

Motion Capture and Future Interaction Technology Research

The Core Functionality: Principles
and Process

Speaker: Fu-Song Hsu

Outline

- Human Pose Estimation
- RGBD Camera
- Functions in RGBD Camera
- Application: Background Subtraction

Introduction: Pose Estimation



Source: <https://www.youtube.com/watch?v=NnzzSkKKoa8>

3D Pose Estimation

- How to reconstruct 3D Pose Estimation from a RGBD camera.
- RGBD cameras (Intel RealSense/Microsoft Kinect) are the popular solution.

Intel RealSense 3D Camera



Functions in Intel RealSense SDK

intel.
REALSENSE™

PRODUCTS DEVELOPERS USE CASES BLOG SUPPORT BUY

SEARCH USER CART

Download Code samples Documentation GitHub Tools Community

D400 L500 T200 SR300

Intel RealSense SDK 2.0

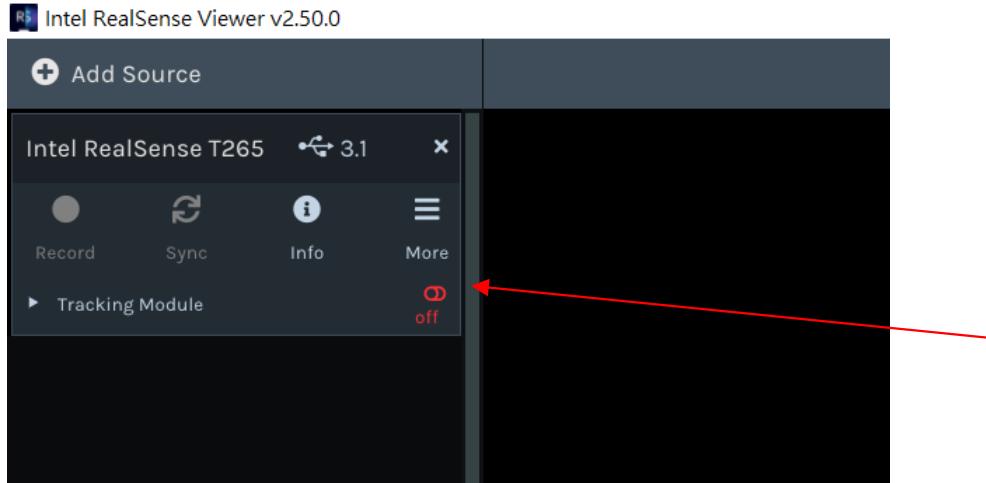
It's easy to build the future with the open source Intel® RealSense™ SDK 2.0 and other tools available provided in one package. Supporting various platforms and programming languages.

Windows Linux Android iOS

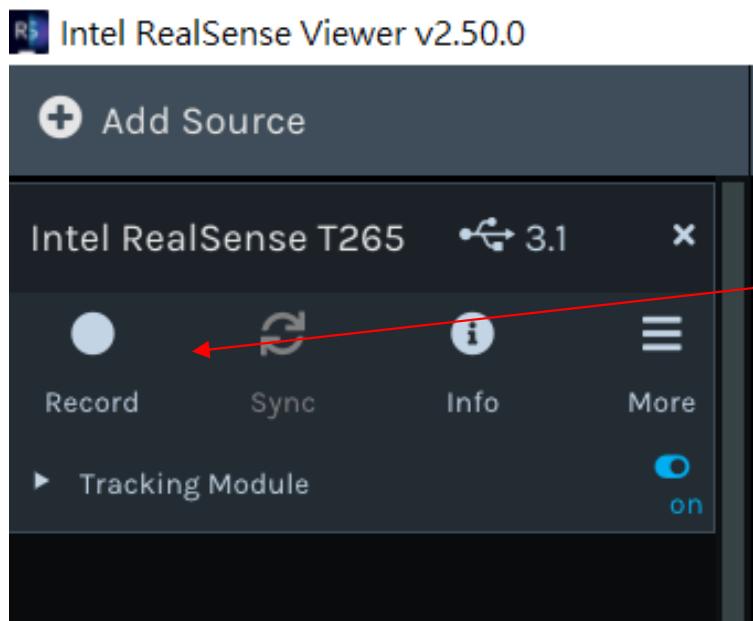
Download SDK

Source: <https://www.intelrealsense.com/sdk-2/>

Data Collection: How to save the .bag file



1. Check the setting of the camera, such as camera resolution, fps and so on.
2. Turn on the camera module



3. Click the 'Record' button

Intel RealSense SDK: Python

The screenshot shows the Intel RealSense Python documentation page. The top navigation bar includes links for PRODUCTS, DEVELOPERS, USE CASES, BLOG, SUPPORT, and STORE, along with a Log In button and a search bar. The left sidebar has a tree view of documentation categories: Documentation (Pi 3, Firefly-RK3399 Installation, Odroid installation), FIRMWARE UPDATE (Overview, Firmware releases D400, Firmware releases L500, Firmware Update Tool (rs-fw-update)), and SDK 2.0 CODE SAMPLES WRAPPERS AND LANGUAGES (Overview, C/C++, Python, ROS & ROS2, OpenCV, Node.js, UnrealEngine4, PCL (Point Cloud Library)). The Python category is currently selected. The main content area features a large title "Python" with a "Suggest Edits" button. Below it are sections for "Installation Guidelines" (link to Python Installation), "Building from Source" (link to Building from Source), and "List of Examples". The "List of Examples" section includes a note about GitHub and a table of contents. A table provides details for two examples: "Streaming Depth" and "Rendering depth".

Documentation ▾ **Python**

Pi 3
Firefly-RK3399 Installation
Odroid installation

FIRMWARE UPDATE

Overview
> Firmware releases D400
Firmware releases L500
Firmware Update Tool (rs-fw-update)

SDK 2.0 CODE SAMPLES WRAPPERS AND LANGUAGES

Overview
> C/C++
Python
Stream over Ethernet - Python Example
Box Measurement and Multi-camera Calibration
> ROS & ROS2
OpenCV
Node.js
UnrealEngine4
PCL (Point Cloud Library)

Python

Suggest Edits

Installation Guidelines

Please refer to installation guideline at [Python Installation](#)

Building from Source

Please refer to the instructions at [Building from Source](#)

List of Examples

These Examples demonstrate how to use the python wrapper of the SDK.

Sample code source code is available on [GitHub](#)
For full Python library documentation please refer to [module-pyrealsense2](#)

	Example	Description	Camera/SKU	Link to GitHub
1	Streaming Depth	This example demonstrates how to start streaming depth frames from the camera and display the image in the console as an ASCII art.	D400/L500	python-tutorial-1-depth
	Rendering depth	This example demonstrates		

<https://dev.intelrealsense.com/docs/python2>

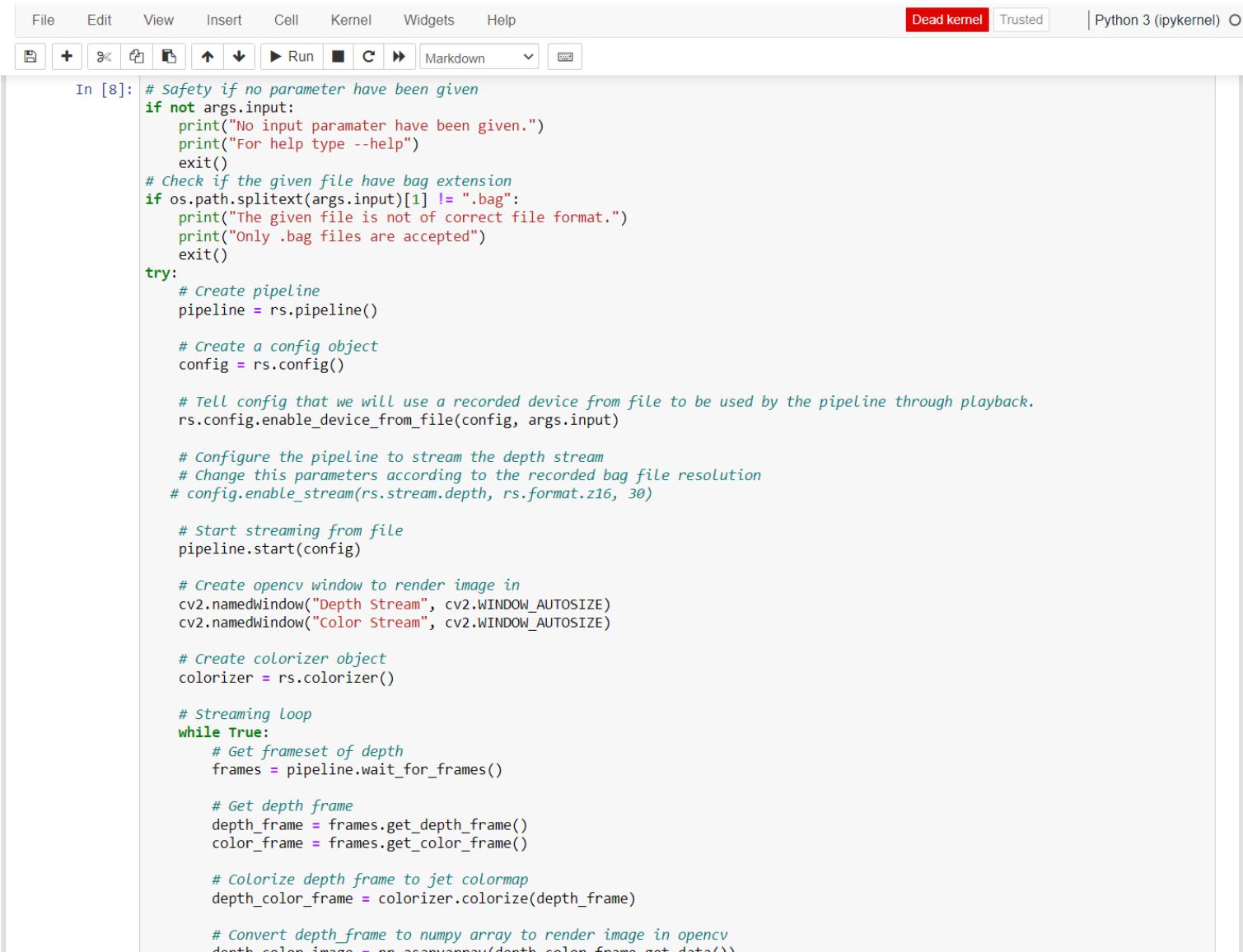
Installation on Mac

- Now the installation is as simple as that

```
pip install pyrealsense2-macosx
```

[pip install pyrealsense2 \(for windows\)](#)

SDK: Read Bag file



The screenshot shows a Jupyter Notebook interface with the following details:

- Toolbar:** File, Edit, View, Insert, Cell, Kernel, Widgets, Help.
- Status Bar:** Dead kernel, Trusted, Python 3 (ipykernel) O.
- Code Cell (In [8]):**

```
# Safety if no parameter have been given
if not args.input:
    print("No input parameter have been given.")
    print("For help type --help")
    exit()
# Check if the given file have bag extension
if os.path.splitext(args.input)[1] != ".bag":
    print("The given file is not of correct file format.")
    print("Only .bag files are accepted")
    exit()
try:
    # Create pipeline
    pipeline = rs.pipeline()

    # Create a config object
    config = rs.config()

    # Tell config that we will use a recorded device from file to be used by the pipeline through playback.
    rs.config.enable_device_from_file(config, args.input)

    # Configure the pipeline to stream the depth stream
    # Change this parameters according to the recorded bag file resolution
    # config.enable_stream(rs.stream.depth, rs.format.z16, 30)

    # Start streaming from file
    pipeline.start(config)

    # Create opencv window to render image in
    cv2.namedWindow("Depth Stream", cv2.WINDOW_AUTOSIZE)
    cv2.namedWindow("Color Stream", cv2.WINDOW_AUTOSIZE)

    # Create colorizer object
    colorizer = rs.colorizer()

    # Streaming Loop
    while True:
        # Get frameset of depth
        frames = pipeline.wait_for_frames()

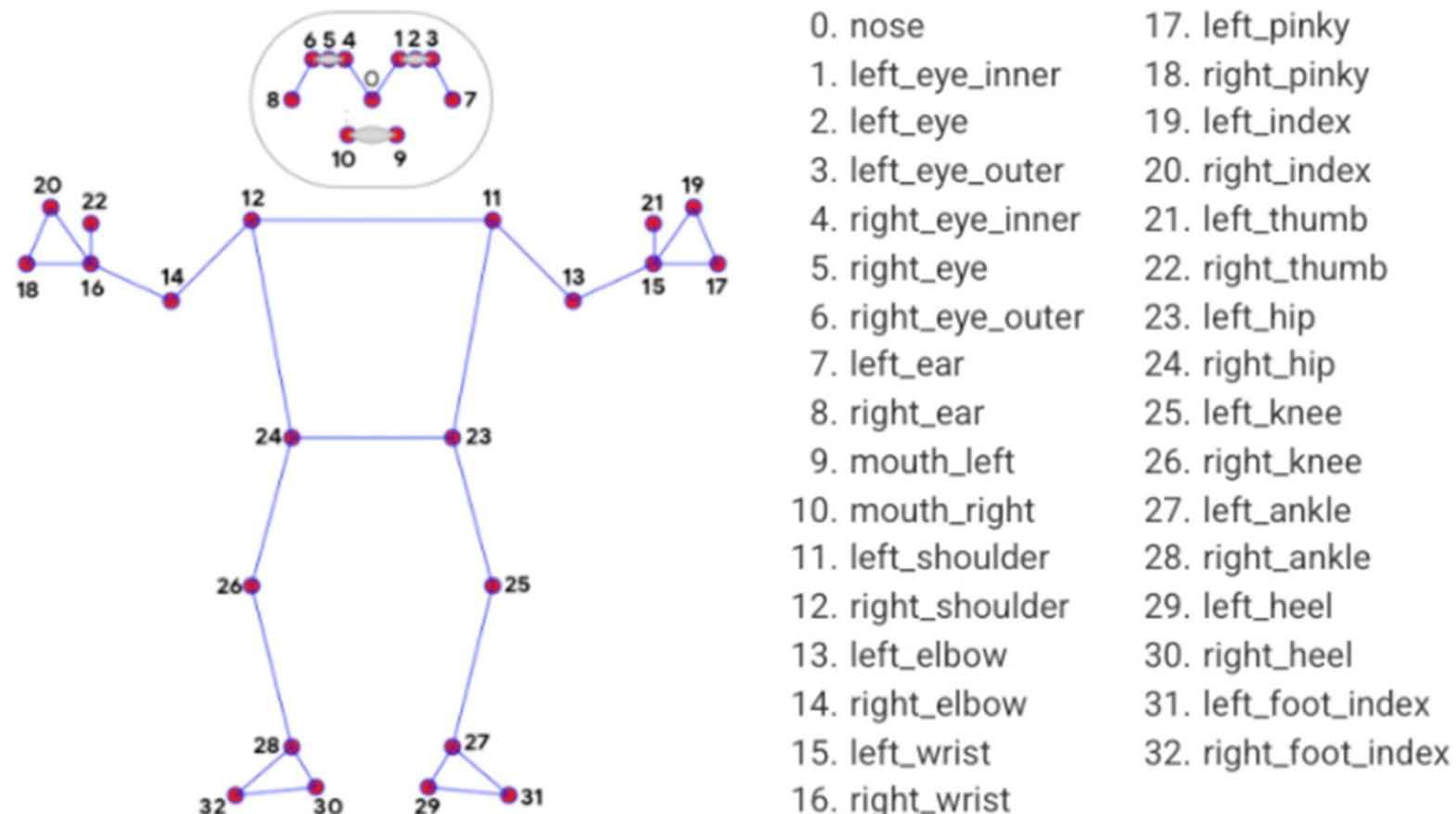
        # Get depth frame
        depth_frame = frames.get_depth_frame()
        color_frame = frames.get_color_frame()

        # Colorize depth frame to jet colormap
        depth_color_frame = colorizer.colorize(depth_frame)

        # Convert depth_frame to numpy array to render image in opencv
        depth_color_image = np.asarray(depth_color_frame.get_data())
```

Source: https://github.com/IntelRealSense/librealsense/blob/master/wrappers/python/examples/read_bag_example.py

Classwork (Teamwork): mediapipe from a bag



Source: <https://google.github.io/mediapipe/solutions/pose.html>

In [3]:

```
1 # First import library
2 import pyrealsense2 as rs
3 import mediapipe as mp
4 import numpy as np
5 import cv2
6
7 mp_drawing = mp.solutions.drawing_utils
8 mp_holistic = mp.solutions.holistic
9
10 # cap = cv2.VideoCapture(0)
11 # Create pipeline
12 pipeline = rs.pipeline()
13
14 # Create a config object
15 config = rs.config()
16
17 # Tell config that we will use a recorded device from file to be used by the pipeline through playback.
18 rs.config.enable_device_from_file(config, '001.bag')
19
20 # Configure the pipeline to stream the depth stream
21 # Change this parameters according to the recorded bag file resolution
22 #config.enable_stream(rs.stream.depth, rs.format.z16, 30)
23 #config.enable_stream(rs.stream.color, 640, 480, rs.format.bgr8, 30)
24 #config.enable_stream(rs.stream.color, 1280, 720, rs.format.bgr8, 30)
25
26
27 # Start streaming from file
28 pipeline.start(config)
29
```

1. import pyrealsense2 as rs

2. Use rs.pipeline() to replace cv2.videocature

3. Create a config object

4. Start streaming from file

RGB-D Camera

What is an RGB-D Camera?

- What is an RGB-D Camera? An RGB-D Camera is a type of depth camera that provides depth and color data as the output in real-time.
- Depth information is through a depth image which is created by a 3D depth sensor, such as a stereo camera or time of flight camera

Two types of the RGBD camera

- **Stereo Camera**

- Microsoft Kinect V1
- Intel Real Sense
- Orbbec Astra



- **Time-of-Flight(ToF) camera**

- Microsoft Kinect V2, Azure Kinect



	STEREO VISION	TIME-OF-FLIGHT
Principle	Compares disparities of stereo images from two 2D sensors	Measures the transit time of reflected light from the target object
Software Complexity	High	Low
Material Cost	Low 	Medium
Depth("z") Accuracy	cm	mm~cm 
Depth Range	Limited	Scalable
Low light	Weak	Good 
Outdoor	Good 	Fair
Response Time	Medium	Fast
Compactness	Low	Low
Power Consumption	Low 	Scalable

Tech Spec of Intel RealSense (D435i)

Depth

Depth technology:

Stereoscopic

Minimum depth distance (Min-Z) at max resolution:

~28 cm

Depth Accuracy:

<2% at 2 m¹

Depth Field of View (FOV):

87° × 58°

Depth output resolution:

Up to 1280 × 720

Depth frame rate:

Up to 90 fps

RGB

RGB frame resolution:

1920 × 1080

RGB frame rate:

30 fps

RGB sensor technology:

Rolling Shutter

RGB sensor FOV (H × V):

69° × 42°

RGB sensor resolution:

2 MP



Read Color and Depth from the Intel RealSense camera

1. Run a jupyter notebook as root. Please do the following command to start the jupyter notebook:

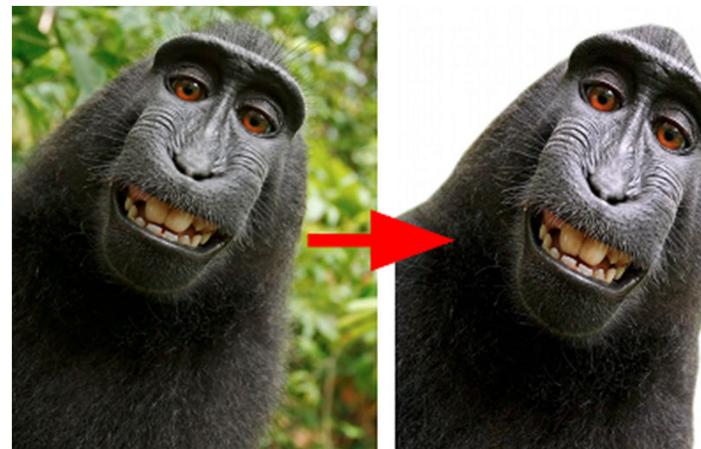
```
sudo jupyter notebook --allow-root
```

```
7  
10 # cap = cv2.VideoCapture(0)  
11 # Create pipeline  
12 pipeline = rs.pipeline()  
13  
14 # Create a config object  
15 config = rs.config()      2. comment the enable_device_from_file  
16  
17 # Tell config that we will use a recorded device from file to be used by the pipeline through playback.  
18 #rs.config.enable_device_from_file(config, '001.bag')  
19  
20 # Configure the pipeline to stream the depth stream  
21 # Change this parameters according to the recorded bag file resolution  
22 config.enable_stream(rs.stream.depth, rs.format.z16, 30)  
23 config.enable_stream(rs.stream.color, 640, 480, rs.format.bgr8, 30)  
24 #config.enable_stream(rs.stream.color, 1280, 720, rs.format.bgr8, 30)  
25 |  
26  
27 # Start streaming from file  
28 pipeline.start(config)  
29
```

3. enable_stream

Example

- Background subtraction using depth camera **(using depth value)**

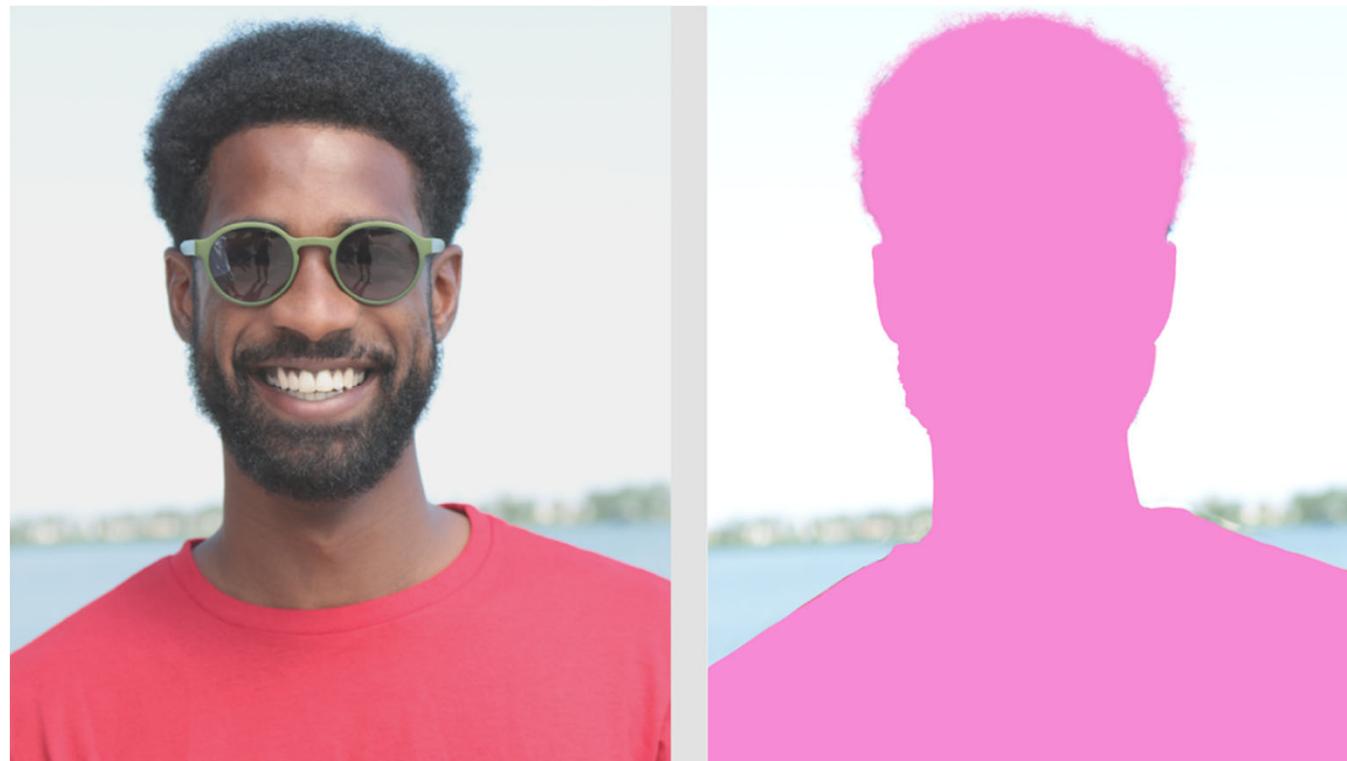


How to work

- Compute the distance from a camera to an object

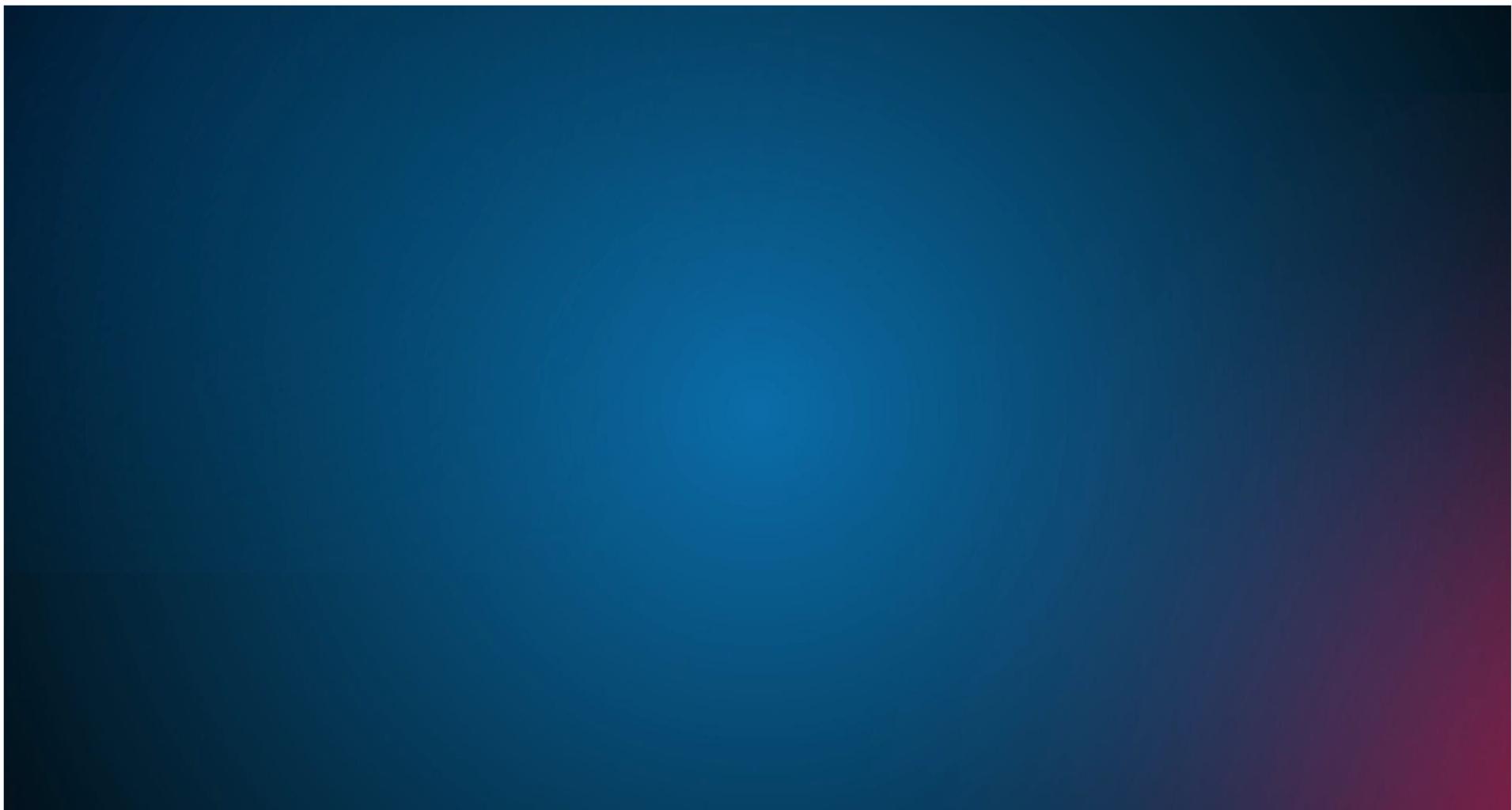
Example

- Background subtraction using color camera (using mediapipe selfie segmentation model)



Classwork (Teamwork)

Person Detection and Distance Estimation



Classwork (Teamwork)

Person Detection and Distance Estimation

Person Detection
in color camera

Distance estimation
in depth camera



1. Create a HOG detector object

```
#People detection
# initialize the HOG descriptor/person detector
hog = cv2.HOGDescriptor()
hog.setSVMClassifier(cv2.HOGDescriptor_getDefaultPeopleDetector())
```

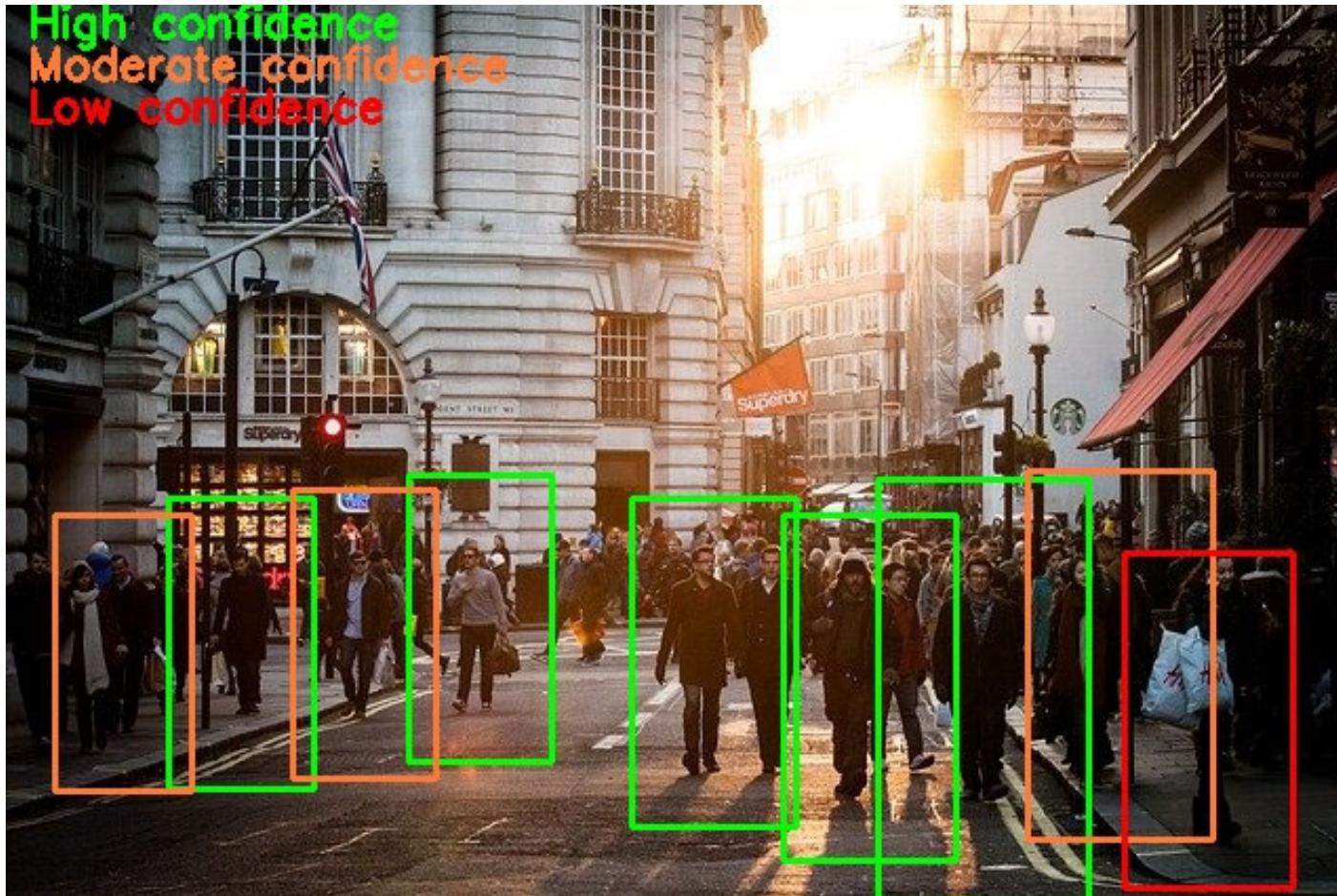
2. Detect people in the color image

```
# detect people in the image
# returns the bounding boxes for the detected objects
boxes, weights = hog.detectMultiScale(gray, hitThreshold=0.8, winStride=(8,8) )

boxes = np.array([[x, y, x + w, y + h] for (x, y, w, h) in boxes])

for (xA, yA, xB, yB) in boxes:
    # display the detected boxes in the colour picture
    cv2.rectangle(color_image, (xA, yA), (xB, yB), (255, 255, 255), 2)
```

Person Detection



Distance Estimation

F You

please write a python code to calculate the average value of an region of interest area in a depth map

O

ChatGPT

Certainly! To calculate the average value of a region of interest (ROI) in a depth map using Python, you can use the following code as a starting point. This assumes that your depth map is represented as a 2D NumPy array:

```
python Copy code
import numpy as np

def calculate_average_depth(depth_map, roi):
    """
    Calculate the average depth value in a region of interest (ROI) of

    Parameters:
    - depth_map: 2D NumPy array representing the depth map.
    - roi: Tuple (x, y, width, height) representing the coordinates and

    Returns:
    - average_depth: Average depth value in the specified ROI.
    """

    # Extract ROI coordinates and size
    x, y, width, height = roi
    ↓
    # Extract the ROI from the depth map
```

Message ChatGPT...



Calculate the average value of a region of interest (ROI) in a depth map

```
import numpy as np
```

```
for (xA, yA, xB, yB) in boxes:  
    # display the detected boxes in the colour picture  
    cv2.rectangle(color_image, (xA+offset, yA+offset), (xB-offset, yB-offset), (255, 255, 255), 2)  
    roi_depth = (xA, yA, xB, yB)  
    mean_depth = np.mean(roi_depth)  
    distance = str(mean_depth) + ' cm'  
    cv2.putText(color_image, distance, (xA+offset, yA+offset), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 255), 1, cv2.LINE_AA)
```

1. Create a ROI as `roi_depth`
2. calculate the average of a ROI
3. show the average to the screen



Write to AVI

1. Video Setups

```
#Prepare a Video Setups
#height, width, channels = color_image.shape
height = 360
width = 640
out = cv2.VideoWriter('out.avi',cv2.VideoWriter_fourcc(*'mp4v'), 30,(width,height))
```

2. Write to AVI

```
#write to AVI file
out.write(color_image)
```

Classwork (Teamwork)

Person Detection and Distance Estimation

Person Detection
in color camera
(30%)

Multi-Persons (20%)
with confidence (20%)

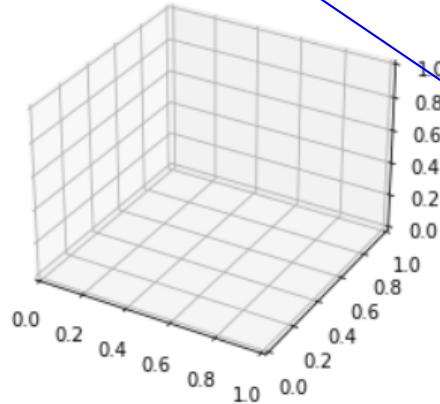
Distance estimation
in depth camera
(30%)



3D plotting in Python

- Matplotlib

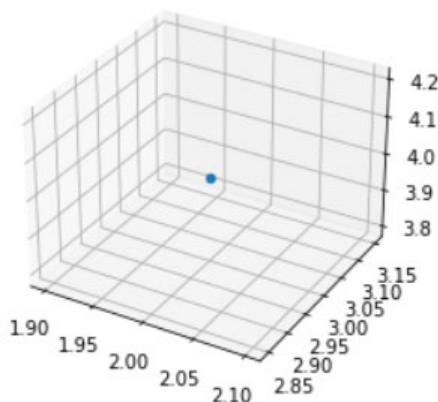
```
In [4]: import matplotlib.pyplot as plt  
  
from mpl_toolkits.mplot3d import Axes3D  
  
fig = plt.figure(figsize=(4,4))  
  
ax = fig.add_subplot(111, projection='3d')
```



Rotate View
%matplotlib notebook

Plot the point

```
import matplotlib.pyplot as plt  
  
from mpl_toolkits.mplot3d import Axes3D  
  
fig = plt.figure(figsize=(4,4))  
  
ax = fig.add_subplot(111, projection='3d')  
  
ax.scatter(2,3,4) # plot the point (2,3,4) on the figure  
  
plt.show()
```

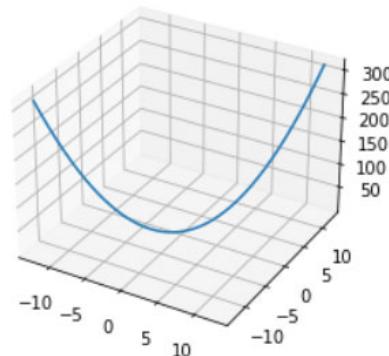


Plotting a 3D continuous line

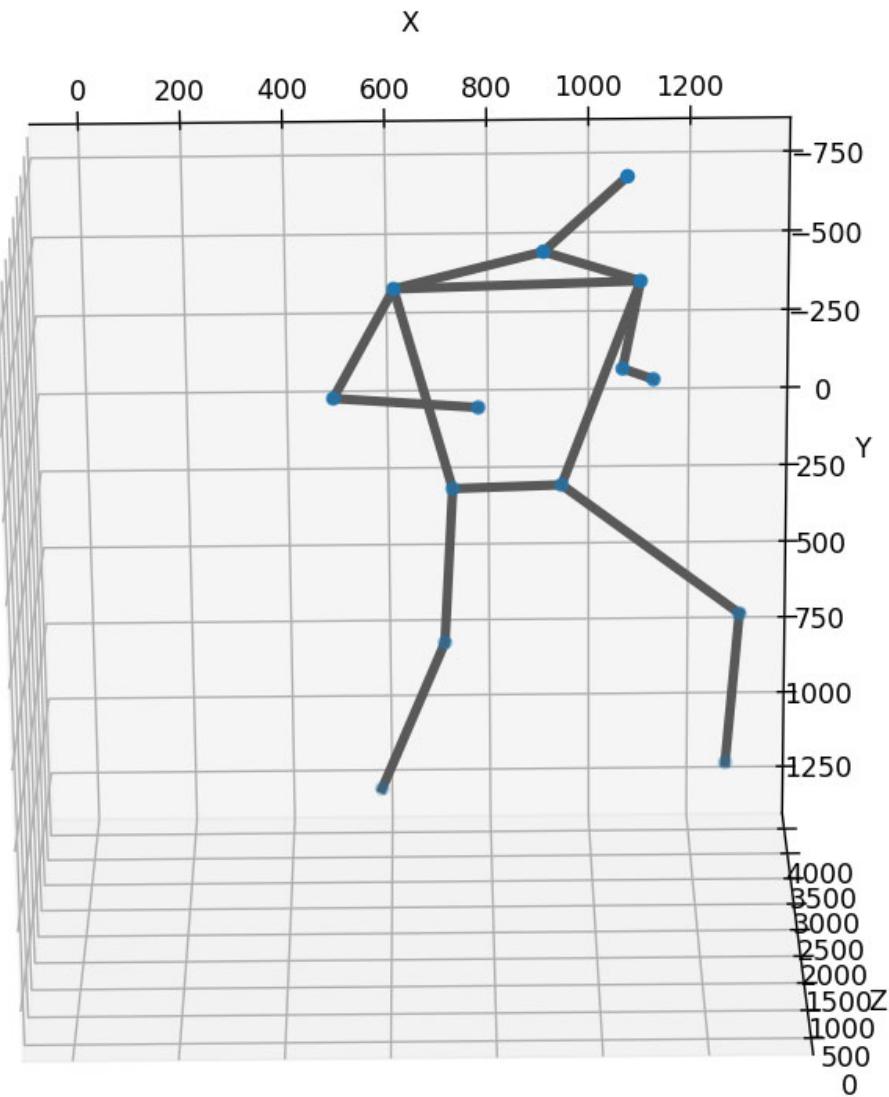
```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np

x = np.linspace(-4*np.pi,4*np.pi,50)
y = np.linspace(-4*np.pi,4*np.pi,50)
z = x**2 + y**2

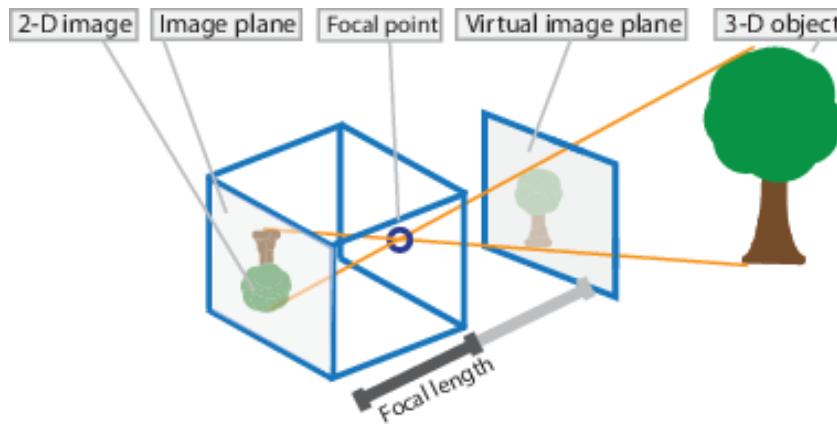
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot(x,y,z)
plt.show()
```



Example: Plotting a 3D human skeleton



How to project the pixel of 2D to 3D?



Pin Hole Camera Model
intrinsic and extrinsic parameters

Intrinsic: provides the geometry and optical parameters, such as focal length, image center and lens distortion (3x3 matrix)

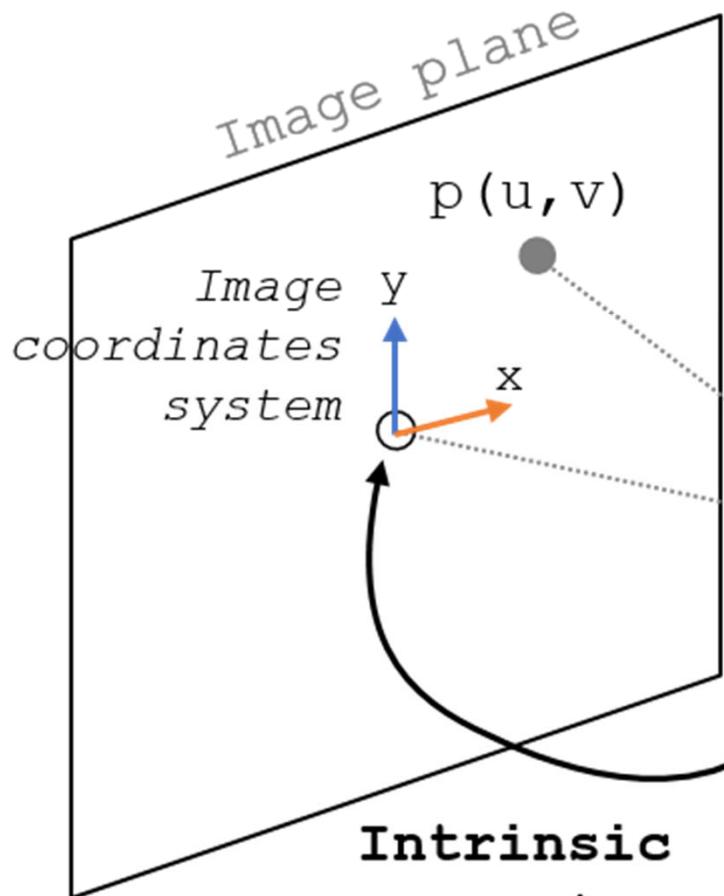
Extrinsic: provides the **RT**. (Rotation and Translation, 4x3 matrix)

Image size (pixels)	[480 640]									
Focal length (pixels)	[839.3458 ± 3.6694, 839.5573 ± 3.7166]									
Principal point (pixels)	[332.3661 ± 1.4489, 259.5099 ± 1.5829]									
Radial distortion	[0.0101 ± 0.0167 -0.1883 ± 0.1895]									
Mean projection error	0.363945706962709									
Intrinsic parameters	<table border="1"><tr><td>839.345758</td><td>0.000000</td><td>0.000000</td></tr><tr><td>0.000000</td><td>839.557331</td><td>0.000000</td></tr><tr><td>332.366095</td><td>259.509924</td><td>1.000000</td></tr></table>	839.345758	0.000000	0.000000	0.000000	839.557331	0.000000	332.366095	259.509924	1.000000
839.345758	0.000000	0.000000								
0.000000	839.557331	0.000000								
332.366095	259.509924	1.000000								

The intrinsic matrix is a 3x3 matrix

Intrinsic

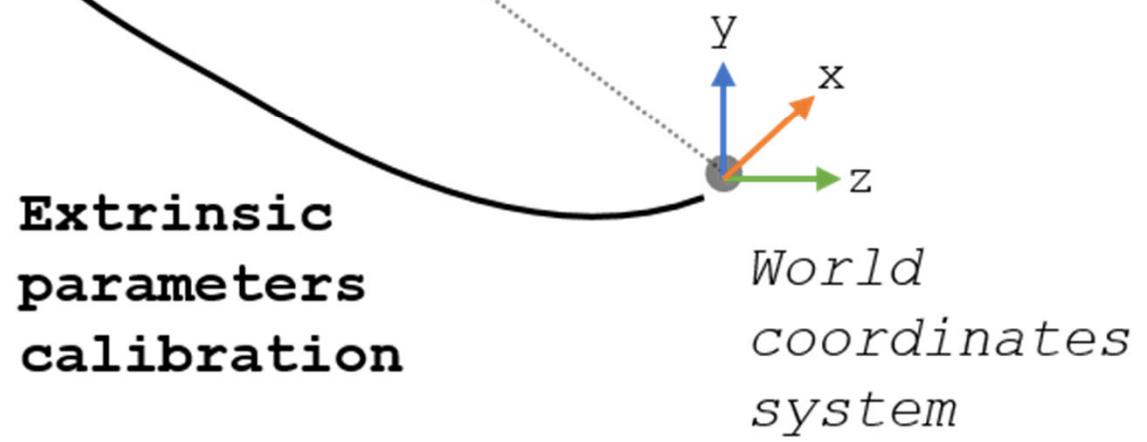
<https://www.youtube.com/watch?v=ND2fa08vxkY>



$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

2D Image Coordinates Intrinsic properties (Optical Centre, scaling)
Extrinsic properties (Camera Rotation and translation) 3D World Coordinates

The extrinsic matrix is a 4×3 matrix



**Intrinsic
parameters
calibration**

**Extrinsic
parameters
calibration**

Projection in Intel RealSense SDK 2.0

intel
REALSENSE

PRODUCTS

DEVELOPERS

USE CASES

BLOG

SUPPORT

STORE

Log In

Documentation

Projection in Intel RealSense SDK 2.0

Search

CTRL-K

Overview

SDK KNOWLEDGE BASE

Overview

SDK Supported Features Matrix

Frame metadata

Frame management

Post-processing filters

Troubleshooting Tips and Tricks

Software support model

Intel RealSense D400 Advanced Mode

Projection in Intel RealSense SDK 2.0

Build Configuration

Intel RealSense SDK 1.0

Skeleton Tracking SDK Installation guide

CALIBRATION

Overview

User guide for Intel RealSense D400 Series calibration tools

Programmer's guide for Intel RealSense D400 Series calibration tools and API

IMU Calibration Tool for Intel® RealSense™ Depth Camera

Intel RealSense D400 Series Custom

Projection in Intel RealSense SDK 2.0

 Suggest Edits

This document describes the projection mathematics relating the images provided by the Intel RealSense depth devices to their associated 3D coordinate systems, as well as the relationships between those coordinate systems. These facilities are mathematically equivalent to those provided by previous APIs and SDKs, but may use slightly different phrasing of coefficients and formulas.

Pixel coordinates

Each stream of images provided by this SDK is associated with a separate 2D coordinate space, specified in pixels, with the coordinate $[0,0]$ referring to the center of the top left pixel in the image, and $[w-1,h-1]$ referring to the center of the bottom right pixel in an image containing exactly w columns and h rows. That is, from the perspective of the camera, the x-axis points to the right and the y-axis points down. Coordinates within this space are referred to as "pixel coordinates", and are used to index into images to find the content of particular pixels.

Point coordinates

Each stream of images provided by this SDK is also associated with a separate 3D coordinate space, specified in meters, with the coordinate $[0,0,0]$ referring to the center of the physical imager. Within this space, the positive x-axis points to the right, the positive y-axis points down, and the positive z-axis points forward. Coordinates within this space are referred to as "points", and are used to describe locations within 3D space that might be visible within a particular image.

Intrinsic camera parameters

The relationship between a stream's 2D and 3D coordinate systems is described by its intrinsic camera parameters, contained in the `rs2_intrinsics` struct. Each model of RealSense device is somewhat different, and the `rs2_intrinsics` struct must be capable of describing the images produced by all of them. The basic set of assumptions is described below:

TABLE OF CONTENTS

[Pixel coordinates](#)

[Point coordinates](#)

[Intrinsic camera parameters](#)

[Distortion Models](#)

[Extrinsic camera parameters](#)

[Depth Image Formats](#)

[Processing blocks helpers](#)

[Point Cloud](#)

[Frame Alignment](#)

[Appendix: Model Specific Details](#)

SOURCE: <https://dev.intelrealsense.com/docs/projection-in-intel-realsense-sdk-20>

Converting 2D pixel coordinate to 3D point coordinate using Intel Realsense SDK

```
In [ ]: import pyrealsense2 as rs
import numpy as np
import cv2

# Create a pipeline
pipeline = rs.pipeline()

# Create a config object
config = rs.config()

# Tell config that we will use a recorded device from file to be used by the pipeline through playback.
rs.config.enable_device_from_file(config, '001.bag')

# Configure the pipeline to stream the depth stream
# Change this parameters according to the recorded bag file resolution
# config.enable_stream(rs.stream.depth, rs.format.z16, 30)

# Start streaming
profile = pipeline.start(config)

align_to = rs.stream.color
align = rs.align(align_to)

frames = pipeline.wait_for_frames()

# align the depth and color frame
aligned_frames = align.process(frames)
aligned_frames = frames

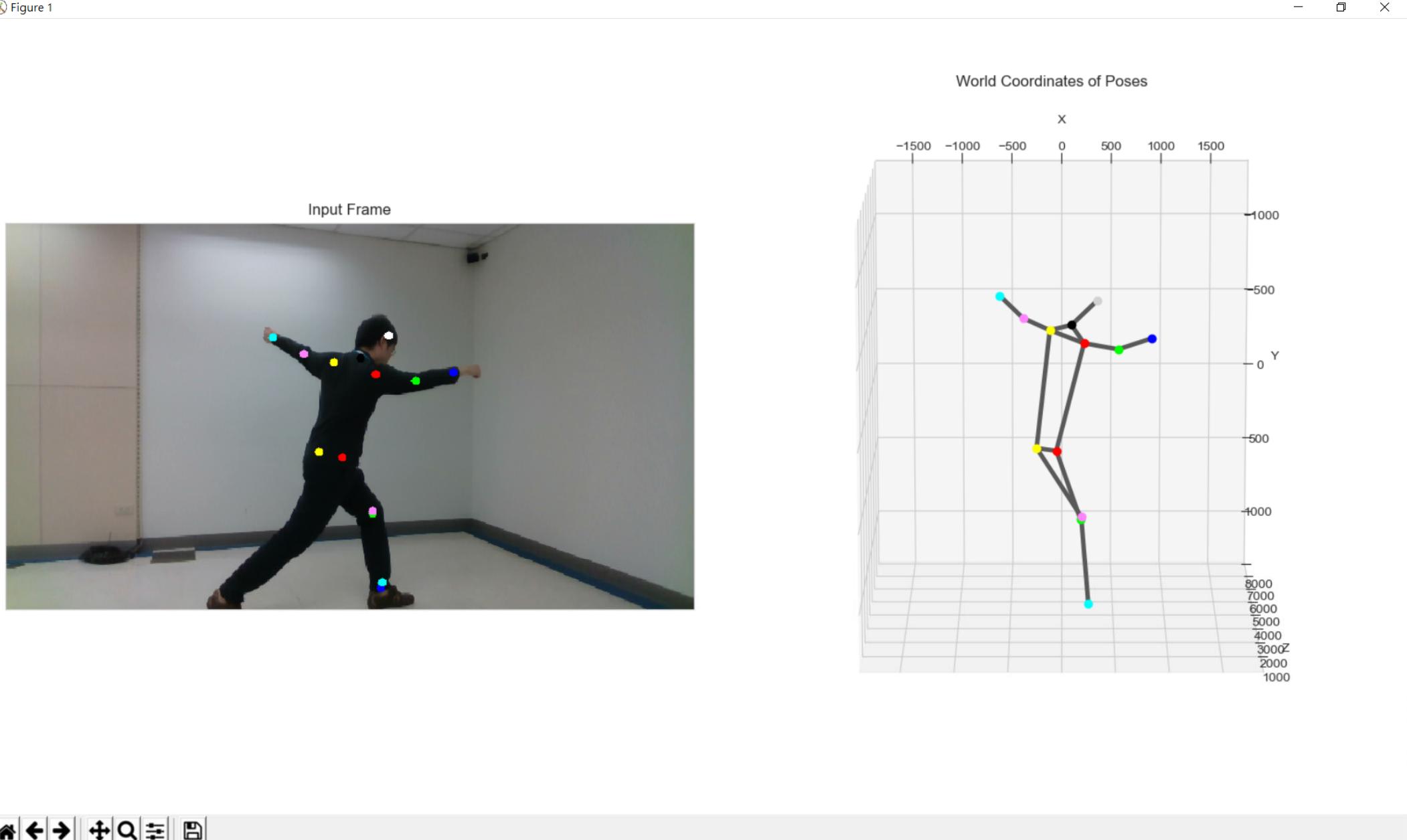
# Get depth and color frame
depth_frame = frames.get_depth_frame()
color_frame = frames.get_color_frame()

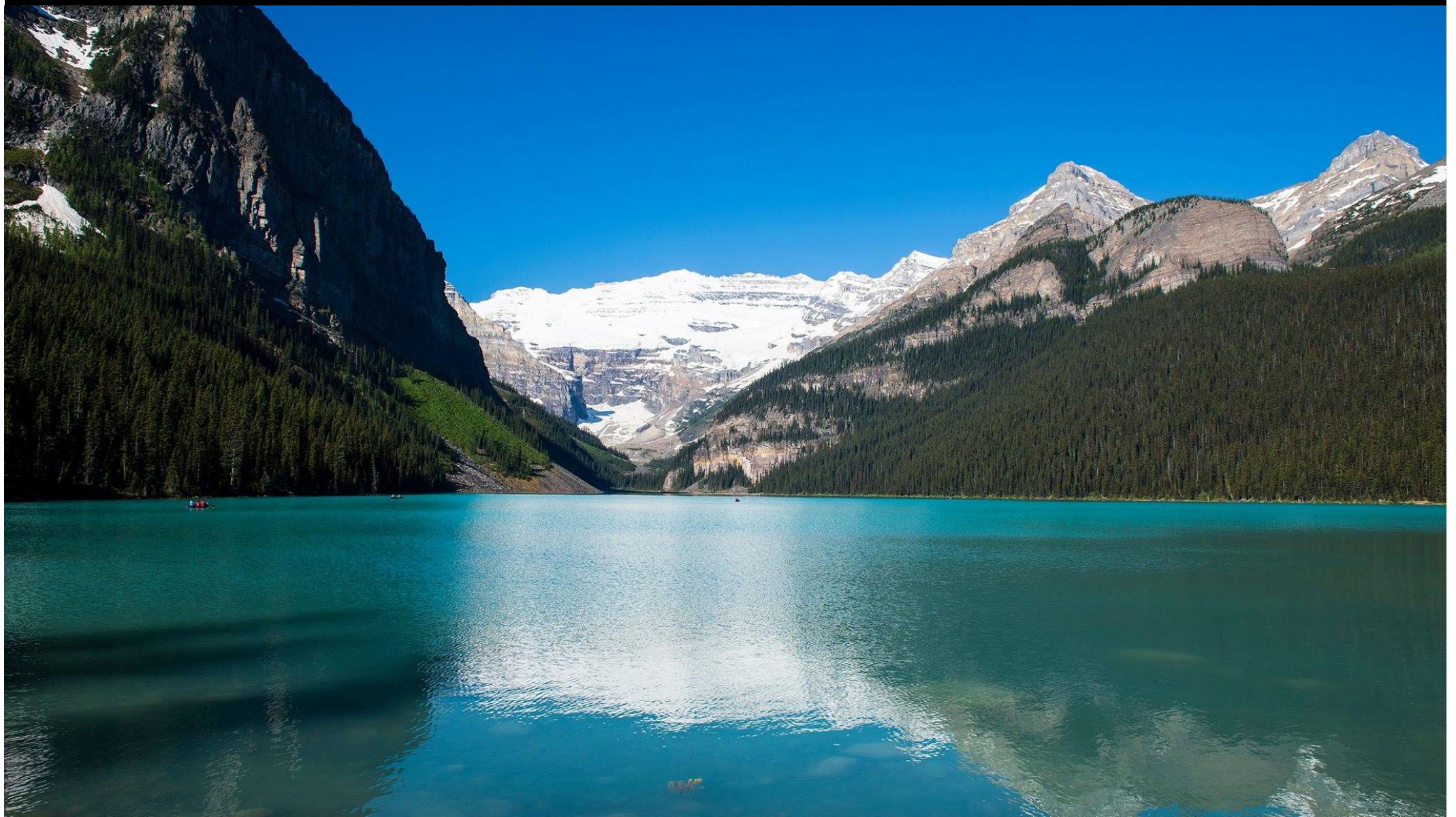
#Get camera intrin
```

Classwork(Teamwork):

Plot the tracking results of mediapipe in the 3D space

Figure 1





Q&A