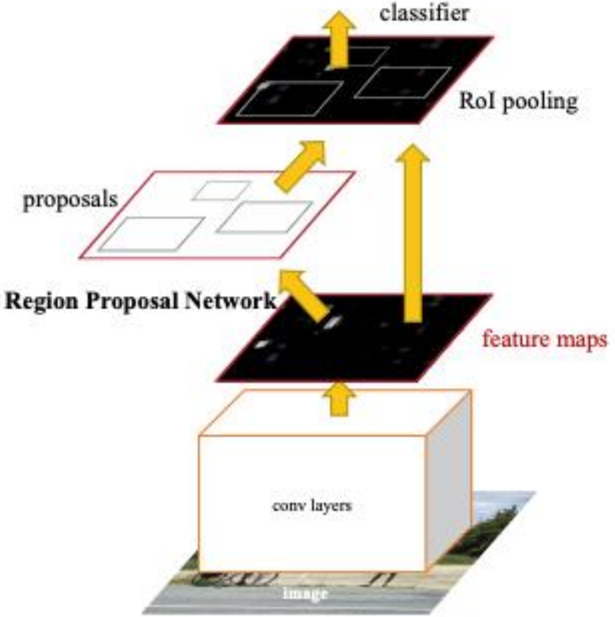
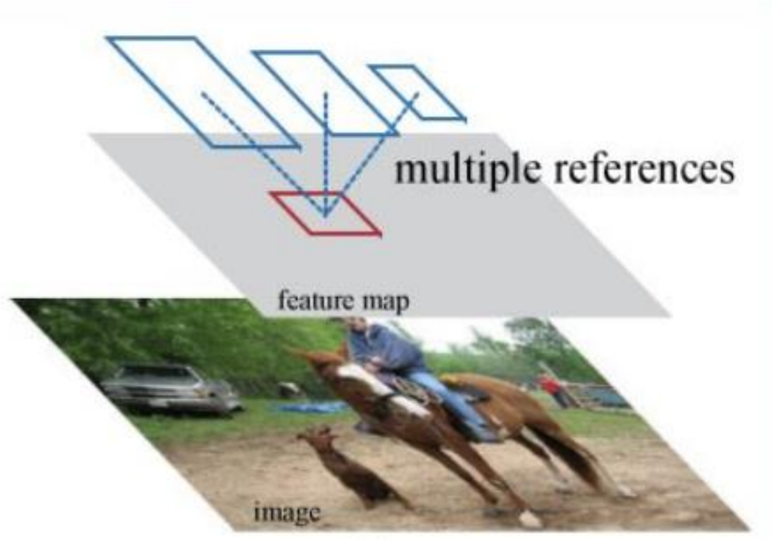


Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

RECOGNITION SYSTEM

- Region Proposal Network
- Input : Atypical Image
- Output : Object Proposal & Objectness Score
- Process : 1. Slides a window through the input
- Process : 2. Refers several anchor boxes in the middle of the window
- Process : 3. Extracts features through an anchor box which has the most IoU score
- About 2K anchor boxes are used



(c)

RECOGNITION SYSTEM

- Classification Network
- Input : Feature vector from feature map
- Output : Classification Score
- Loss : log loss
    - where p is a score vector of K+1 objects
    - where u is a class index

$$L_{\text{cls}}(p, u) = -\log p_u$$

classification loss

RECOGNITION SYSTEM

DATA PREPROCESSING

- Data Centering(Standardization)
  - After we gain the mean values of R,G,B from the whole dataset of COCO
  - Subtract the means from each pixel's value
- Data type conversion
  - From type float32 to int8

RECOGNITION SYSTEM

FALL DETECTION

```
def is_down(box):
    ymin, xmin, ymax, xmax = box
    y_ = ymax - ymin
    x_ = xmax - xmin

    # Multiply WIDTH & HEIGHT to y_ & x_ perspectively
    y_ *= IM_HEIGHT
    x_ *= IM_WIDTH

    # if width is bigger than 1.5x height
    if x_ > (y_*1.5):
        return True
    else:
        return False
```

```
# if fall detected run alarm code
if is_down(box):
    os.system('./detected.sh')
```

RECOGNITION SYSTEM

Pipeline

```
# tensorflow graph file for object detection
PATH_TO_CKPT = os.path.join(CWD_PATH,MODEL_NAME,'frozen_inference_graph.pb')

# label file for classification in object detection
PATH_TO_LABELS = os.path.join(CWD_PATH,'data','mscoco_label_map.pbtxt')

# mapping label’s index to category
# index 1 is implying people
label_map = label_map_util.load_labelmap(PATH_TO_LABELS)
categories = label_map_util.convert_label_map_to_categories(label_map, max_num_classes=NUM_CLASSES,
use_display_name=True)
category_index = label_map_util.create_category_index(categories)
```

RECOGNITION SYSTEM

Pipeline

```
# load the model
detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.compat.v1.GraphDef()
    with tf.io.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
        tf.import_graph_def(od_graph_def, name='')
    sess = tf.compat.v1.Session(graph=detection_graph)
```

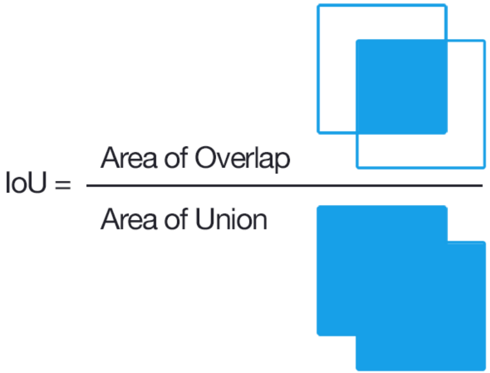
```
# image as input
image_tensor =
detection_graph.get_tensor_by_name('image_tensor:0')

## detection box, score, classes as output
detection_boxes =
detection_graph.get_tensor_by_name('detection_boxes:0')
detection_scores =
detection_graph.get_tensor_by_name('detection_scores:0')
detection_classes =
detection_graph.get_tensor_by_name('detection_classes:0')
```

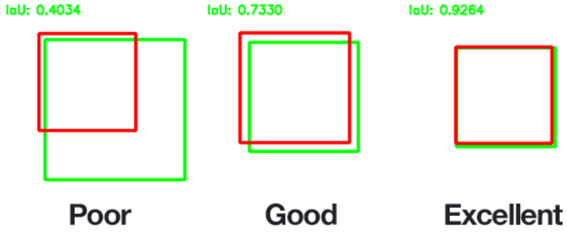
TESTING

- We have checked validation of our model in remote PC
- Validation check metric is based on IoU

```
if IoU >= 0.8:
    it is valid
```

- mAP as total accuracy
  - AP is the area under PR graph
  - we use the mean AP of output as total accuracy





CONCLUSION

HW limitation

- The trained model could not be loaded properly

Scalability

- Rapberry4 allows use models with high performance