ELEG4701: Intelligent Interactive Robot Practice

Lecture 7: Introduction to Sensors

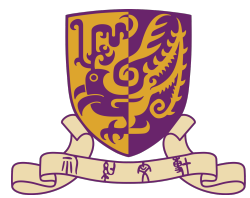
Jiewen Lai

Research Assistant Professor

jwlai@ee.cuhk.edu.hk

EE, CUHK

2024 Spring



Outline

Part 1. Introduction to Depth Camera

Differences between RGB-D and RGB image

Figure: RGB image

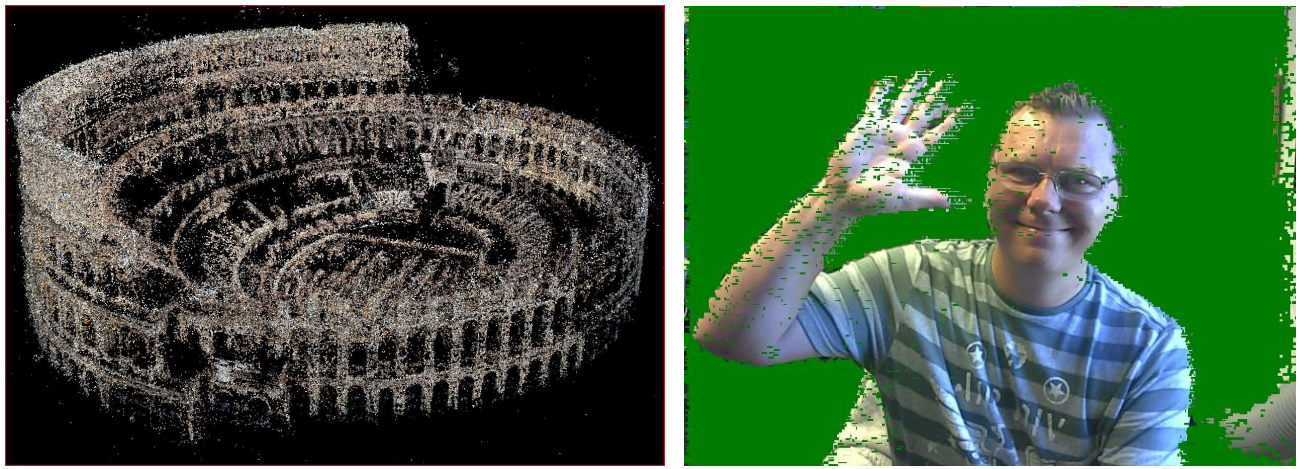


Figure: RGB-D (Point Cloud)

Introduction to Depth Camera

- A RGB camera provides an RGB (red-green-blue) two-dimensional image of the three-dimensional world.
- The RGB values at each pixel location depend on many factors:
  - Light source
  - Object color
  - Sensor capabilities

How do we infer 3D information from a single RGB image?

Figure: This image looks like "3D", but it's actually an RGB image, which is 2D.

Types of depth camera

Stereo vision

- Multiple-camera system
- Just like humans can tell the distance by comparing the differences between the image of our two eyes.

Figure: ZED cameras[1] based on stereo vision.

[1] https://www.stereolabs.com/

Types of depth camera

3D Sensor Technologies

Sensors that can measure depth (or distance, or range) to an object based on illuminating the object with a laser (or other kinds of light source), and measuring the backscattered light.

- Projected-light sensors that combine the projection of a light pattern with a standard 2D camera and that measure depth via triangulation.
- Time-of-flight sensors that measure depth by estimating the time delay from light emission to light detection.

Figure: The Kinect series cameras

Introduction to Depth Camera

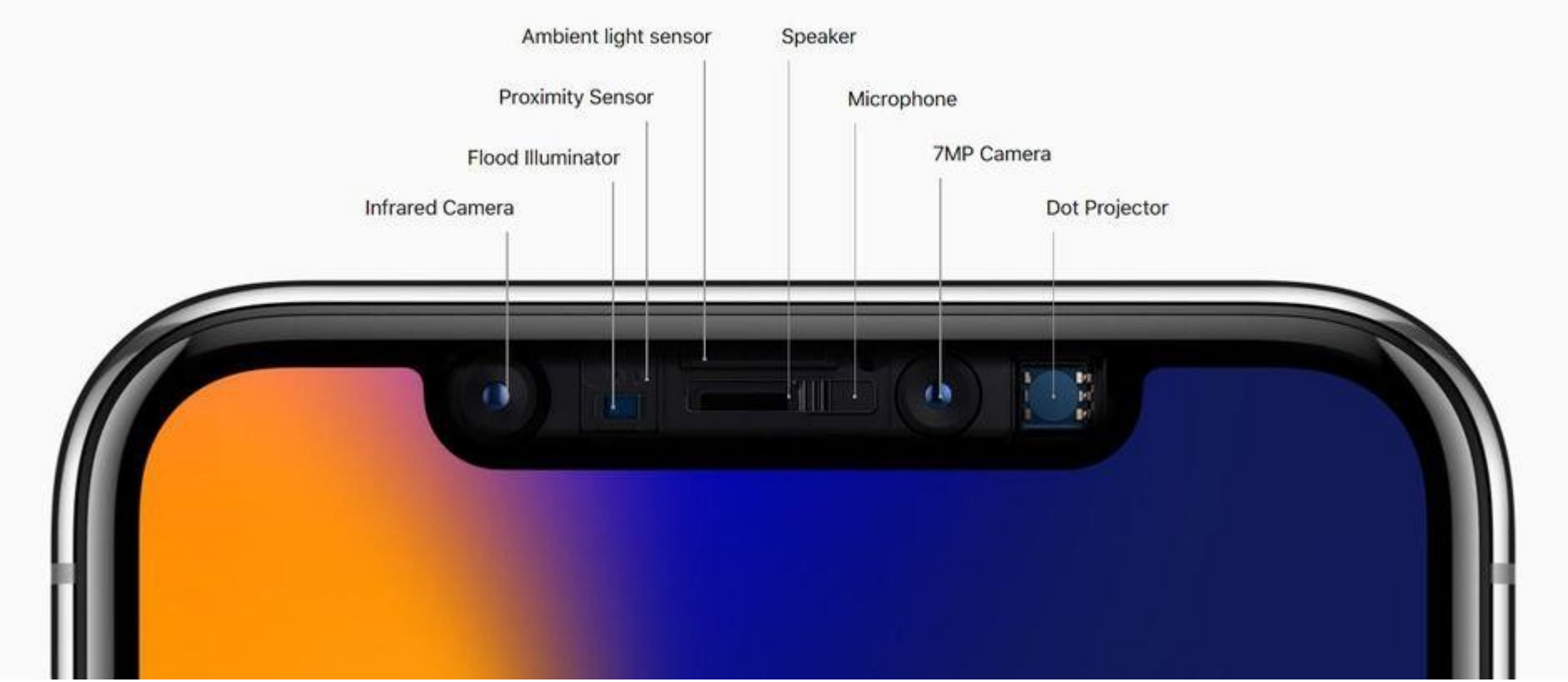


Figure: Apple's TrueDepth camera since iPhone X (2017) - enabling Face ID.

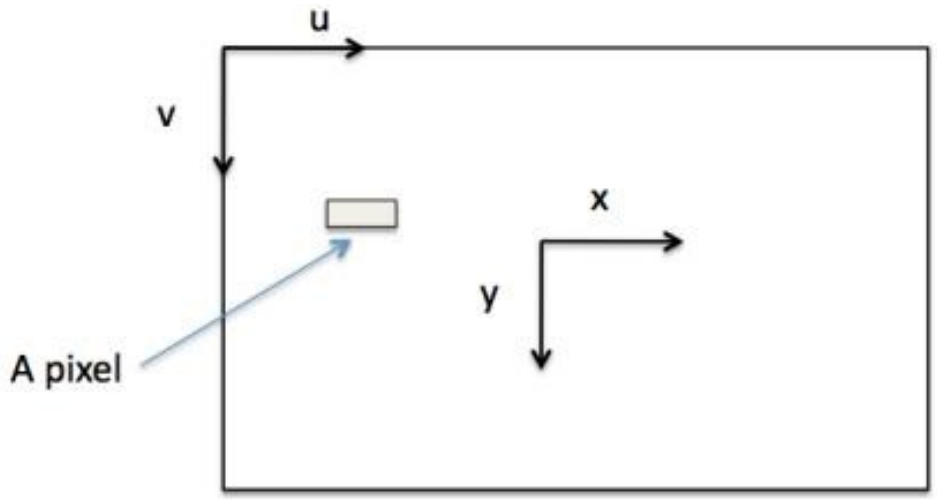
Data from RGB-D camera

RGB camera: 3 Integers

- uint8 red
- uint8 green
- uint8 blue

RGB-D camera: 4 Integers

- uint8 red
- uint8 green
- uint8 blue
- uint8 depth



Part 2. Introduction to Light Detection and Ranging (Lidar) Sensors

Introduction to Lidar Sensors

Light detection and ranging, or **lidar**, is a remote-sensing technology that uses **pulsed laser energy** (light) to measure ranges (distance).

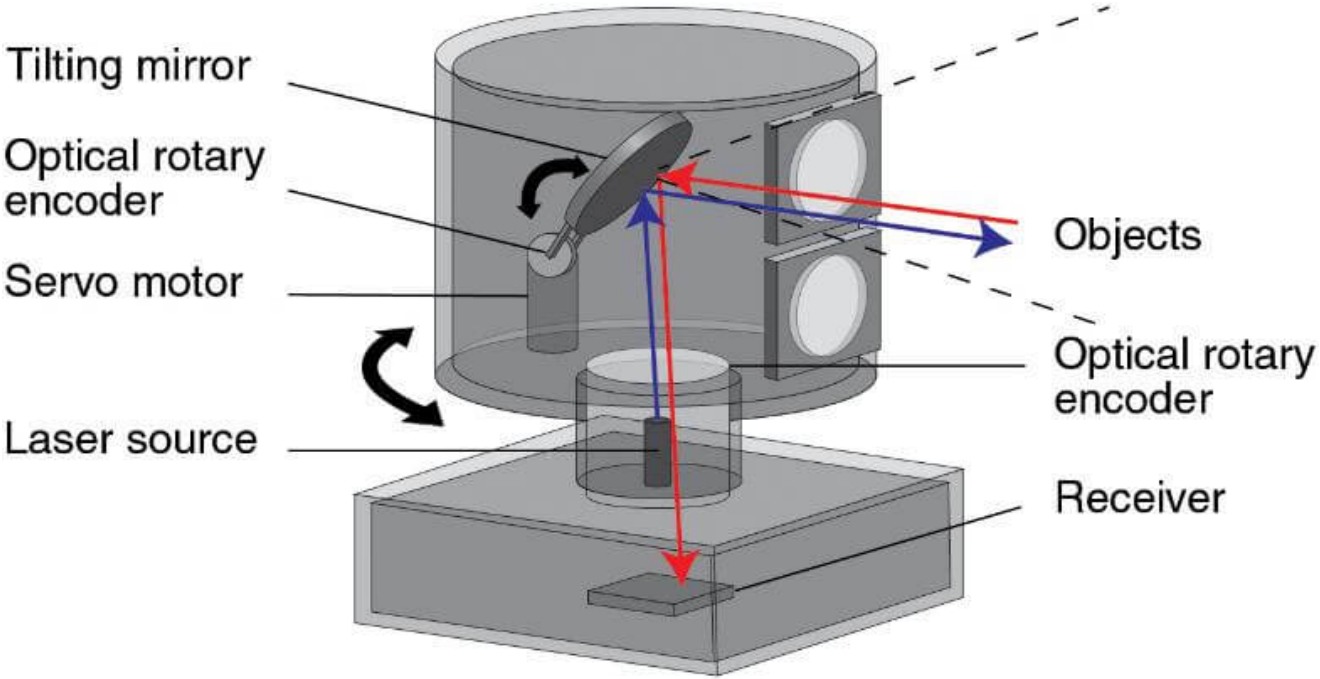
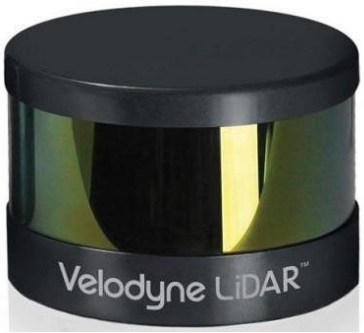


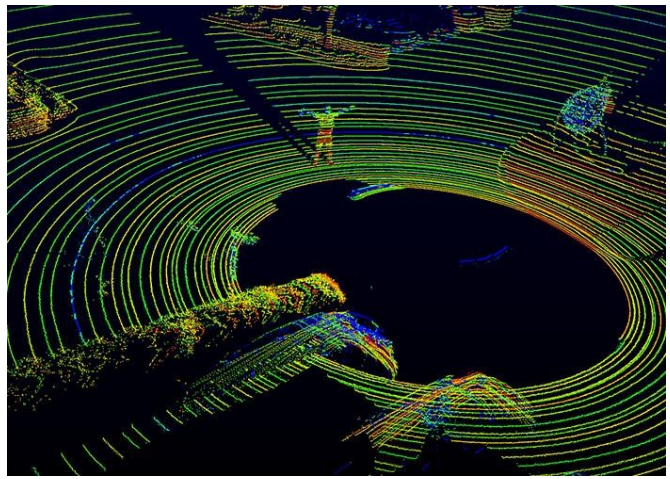
Figure: How lidar works?

Categories of Lidar

2D Lidar

3D Lidar





Part 3. Task I: Preparation

Task I: Preparation

1. Download the package 'lab7' from your Blackboard
2. Copy/paste the downloaded package to '<your_workspace>/src'
3. Build the whole workspace
4. Check the downloaded package and rosbag

What is a 'rosbag':

'rosbag' is a set of tools for recording from and playing back to ROS topics.

Use:

Terminal

```
rosbag play recorded1.bag
```

In our tutorial, we have included the command in the '.launch' file.

Part 4. Task II: Subscribe and save the color (RGB) image

Task II: Subscribe and save the RGB image

Programming

```
$ cd ~/catkin_ws/src/lab7/scripts
# Modify the task_II.py
$ chmod +x task_II.py
```

Terminal 1

```
$ roscore
$ cd ~/catkin_ws
$ source devel/setup.bash
$ rosrun lab7 task_II.py
```

Terminal 2

```
$ source devel/setup.bash
$ roslaunch lab7 task_II.launch
```

Task II: Subscribe and save the RGB image

opencv-python – application programming interface (API)

Saves an image to a specified file. The function 'imwrite' saves the image to the specified file. The image format is chosen based on the filename extension.

Python Code

```
cv2.imwrite(file, img[, params]) -> retval
```

Example:

```python
import cv2

# read image as grey scale
grey_img = cv2.imread('/home/img/bananaCat.png', cv2.IMREAD_GRAYSCALE)
# save image
fileNAME = 'home/sam/catkin_ws/src/lab7/imgs/bananaCat.png'
status = cv2.imwrite(fileNAME, grey_img)
print('Image written to the file-system: ', status)
```

Task II: Subscribe and save the RGB image

Some useful information

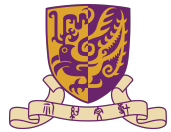
Python: Init the ROS node

```
rospy.init_node(<node_name>)
```

Python: Subscribe the topic:

```
rospy.Subscriber(<topic_name>, <msg_data_type>, ...
... <callback_function_name>)
```

The suggested path for saving the image is '~/catkin_ws/src/lab7/imgs'.



Part 5. Task III: Subscribe and save all the RGB and Depth images

Task III: Sub & Save all the RGB and Depth Images

- In Task II, we only save one RGB image with the Python file, because in each callback function, the newer one will overwrite the older one.
- But the 'rosbag' contains two sequences of images.
- One sequence for the color (RGB) image, and the other for the depth image.
- Find your ways to save all the images in both sequences.
- Considering giving each image file a unique name, then they will not overwrite the older file.

Task III: Sub & Save all the RGB and Depth Images

Programming

```
$ cd ~/catkin_ws/src/lab7/scripts
# Modify the task_III.py
$ chmod +x task_III.py
```

Terminal 1

```
$ roscore
$ cd ~/catkin_ws
$ source devel/setup.bash
$ rosrun lab7 task_III.py
```

Terminal 2

```
$ source devel/setup.bash
$ roslaunch lab7 task_III.launch
```

Task III: Sub & Save all the RGB and Depth Images

Some useful information

Py: Use two lines for subscribe and two callback functions.

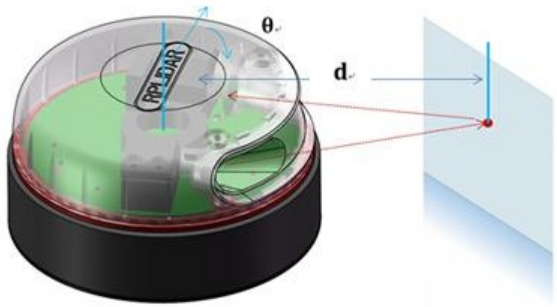
```
def callbackFunction_I(msg):
    ...
def callbackFunction_II(msg):
    ...
def main():
    ...
    rospy.Subscriber(<topic_name_I>, <msg_data_type_I>, <callbackFunction_I>)
    rospy.Subscriber(<topic_name_II>, <msg_data_type_II>, <callbackFunction_II>)
    ...
```

Color img topic name: `/camera/color/image_raw`
ROS data type: `sensor_msgs/Image`
Related cv2 data type: `bgr8`

Depth image topic name: `/camera/depth/image_rect_raw`
ROS data type: `sensor_msgs/Image`
Related cv2 data type: `16UC1`

Part 6. Task IV: Calculate average distance to the obstacles

Task IV: Calculate average distance to the obstacles



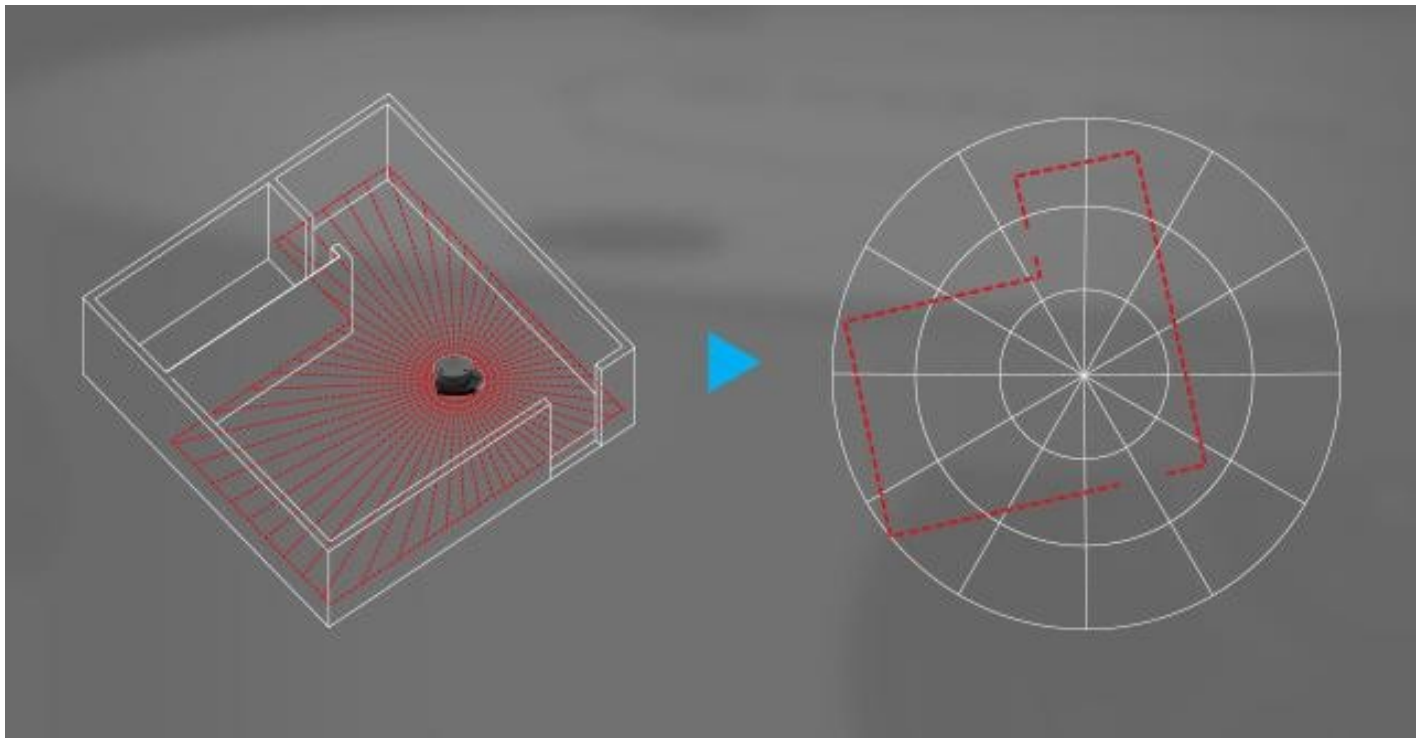


Figure: Example

Task IV: Calculate average distance to the obstacles

Programming

```
$ cd ~/catkin_ws/src/lab7/scripts
# Modify the task_III.py
$ chmod +x task_IV.py
```

Terminal 1

```
$ roscore
$ cd ~/catkin_ws
$ source devel/setup.bash
$ rosrun lab7 task_IV.py
```

Terminal 2

```
$ source devel/setup.bash
$ roslaunch lab7 task_IV.launch
```

Task IV: Calculate average distance to the obstacles

Some useful information Topic name: `/scan`
Data type[2]: `sensor_msgs/LaserScan.msg`

- You only need to consider the '`float32[] ranges`'
- Sum of a vector in python: '`sum(<vector>)`'
- Length of a vector in Python: '`len (<vector>)`'
- Calculate the average distance in Python: '`sum(<vector>)/len(<vector>)`'

.msg

```
std_msgs/Header header
float32 angle_min
float32 angle_max
float32 angle_increment
float32 time_increment
float32 scan_time
float32 range_min
float32 range_max
float32[] ranges
float32[] intensities
```

[2] (http://docs.ros.org/en/noetic/api/sensor_msgs/html/msg/LaserScan.html)

Thanks for listening!