3D Coordinates from RealSense Camera using Mouse Click

```
import pyrealsense2 as rs
import numpy as np
import cv2

pipeline = rs.pipeline()
config = rs.config()
config.enable_stream(rs.stream.depth, 640, 480, rs.format.z16, 30)
config.enable_stream(rs.stream.color, 640, 480, rs.format.bgr8, 30)
pipeline.start(config)

for _ in range(30):
    pipeline.wait_for_frames()

frames = pipeline.wait_for_frames()
depth_frame = frames.get_depth_frame()
color_frame = frames.get_color_frame()
if not depth_frame or not color_frame:
    raise RuntimeError("Could not get frames.")

color_image = np.asanyarray(color_frame.get_data())

depth_intrinsics = depth_frame.profile.as_video_stream_profile().intrinsics

def mouse_callback(event, x, y, flags, param):
    if event == cv2.EVENT_LBUTTONDOWN:
        depth = depth_frame.get_distance(x, y)

        if depth == 0 or depth > 3:
            print("Invalid or out-of-range depth")
        else:
            camera_coordinates = rs.rs2_deproject_pixel_to_point(depth_intrinsics, [x,
y], depth)
            print(f"Clicked at pixel: ({x}, {y}), Depth: {depth:.4f}m")
            print(f"3D Camera Coordinates (X, Y, Z): {camera_coordinates}")

cv2.namedWindow("Color Frame")
cv2.setMouseCallback("Color Frame", mouse_callback)

while True:
    frames = pipeline.wait_for_frames()
    depth_frame = frames.get_depth_frame()
    color_frame = frames.get_color_frame()
    color_image = np.asanyarray(color_frame.get_data())

    cv2.imshow("Color Frame", color_image)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

pipeline.stop()
cv2.destroyAllWindows()
```

1. Import Required Libraries

```
import pyrealsense2 as rs
import numpy as np
import cv2
```

`pyrealsense2`: to access the RealSense camera.
`numpy`: For array operations (used to handle image data).
`cv2`: OpenCV, used to display the video feed and handle mouse interactions.

2. Set Up RealSense Camera Pipeline

```
pipeline = rs.pipeline()
config = rs.config()
config.enable_stream(rs.stream.depth, 640, 480, rs.format.z16, 30)
config.enable_stream(rs.stream.color, 640, 480, rs.format.bgr8, 30)
pipeline.start(config)
```

`pipeline`: Represents the flow of camera data (like video/audio stream).
`config`: Configuration object to specify what data to stream.
`enable_stream`: Enable both **depth** and **color** streams at resolution 640x480 and 30 FPS.
`pipeline.start`: Starts the RealSense pipeline with the config.

3. Warm Up the Camera

```
for _ in range(30):
    pipeline.wait_for_frames()
```

Captures 30 frames to let the camera adjust exposure and auto settings. This improves initial frame quality.

4. Get Initial Frames

```
frames = pipeline.wait_for_frames()
depth_frame = frames.get_depth_frame()
color_frame = frames.get_color_frame()
if not depth_frame or not color_frame:
    raise RuntimeError("Could not get frames.")
```

Retrieves the most recent set of synchronized frames.
Separates out the depth and color frames.
If either frame is missing, it throws an error.

5. Convert Color Frame to Numpy Array and Get Camera Intrinsics

```
color_image = np.asanyarray(color_frame.get_data())
depth_intrinsics =
depth_frame.profile.as_video_stream_profile().intrinsics
```

Converts the color frame into a NumPy array so it can be displayed using OpenCV.
Gets the intrinsic parameters (focal length, principal point, etc.) of the depth camera.
Required to convert 2D image coordinates and depth values into 3D world coordinates.

7. Define Mouse Click Event Handler

```
def mouse_callback(event, x, y, flags, param):
    if event == cv2.EVENT_LBUTTONDOWN:
        depth = depth_frame.get_distance(x, y)

        if depth == 0 or depth > 3:
            print("Invalid or out-of-range depth")
        else:
            camera_coordinates = 
rs.rs2_deproject_pixel_to_point(depth_intrinsics, [x, y], depth)
            print(f"Clicked at pixel:({x},{y}), Depth:{depth:.4f}m")
            print(f"3D Camera Coordinates (X,Y,Z): 
{camera_coordinates}")
```

This function is called whenever a mouse event occurs on the OpenCV window.

On **left click**:
-Gets the depth at the clicked pixel.
-If depth is valid and within 3 meters, it uses the deprojection function to convert the 2D pixel & depth into 3D camera-space coordinates.
-Prints the clicked position and corresponding 3D coordinates.

7.Set Up OpenCV Window and Mouse Callback

```
cv2.namedWindow("Color Frame")
cv2.setMouseCallback("Color Frame", mouse_callback)
```

Creates a window titled "Color Frame".

Binds the mouse click callback to this window.

8. Main Loop to Display Real-Time Video

```
while True:
    frames = pipeline.wait_for_frames()
    depth_frame = frames.get_depth_frame()
    color_frame = frames.get_color_frame()

    color_image = np.asanyarray(color_frame.get_data())

    cv2.imshow("Color Frame", color_image)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

Continuously grabs new frames from the camera.
Converts the color frame to a NumPy image.
Displays the live color feed using OpenCV.
If the q key is pressed, exits the loop.

Output :

```
Clicked at pixel:(320,240), Depth:0.8463m
3D Camera Coordinates (X,Y,Z): [0.015793442726135254,
-0.012248567305505276, 0.8462697267532349]
```