Connecting with Rasberry Pi

In user terminal :

Scp – r “path of file to be sent” pi@address_of_pi:~/

Eg.: >scp -r "C:\Users\NHI658\Desktop\Brane\Try2.zip" pi@10.11.2.10:~/

Password : 3.14

In Rasberry Pi :

Ifconfig – to know ip address

To accelerate speed in rpi : use intel ncu stick

->JetSon Nano supports GPU

Connecting with SSH (local host)

Ssh –X User@IP_Address

Eg.: ssh -X rohitpandey@10.11.9.185

To create a virtual environment : conda create --name <myenv>

To list the available virtual environments : conda env list

To activate the virtual environment : conda activate env_name

To deactivate an active environment : conda deactivate

To view the libraries with the versions present : pip3 freeze

To delete a virtual environment : conda remove --name ENV_NAME --all

Working with robotic arm

Library required : ikpy

Detection models :

- one-stage and single-stage target detection : wholly completed by one network through inputting images and outputting bounding box and classification label
- two-stage target detection : detection is completed through two different networks through generating recommended regions by inputting images, and then sending them to a classifier for classification. Eg.: R-CNN, Fast R-CNN, Faster R-CNN

R-CNN

Steps :

1. conducts sparse sampling on the input original image by using candidate regions
2. extracts features from the candidate regions by CNN
3. classifies the extracted information by SVM (support vector machine)

Cons :

- the network separately extracts features from a large number of candidate regions, which leads to a large number of repeated calculations conducted by the network, resulting in a lot of redundant calculation information and higher calculation cost

Fast R-CNN

- uses multi-task loss to replace the support vector machine (SVM) for classification so that the regression of frame can also be trained through the network, and thereby classification and frame regression can be trained through the same network

Cons :

- the method of generating candidate region based on traditional methods needs to be processed on CPU, which limits the running speed of the system to a certain extent.

PyBullet

- PyBullet is an Application Programming Interface (API) that allows you to have quick access to robot simulation using the Bullet Physics SDK simulation engine.
- It is a quick tool for research and training.
- Robot simulation software like ROS2 or PyBullet needs to read robot files in certain file formats and one of these formats is called URDF (Unified Robot Description Format) which defines the physical properties of a robot.
- KDL (kinematics dynamic libraries) are present for the kinematics equations
- TCP : Tool Center Point

URDF(unified robot description format) Files

- Similar to xml files
- Used to specify the environment specifications of a robot like the axis color, background settings, light conditions, camera specifications, etc.
- SDF files give more specific environment settings
- Urdf has one root tag called the robot which has 1 attribute I.e., name

- Joints : define the relationship between the origin or the coordinate frames of the links which defines the position & rotation of each link in space
- Each link has a corresponding joint

- Types of joints :
    - Revolute : has fixed limits, Eg.:arm
    - Continuous : no fixed limits, Eg.: wheel, spinning gripper
    - Prismatic : moves along a linear path, Eg.: linear actuater
    - Fixed

<?xml version=”1.0”?>

<robot name = “robot”>

<link name=”link”>

<visual>

<geometry> //overall shape is specified

//specify the shape or give path to the 3d mesh

<origin>

<material> //set colour

//rgb values or reference to name already being set before

</visual>

<collision> //used for physics collision calculations

<geometry>

<origin>

</collisoin>

<inertial> //used for physics calculations

<mass>

<origin>

<inertia>

</inertial>

</link>

<join name=”join” type=”specify_type”>

<parent link=”link”/>

<child link=”link”/>

<origin xyz=” ” rpy=”0 0 0”/>

//above 3 enough for a fixed joint

<axis xyz = “0 -1 0”/> //specifies about which axis the arm moves

<limit lower = “0” upper=”1” velocity = “100” effort = “100”/>

</join>

//-y : upward rotation

References

urdf : https://articulatedrobotics.xyz/ready-for-ros-7-urdf/

Urdf visualization : https://gkjohnson.github.io/urdf-loaders/javascript/example/bundle/index.html

GrabCut Algorithm

- GrabCut cuts the image to obtain the best configuration, and continuously iterates and optimizes the results.
- It is not based on image morphology, but on the graph cut theory.
- It is more suitable for static images
- It is mainly used for image segmentation
- It alone cannot detect the % cooked and other things, it needs the help of other algorithms
- `grabCut(img, mask, rect, bgdModel, fgdModel, iterCount[, mode]) -> mask, bgdModel, fgdModel`

img

: The input image, which GrabCut assumes to be an 8-bit, 3-channel image (i.e., unsigned 8-bit integer in BGR channel ordering).

mask

: The input/output mask. This mask is assumed to be a single-channel image with an unsigned 8-bit integer data type. This mask is initialized automatically if you use bounding box initialization (i.e.,

cv2.GC_INIT_WITH_RECT

); otherwise, GrabCut assumes you are performing mask initialization (

cv2.GC_INIT_WITH_MASK

).

rect

: The bounding box rectangle that contains the region that we want to segment. This parameter is only used when you set the

mode

to

cv2.GC_INIT_WITH_MASK

).

bgModel

: Temporary array used by GrabCut internally when modeling the background.

fgModel

: Temporary array used by GrabCut when modeling the foreground.

iterCount

: Number of iterations GrabCut will perform when modeling the foreground versus background. The more iterations, the longer GrabCut will run, and ideally the results will be better.

mode

: Either

cv2.GC_INIT_WITH_RECT

or

cv2.GC_INIT_WITH_MASK

, depending on whether you are initializing GrabCut with a bounding box or a mask, respectively.

OpenCV's GrabCut implementation returns a 3-tuple of:

mask

: The output mask after applying GrabCut

bgModel

: The temporary array used to model the background (you can ignore this value)

fgModel

: The temporary array for the foreground (again, you can ignore this value)

Aspect	Machine Vision	Computer Vision
Definition	Concerned with industrial applications and the computer's ability to "see" in specific tasks.	Refers to any technology in which a computer digitizes an image, processes its data, and takes action.
Processing Power	Typically has less processing power and used in lean manufacturing for high-speed data acquisition.	Aim to fully understand objects or scenes, collecting general, transferable information.
Data Collection	Focuses on practical tasks to complete specific jobs efficiently.	Collects as much data as possible about objects or scenes for broader applications.
Source of Images	Primarily used in manufacturing environments.	Can process images from any source, including the internet.
Application Scope	Limited to specific industrial tasks.	Versatile and applicable to various tasks.

Pybotics :

- It is an open-source Python toolbox for robot kinematics and calibration
- The toolbox is specifically designed for use with the Modified Denavit–Hartenberg parameters convention
-

References :

- https://pybotics.readthedocs.io/en/latest/
- https://github.com/engnadeau/pybotics/blob/master/examples/kinematics.ipynb