

Shoebot : operation manual

@ Date : Spring semester, 2023

@ Part : Automation part in Industrial AI & Automation class

@ Project #2

@ Instructor : prof. Young-Keun Kim

@ github : https://github.com/HanMinung/Robotarm_Automation

Shoebot : operation manual

1. Operation environment
2. Pre-work
3. Robot speed control
4. File download & implementation
5. yolov5_ros.py modification
6. Code implementation
 - 6.1. Implementation sequence

1. Operation environment

python : 3.9

CUDA : 11.3.1

cuDNN : 8.2.1

pyTorch : 1.12.1

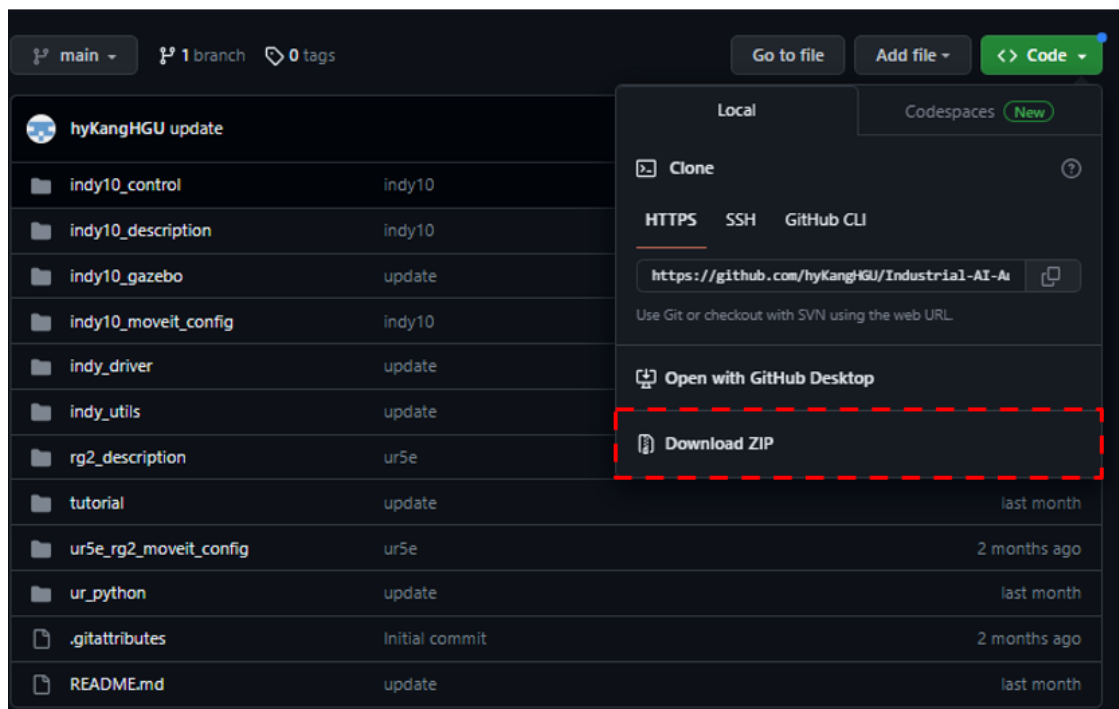
detailed environment setting is specified in other md file : **project description.md** (link : [click here to view](#))

2. Pre-work

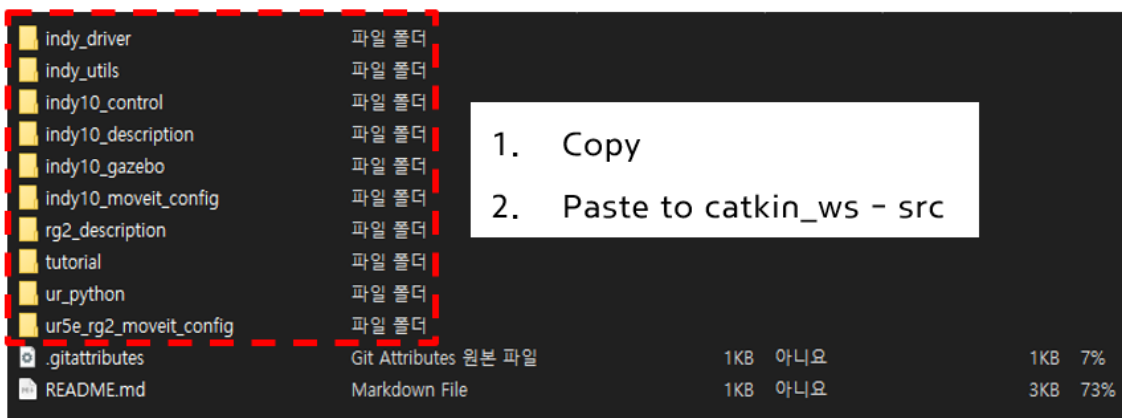
All the packages and codes constructed while conducting the project were managed in catkin_ws path. Therefore, basic settings for catkin_ws are essential first. After that, it will be explained step by step on how to insert the packages.

- First, go to the following link (industrial AI automation: automation part github) and download the Github material as a zip file, as shown in the figure below.

link : [click here to go to github](#)



- Next, in the Ubuntu environment, the home-catkin_ws path is created by default. In that path, there is a folder called 'src' in catkin_ws. If users copy and paste the downloaded file into that path as shown below, the basic setting is completed.



- The compressed file contains various configuration files for operating the indy10 and ur5e robots, so the user can freely construct and operate the code for the project. If you've pasted the file correctly as shown in the figure below, you need to open the terminal window by pressing `ctrl+alt+t` and save the changes by entering the following commands and doing `catkin_make`.

```
# setting path
cd catkin_ws

# build
catkin_make
```

3. Robot speed control

To adjust the driving speed of the robot, user needs to execute the `set_velocity.py` Python file in the terminal first. To set the speed limit for the robot's movement, following parameters need to be modified within the code:

- `indy.set_joint_vel_level`
- `indy.set_task_vel_level`

These parameters can be set to levels 1 to 9, where higher numbers allow for faster movement. The IP address of the Indy 10 robot used is "**192.168.0.18**".

- `set_velocity.py` file description

```
#!/usr/bin/env python3.9
#-*- coding:utf-8 -*-

import indydcpc_client as client
import copy

def main():

    robot_ip = "192.168.0.18"           # 예시 STEP IP 주소
    robot_name = "NRMK-Indy10"         # IndyRP2의 경우
    "NRMK-IndyRP2"
    indy = client.IndyDCPCClient(robot_ip, robot_name) # indy 객체 생성

    indy.connect()

    indy.set_joint_vel_level(9)         # 1 ~ 9
    indy.set_task_vel_level(9)         # 1 ~ 9

    indy.disconnect()                  # 연결 해제

if __name__ == '__main__':
    try:
        main()

    except Exception as e:
        print("[ERROR]", e)
```

Please provide the IP address of the robot in the `robot_ip` field.

Create the above Python code in the following file path. (**catkin_ws - src - indy_utils**)

Grant file permissions to this file

```
# path setting
cd catkin_ws/src/indy_utils

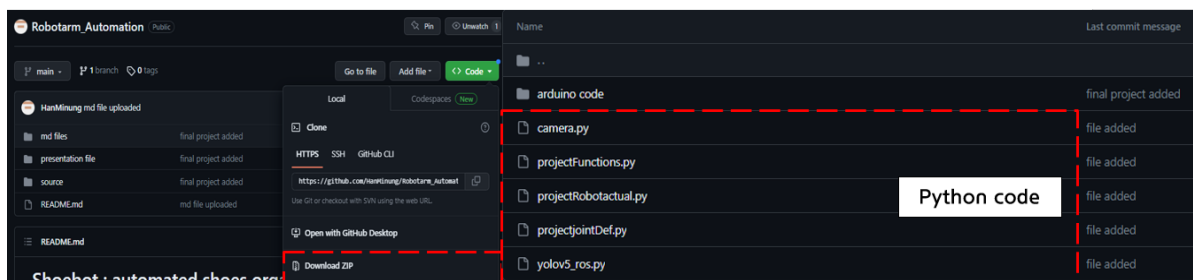
# file permission grant
chmod +x set_velocity.py
```

4. File download & implementation

download link : [click here to download](#)

Go to above link and download zip file.

- best.pt : trained model file with YOLO V5 (to detect shoes)
 - reference : best.pt file is not uploaded with github source since it has too large size to upload.
 - Download link : [click here to download](#)
- Those are with `.py` extension files are main code for manipulation.
- Extract zip file in the path (**indy_driver - src**)
- Arduino process is for just for reference. It is embedded in the UNO board.



module installation

Install modules to run code

- Install psutil

```
pip install --ignore-installed psutil
```

- Reinstall Kiwisolver

```
# remove
sudo rm -r /usr/lib/python3/dist-packages/kiwisolver-1.0.1.egg-info

# install
pip install kiwisolver
```

- Download package

```
# setting path
cd catkin_ws/src

# download
git clone https://github.com/mats-robotics/detection_msgs.git
```

- And please type below commands to terminal for adjustment.

```
# setting path
cd catkin_ws

# build
catkin_make
```

camera.py modification

Check your webcam device number

```
ls -ltr /dev/video*
```

```
mat@mat-ThinkPad-T480: ~
File Edit View Search Terminal Help
~$ ls /dev/video*
/dev/video0 /dev/video1 /dev/video2
~$
```

Enter the confirmed number in camera.py

```
class CameraNode:
    def __init__(self):
        self.bridge = CvBridge() # cv_bridge 객체 생성
        self.image_pub = rospy.Publisher("camera/image_raw", Image, queue_size=1) # "camera/image_raw"라는 토픽으로 메시지를 publish할 publisher 객체 생성
        self.cap = cv2.VideoCapture(2) # 카메라 연결을 위한 VideoCapture 객체 생성

    def inversePerspective(self, image):
        srcPoint = np.float32([[192, 195], [135, 361], [349, 354], [369, 198]])
        dstPoint = np.float32([[0, 0], [0, 300], [400, 300], [400, 0]])

        M = cv2.getPerspectiveTransform(srcPoint, dstPoint)
        warped_image = cv2.warpPerspective(image, M, (image.shape[1], image.shape[0]), flags = cv2.INTER_LINEAR)

        return warped_image
```

5. yolov5_ros.py modification

```
# add yolov5 submodule to path
ROOT = Path("/home/tk032/catkin_ws/src/yolov5") # YOLOv5 root directory
if str(ROOT) not in sys.path:
    sys.path.append(str(ROOT)) # add ROOT to PATH
ROOT = Path(os.path.relpath(ROOT, Path.cwd())) # relative path
```

In the `yolov5_ros.py` code, specify the folder path where your own yolov5 exists.

```
def __init__(self):
    self.conf_thres = rospy.get_param("~confidence_threshold", 0.7)
    self.iou_thres = rospy.get_param("~iou_threshold", 0.8)
    self.agnostic_nms = rospy.get_param("~agnostic_nms", False)
    self.max_det = rospy.get_param("~maximum_detections", 5)
    self.classes = rospy.get_param("~classes", None)
    self.line_thickness = rospy.get_param("~line_thickness", 1)
    self.view_image = rospy.get_param("~view_image", True)
    # Initialize weights
    self.weights = rospy.get_param("~weights", "/home/tk032/catkin_ws/src/yolov5/runs/train/Slippers/weights/best.pt")
    # Initialize model
    self.device = select_device(str(rospy.get_param("~device", "")))
    self.model = DetectMultiBackend(weights=self.weights, device=self.device, dnn=rospy.get_param("~dnn", True), data=rospy.get_param("~data", "/home/tk032/catkin_ws/"))
    self.stride, self.names, self.pt, self.jit, self.onnx, self.engine = []
    self.model.stride,
    self.model.names,
    self.model.pt,
    self.model.jit,
    self.model.onnx,
    self.model.engine,
```

Change the path of the weights to the path where you have downloaded the `best.pt` file.

Change the path of data to the path where you have downloaded the `data.yaml` file.

6. Code implementation

Prepare five terminals and enter the following commands in each of them.

- Terminal 1

```
# setting path
cd catkin_ws/src/indy_utils

# setting velocity limitation
python3.9 set_velocity.py
```

- Terminal 2

```
# setting path
cd catkin_ws

# robot connection
source devel/setup.bash
roslaunch indy10_moveit_config moveit_planning_execution.launch
robot_ip:=192.168.0.18
```

- Terminal 3

```
# setting path
cd catkin_ws

# camera process implementation
source devel/setup.bash
roslaunch indy_driver camera.py
```

- Terminal 4

```
# setting path
cd catkin_ws

# main process implementation
source devel/setup.bash
roslaunch indy_driver projectRobotactual.py
```

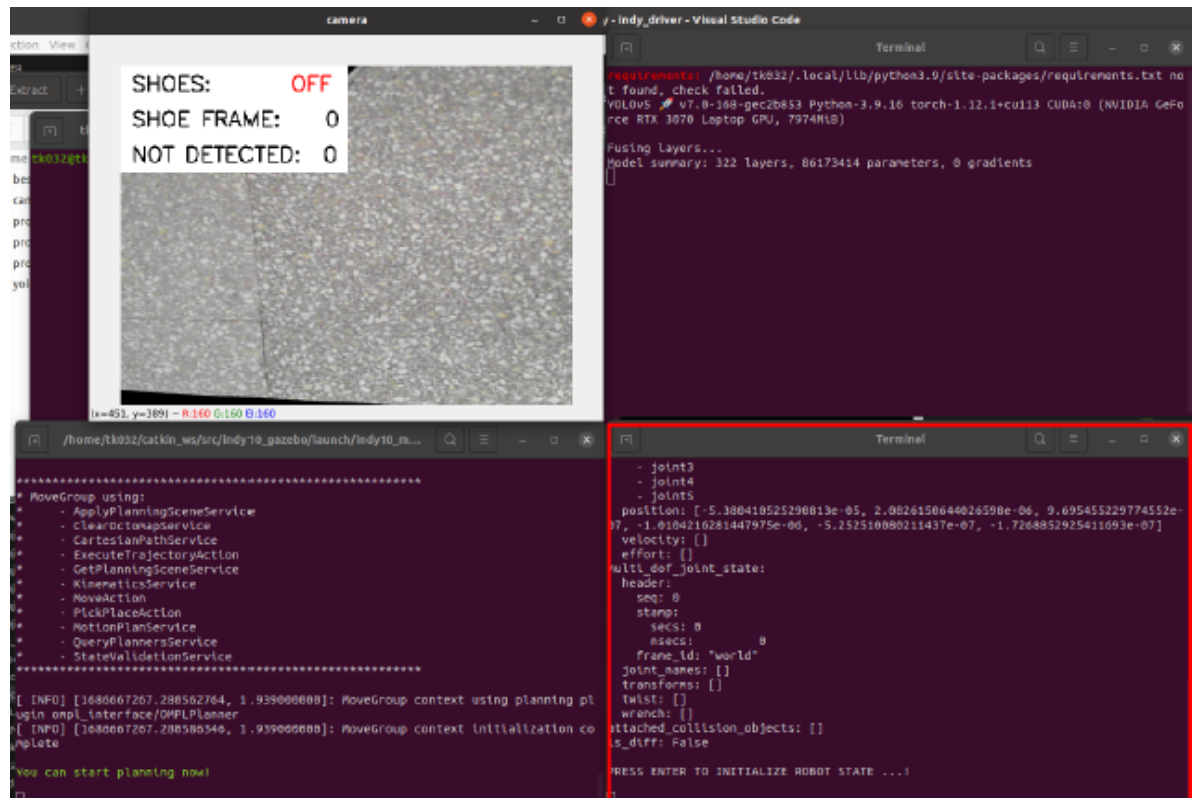
- Terminal 5

```
# setting path
cd catkin_ws

# deep learning object detection process implementation
source devel/setup.bash
roslaunch indy_driver yolov5_ros.py
```

Execute each terminal command in order.

The progress of the program will be carried out in Terminal 4, which corresponds to the red-boxed terminal below.



All the steps mentioned above can be managed together using a bash shell script. The instructions for creating the bash script and executing all the steps at once are provided in another Markdown file named `project_description.md`.

6.1. Implementation sequence

[Caution]

- When the LED is **red**, it means the robot is in motion, so do not place or remove the shoes and do not enter any commands in the terminal.
- Only when the LED is **green**, enter commands in the terminal and place or remove the shoes.

1. Robot state initialization

- Press enter to initialize robot state when program is initialized.

```
Terminal
- joint3
- joint4
- joint5
position: [0.027240440339420392, -0.25287546513024733, 2.4155161661626345, 0.0
13339272909507668, -0.7139304161579395, -0.06885684738988238]
velocity: []
effort: []
multi_dof_joint_state:
header:
seq: 0
stamp:
secs: 0
nsecs: 0
frame_id: "world"
joint_names: []
transforms: []
twist: []
wrench: []
attached_collision_objects: []
is_diff: False

PRESS ENTER TO INITIALIZE ROBOT STATE ...!
```

1. Mode selection

- Press 1 : shoes pickin mode
- Press 2 : shoes pickout mode
- If '1' is selected, empty cabinet is automatically assigned and the robot manipulation is started and the LED turns on with red light.

```
Terminal
frame_id: "world"
joint_names: []
transforms: []
twist: []
wrench: []
attached_collision_objects: []
is_diff: False
INITIALIZATION COMPLETED ...!

-----
CABINET STATE
-----
CABINET 0 : 1 | CABINET 3 : 0
CABINET 1 : 0 | CABINET 4 : 0
CABINET 2 : 0 | CABINET 5 : 0
-----

MODE 1 : PICK IN (PRESS '1')
MODE 2 : PICK OUT (PRESS '2')

-----
PLEASE INPUT MODE WHAT YOU WANT ...!

1
CABINET NUMBER '1' IS ASSIGNED ...!
```

Mode selection

Cabinet assign

1. Shoes detection


```
Terminal
multi_dof_joint_state:
  header:
    seq: 0
    stamp:
      secs: 0
      nsecs: 0
    frame_id: "world"
  joint_names: []
  transforms: []
  twist: []
  wrench: []
attached_collision_objects: []
is_diff: False

----- CABINET STATE -----
| CABINET 0 : 1 | CABINET 3 : 0 |
| CABINET 1 : 0 | CABINET 4 : 0 |
| CABINET 2 : 0 | CABINET 5 : 0 |
-----

WAITING FOR CLIENT...!
WAITING FOR CLIENT...!
WAITING FOR CLIENT...!
WAITING FOR CLIENT...!
```

- Bring an empty plate and wait for the client to place the shoes completely. During the waiting period, the message "waiting for client" will be displayed. Once the shoes are placed, it will be detected, and the plate with the shoes will be placed back.

1. Pick out mode

```
Terminal
| CABINET 2 : 0 | CABINET 5 : 0 |
-----

| MODE 1 : PICK IN (PRESS '1') |
| MODE 2 : PICK OUT (PRESS '2') |
|                               |
-----

PLEASE INPUT MODE WHAT YOU WANT ...!
2
-----

CANINET '0' IS AVAILABLE TO PICK OUT...!
----- CABINET STATE -----
| CABINET 0 : 1 | CABINET 3 : 0 |
| CABINET 1 : 0 | CABINET 4 : 0 |
| CABINET 2 : 0 | CABINET 5 : 0 |
-----

PLEASE INPUT CABINET # NUMBER THAT YOU WANT TO PICK OUT...!
```

Once the final operation is completed, the process of selecting the mode is repeated. If option 2 (Shoe Retrieval Mode) is selected, the user will be prompted to choose which cabinet to retrieve shoes from.

In contrast to the Shoe Placement Mode, in this case, the user will input the index of the cabinet from which to retrieve shoes. The system will then detect the disappearance of the shoes from the camera's view.