Manual

# Real world environment setup

Step 1. Print 2 AR marker2.png in multiple\_grasping\_pose\_learning/utils and put them on the shelf to set up a rectangle workspace



Step 2. Make a simple square turntable, here we use a 18cm x 18cm instant ramen bowl, if you use other sizes, you may need to modify the codes in multiple\_grasping\_pose\_learning /euslisp/collect\_data\_turntable.l. (Adjust the translate values in the first\_push() and second\_push() functions)





Remember to put a white paper on the top of the turntable to remove background effects

# Build ROS Package and collect data

## 2.1 Build ROS package

Environment: Ubuntu18.04 and ROS Melodic

1. cd ~/aerov\_grasp\_ws/src
2. git clone <https://github.com/himlen1990/multiple-grasping-pose-learning.git>
3. catkin build

If nothing goes wrong. We can start collect data

## Data collection

Put the turntable on the shelf within the workspace and adjust its pose



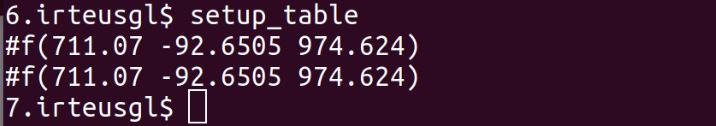
Open a terminal and connect to the aero robot

1. roslaunch multiple\_grasping\_pose\_learning\_data\_collection\_turntable.launch

\*if you are not collect data from the very beginning, modify the “start\_obj\_num” value in the launch file

Open another terminal

1. roscd multiple\_grasping\_pose\_learning/euslisp/
2. roseus collect\_data\_turntable.l
3. In the roseus prompt, type setup\_table, then it should return the table’s coordinate

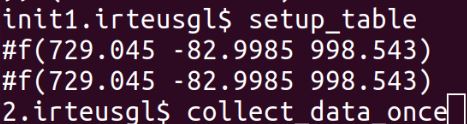


\*(if the output is #f(), please check the robot camera image to see whether the ar markers are visible)

1. Put the target object on the turn table (do not put the object on the turntable before running the setup\_table function because this function computes the table’s height)

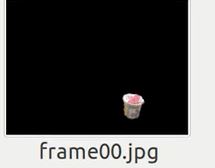


1. In the roseus prompt, type collect\_data\_once, then the robot will collect data automatically. The robot pushing trajectory is approximately shown as follows:



This process will be executed twice (the object will be rotated 180 degree), the images are stored in dataset/rgb and dataset/mask. The mask output should like this (only contains the target object)



\*(sometimes the aero robot may fail to solve the IK and can not reach the desired position. Instead, the robot may reach an approximate position and lead to failures, you may need to adjust the robot’s location or even adjust the parameters in the collect\_data\_turntable.l)

1. Open another terminal to publish a topic to add a new object category

rostopic pub -1 /aero\_data\_collection std\_msgs/String "add new"

(if you want to recollect the data, you can publish a command “do again”)

\*rostopic pub -1 /aero\_data\_collection std\_msgs/String "do again"

1. After the process is finished, adjust the turntable’s pose then run setup\_table again (as a remind: do not put anything on the table when running this function), and repeat step 1-4 until you finish the data collection process.

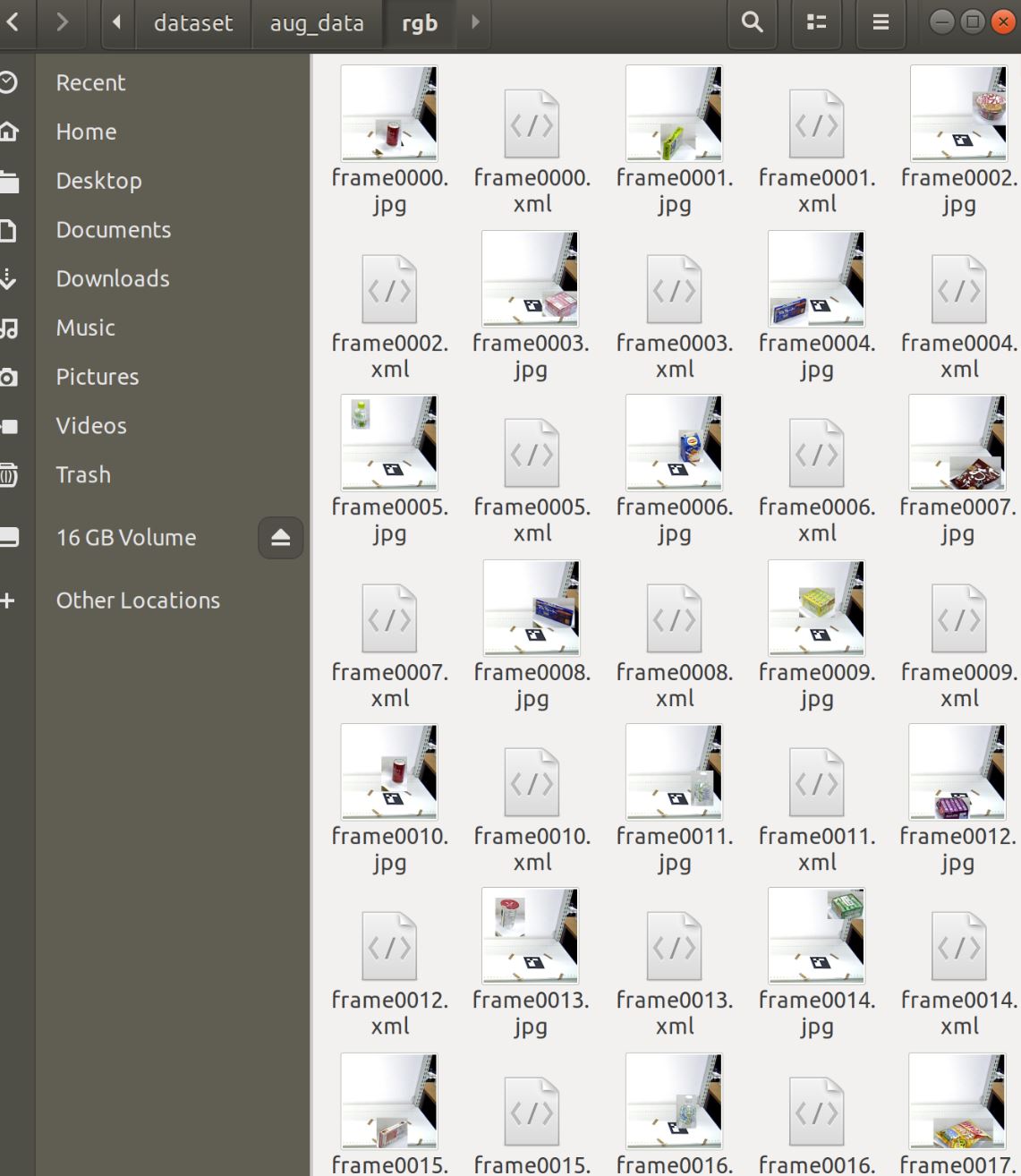
# Data augmentation and training

## 3.1 data augmentation

1. roscd multiple\_grasping\_pose\_learning/utils/
2. python data\_augmentation\_bbox\_turntable.py

\*if there are any import errors, you may need to install the corresponding packages via pip install

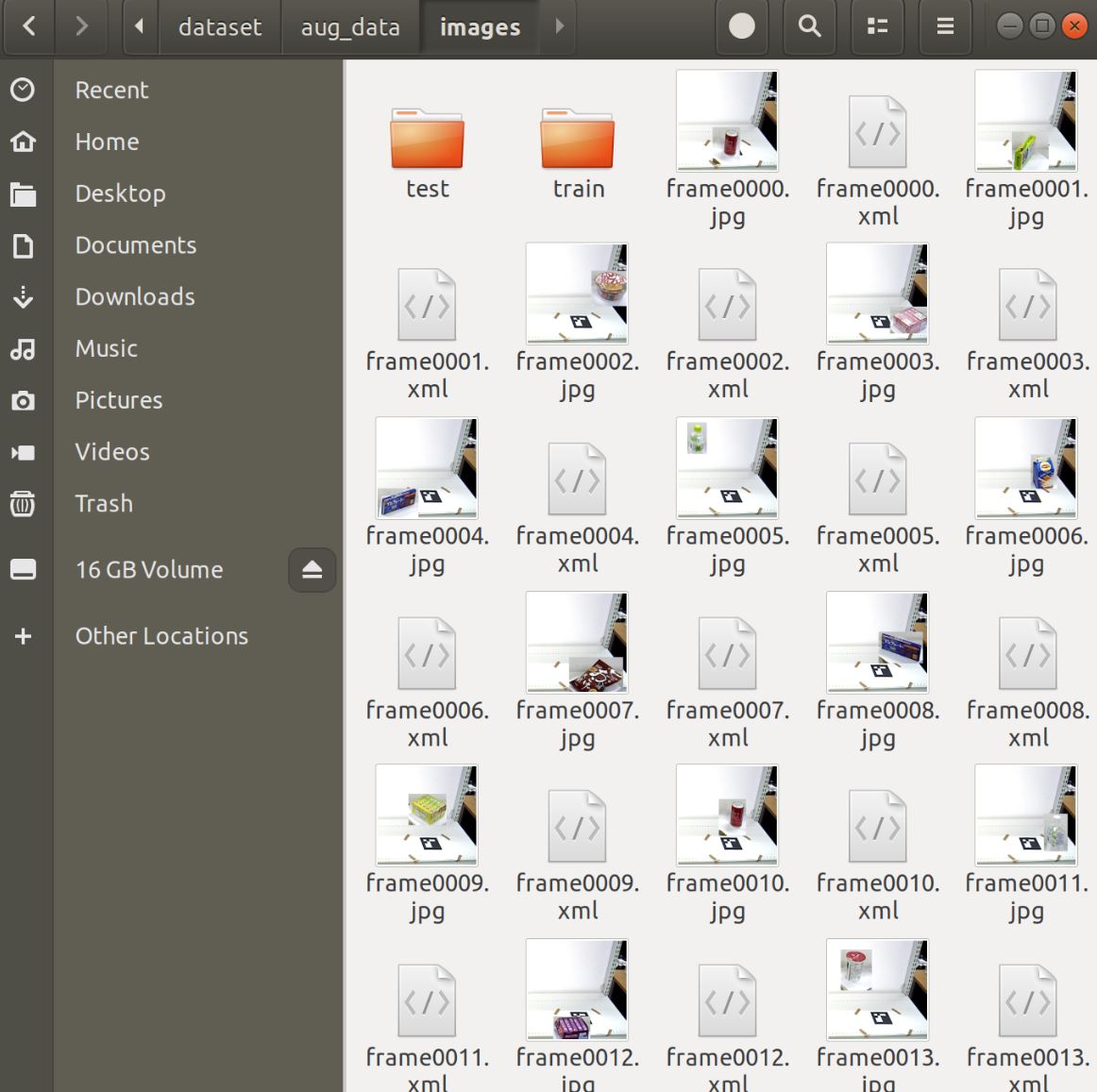
Then check images in the dataset/aug\_data/rgb/ folder, it should look like this



then partition the dataset by running

1. sh partition.sh

it will create a train and a test folder under the rgb folder



## \*3.2 Docker environment setup

install docker cuda image

1. docker pull nvidia/cuda:10.0-devel-ubuntu18.04

create a docker container

1. docker run --gpus all -it nvidia/cuda:10.0-devel-ubuntu18.04

in the container

1. cd
2. apt-get update
3. apt-get install python3.6 python3-pip git wget zip
4. apt-get install –y libgl1-mesa-dev
5. pip3 install --upgrade pip
6. pip3 install Cython pandas scipy==1.5.4 tf-slim==1.1.0 matplotlib==3.3.4 six==1.16.0 opencv-python==4.5.2.52 tensorflow-gpu==1.15.3
7. mkdir Tensorflow
8. cd Tensorflow
9. git clone <https://github.com/tensorflow/models.git>
10. cd Tensorflow/models/research/
11. wget <https://github.com/protocolbuffers/protobuf/releases/download/v3.17.3/protoc-3.17.3-linux-x86_64.zip>

\*This address may change, please check the tutorial on <https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/install.html> and the proto release page

<https://github.com/protocolbuffers/protobuf/releases> , you can get the address by finding out the protoc such as "protoc-3.17.3-linux-x86\_64.zip" , then right click and copy link address.

1. unzip protoc-3.17.3-linux-x86\_64.zip
2. bin/protoc object\_detection/protos/\*.proto --python\_out=.
3. cp object\_detection/packages/tf1/setup.py .
4. python3 setup.py install

## 3.3 Training

1. cd ~/Tensorflow/

Copy the multiple\_grasping\_pose\_learning/utils/demo folder to the docker container, under ~/Tensorflow/

Copy the train and test folder (create in 3.1) to ~/Tensorflow/demo/images/

Then

1. cd ~/Tensorflow/demo/pre-trained-models
2. wget <http://download.tensorflow.org/models/object_detection/ssd_mobilenet_v2_quantized_300x300_coco_2019_01_03.tar.gz>
3. tar -xvzf ssd\_mobilenet\_v2\_quantized\_300x300\_coco\_2019\_01\_03.tar.gz
4. cd ~/Tensorflow/demo
5. sh generate\_record.sh
6. sh train.sh

\*when you want to train on a new dataset, remember to delete all files except pipeline.config in the demo/models/my\_ssd folder

Since the default number of object categories is 30, if you want to train your dataset, you need to modify the label\_map.pbtxt file in /demo/annotations/ and the pipeline.config file in demo/models/my\_ssd to fit your dataset

## Convert the training results to tflite

Install edgetpu\_complier

1. curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
2. echo "deb https://packages.cloud.google.com/apt coral-edgetpu-stable main" | sudo tee /etc/apt/sources.list.d/coral-edgetpu.list
3. sudo apt-get update
4. sudo apt-get install edgetpu-compiler

convert to tflite

1. cd ~/Tensorflow/demo/models
2. sh convert.sh

# Deploy