Robotics Lab 08

Syeda Manahil Wasti, Syed Mujtaba Hassan, Aatiqa Khalid

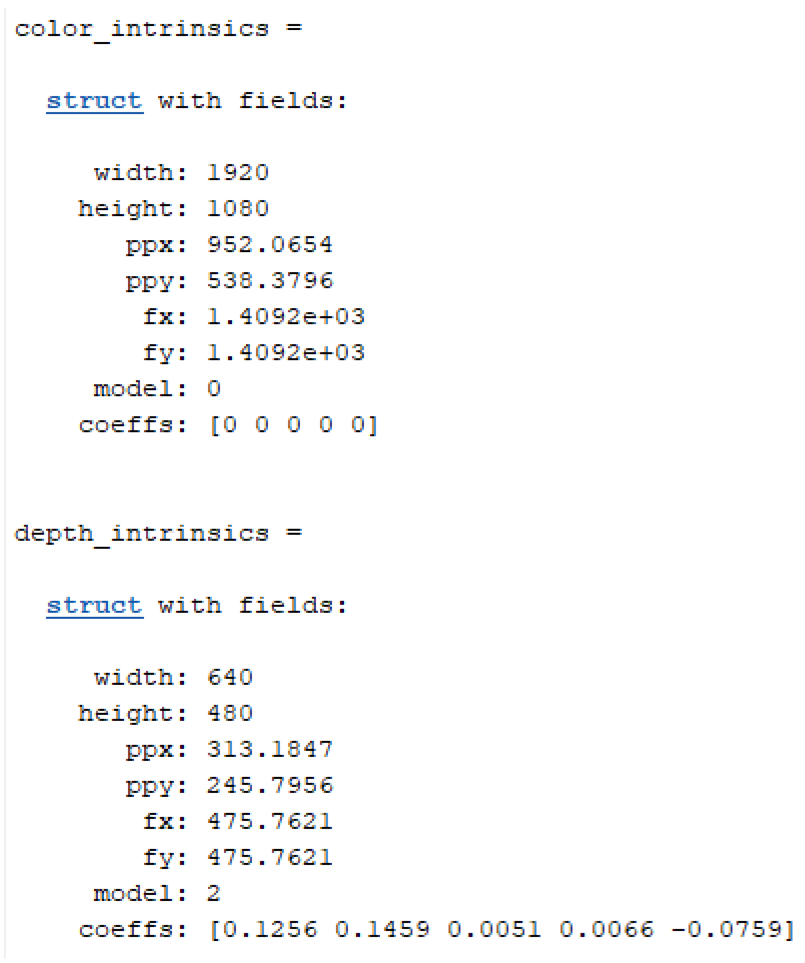
May 14, 2023

1. The MATLAB function determineIntrinsics(), provided on LMS, uses Intel's SDK to display the intrinsic parameters of the color camera. Modify the code to also display the intrinsic parameters of the IR camera. Elaborate each entry in light of [2].

Solution:

```
function determineIntrinsics()
    % Make Pipeline object to manage streaming
    pipe = realsense.pipeline();

    % Start streaming on an arbitrary camera with default settings
    profile = pipe.start();

    % Extract the color stream
    color_stream = profile.get_stream(realsense.stream.color).as('
       video_stream_profile');
    % Extract the depth stream
    depth_stream = profile.get_stream(realsense.stream.depth).as('
       video_stream_profile');

    % Get and display the intrinsics
    color_intrinsics = color_stream.get_intrinsics() % color camera
    depth_intrinsics = depth_stream.get_intrinsics() % IR camera

    % intrinsic matrix of color camera
    I = [color_intrinsics.fx, 0, color_intrinsics.ppx;
        0, color_intrinsics.fy, color_intrinsics.ppy;
        0, 0, 1]
end
```

We get the following intrinsics for the color and IR cameras:

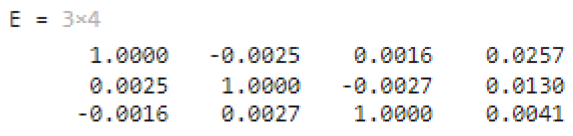


```
color_intrinsics =

  struct with fields:

     width: 1920
    height: 1080
       ppx: 952.0654
       ppy: 538.3796
        fx: 1.4092e+03
        fy: 1.4092e+03
     model: 0
    coeffs: [0 0 0 0 0]


depth_intrinsics =

  struct with fields:

     width: 640
    height: 480
       ppx: 313.1847
       ppy: 245.7956
        fx: 475.7621
        fy: 475.7621
     model: 2
    coeffs: [0.1256 0.1459 0.0051 0.0066 -0.0759]
```

1. **width** and **height** describe the number of rows and columns in the image respectively.
2. **ppx** and **ppy** describe the pixel coordinates of the principal point (center of projection).
3. **fx** and **fy** describe the focal length of the image, as a multiple of pixel width and height.
4. **model** describes the distortion model used to calibrate the image.
5. **coeffs** provides an array of up to five coefficients describing the distortion model.

We can see that the color camera has **model: 0** which means that, "an image has no distortion, as though produced by an idealized pinhole camera. This is typically the result of some hardware or software algorithm undistorting an image produced by a physical imager, but may simply indicate that the image was derived from some other image or images which were already undistorted."[2]
On the other hand, IR camera has **model: 2** which means that, "an image is distorted, and has been calibrated according to the inverse of the Brown-Conrady Distortion model. This model provides a closed-form formula to map from distorted points to undistored points, while mapping in the other direction requires iteration or lookup tables."[2]

2. The MATLAB function determineExtrinsics(), provided on LMS, will display the homogeneous transformation from the depth camera to color camera. Write it in the standard format of a homogeneous transformation.

> **Solution:**
>
> ```
> function determineExtrinsics()
>     % Make Pipeline object to manage streaming
>     pipe = realsense.pipeline();
>
>     % Start streaming on an arbitrary camera with default settings
>     profile = pipe.start();
>
>     % Extract the color and depth streams
>     color_stream = profile.get_stream(realsense.stream.color).as('
>        video_stream_profile');
>     depth_stream = profile.get_stream(realsense.stream.depth).as('
>        video_stream_profile');
>
>     % Get and display the extrinsics
>     Tdc = depth_stream.get_extrinsics_to(color_stream);
>
>     % extrinsic matrix of color camera
>     E = [Tdc.rotation(1,1:3), 1000*Tdc.translation(1);
>          Tdc.rotation(1,4:6), 1000*Tdc.translation(2);
>          Tdc.rotation(1,7:9), 1000*Tdc.translation(3);
>          0, 0, 0, 1]
> end
> ```
>
> We get the following homogeneous transformation from the depth camera to color camera in standard format:
>
> 
>
> ```
> E = 3×4
>
>      1.0000   -0.0025    0.0016    0.0257
>      0.0025    1.0000   -0.0027    0.0130
>     -0.0016    0.0027    1.0000    0.0041
> ```

3. From the provided datasheet [1], determine the values of the following and provide an explanation for each quantity:

   - Resolution of color camera and IR camera;

     > **Solution:**
     > **color:** 1920 x 1080
     > **IR:** 680 x 480

   - Frame rates of both cameras;

     > **Solution:** **color:** 30, 60, 120, 200
     > **IR:** 10, 30

   - Depth field of view;

     > **Solution:**
     > We have the following FOV angles given to us in the datasheet:

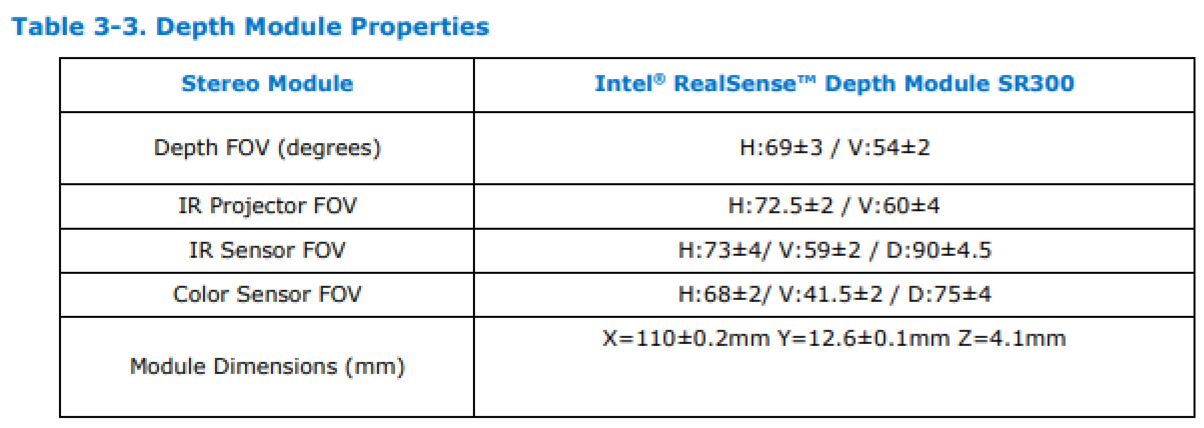


Table 3-3. Depth Module Properties

Stereo Module	Intel® RealSense™ Depth Module SR300
Depth FOV (degrees)	H:69±3 / V:54±2
IR Projector FOV	H:72.5±2 / V:60±4
IR Sensor FOV	H:73±4/ V:59±2 / D:90±4.5
Color Sensor FOV	H:68±2/ V:41.5±2 / D:75±4
Module Dimensions (mm)	X=110±0.2mm Y=12.6±0.1mm Z=4.1mm

- Depth start point.

Solution:
For SR300 we are given the following depth start point:

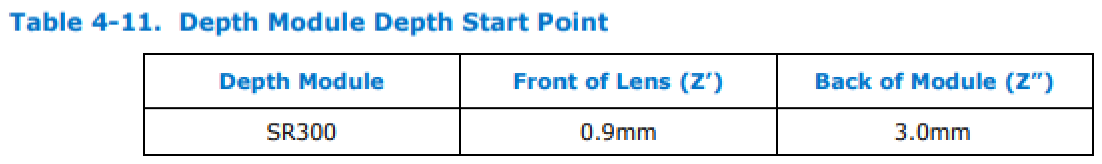
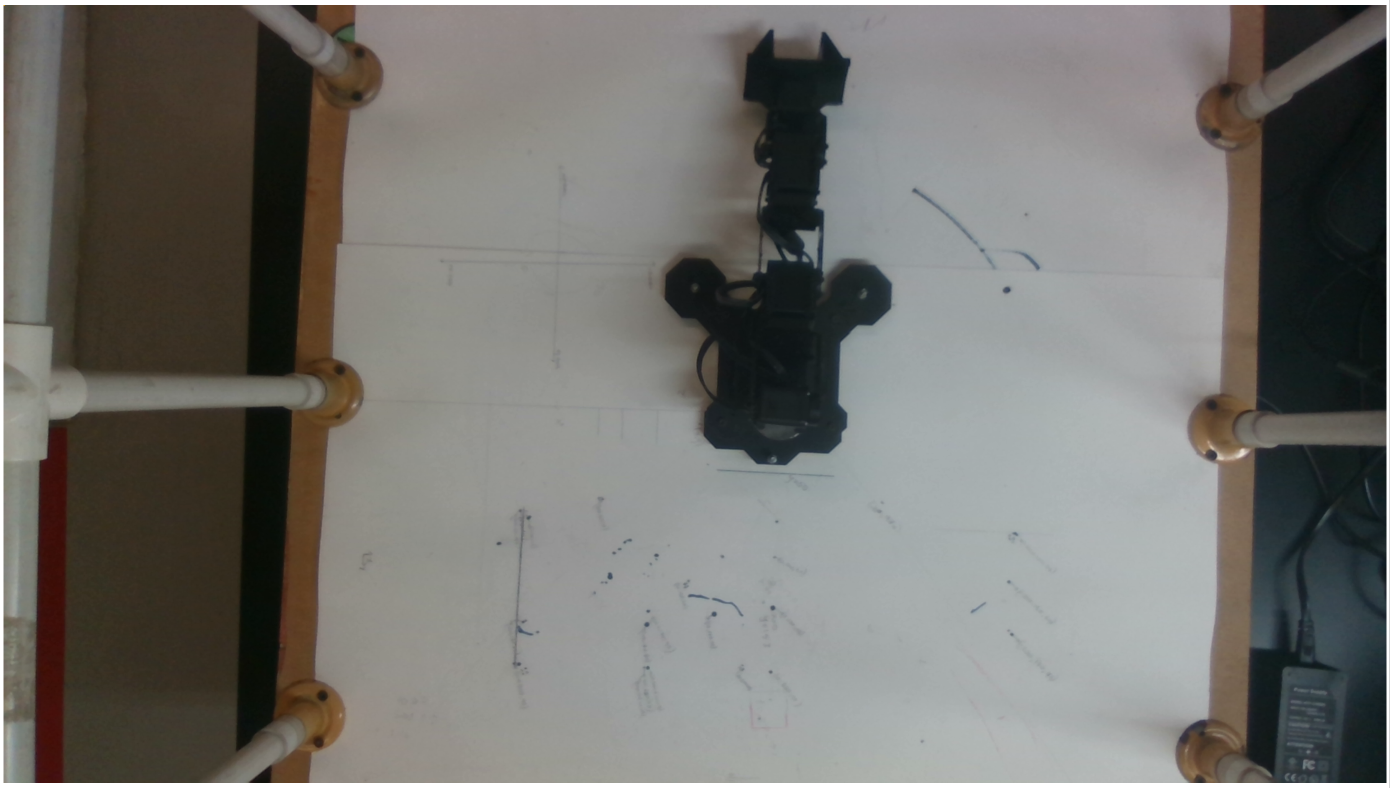


Table 4-11. Depth Module Depth Start Point

Depth Module	Front of Lens (Z′)	Back of Module (Z″)
SR300	0.9mm	3.0mm

4. Using the value for the depth field of view, compute an appropriate height (working distance) for the placement of camera so that the robot's workspace, determined earlier, is entirely in the depth FOV. Verify your computation by observing the complete workspace in the viewer. Use some way to mark the placement of your camera as we'll require that the camera be fixed during the operation of our robotic system.

Solution:
From table 3.3 of datasheet provided above, we choose the minimum angle to get the optimum height, i.e. the vertical field of view of color camera:
$\alpha = 41.5°$
$\frac{\alpha}{2} = \tan^{-1}(\frac{d}{2f})$
We measure d in real life to be 610 mm.
$f = \frac{610}{2\tan(20.75°)}$
f $\approx$ 805 mm
After setting the camera at that height, we get the whole workspace view in our image as follows:



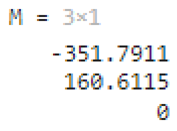
5. Determine the homogeneous transformation that transforms coordinates from the camera frame to robot base frame. Physically verify your computation and include your test results.
   You can do this by finding the coordinates of an identifiable point physically and comparing them to the coordinates determined from the camera transformed to real-world coordinates. Note that you can directly determine the depth of pixel coordinates (u, v) by using the function depth.get_distance(u,v), if depth is the extracted depth frame.

Solution:
We get the homogeneous transformation from camera to robot base frame by rotating the camera frame 180 degrees about x-axis and translating it by the vector [35; 0; 805].
We wrote the following code to use that transformation and get real-world coordinates from camera coordinates:

```
K = [1408.9, 0, 950.7;
      0, 1408.9, 533.1;
      0, 0, 1]; % camera calibration matrix of depth camera

% point (u, v) is centroid of a random object we got from lab 7
u = 335;
v = 252;
Z = 805;

% transformation from camera to base frame (180 rotation about x axis and then
   translation of [35; 0; 805])

% Rx = [1, 0, 0;
%       0, cosd(180), -sind(180);
%       0, sind(180), cosd(180);]

% p = [35; 0; 805]; % found by measuring physically
```

```
T = [1, 0, 0, -0;
     0, -1, 0, 0;
     0, 0, -1, 805;
     0, 0, 0, 1];

M = (K*T(1:3,1:4))\[u;v;1].*Z;
M = M(1:3,1) % real world coordinates (X, Y, Z)
```

We get the following real-world coordinates:



```
M = 3×1

  -351.7911
   160.6115
          0
```

6. Randomly place a single cube in your robot's workspace in an orientation such that it is possible for the arm to pick it up. In this task, your robotic system should correctly identify the location of the cube, pick it, and place it at a pre-specified free region in the workspace, using the motion control FSM from the previous lab. You'll have to connection your perception pipeline to motion control FSM. You're to submit a video of your successful test run and a note on the performance of your system with supporting data.

Solution:
We get the real world coordinates by using the code from previous task and input it into the FSM we designed in lab 6. We added our code from previous task to the code of lab 7 and tried executing the FSM code but it was not running properly. It took a lot of time to debug but still the pick and place could not be executed. However, the arm was getting the correct real world coordinates from image and reaching to the desired location.

Points to note are that the system could also have errors if object is not detected properly or if there was error in measuring the translation vector we used in finding the homogeneous transformation from camera to base frame.

References

[1] "Intel ® RealSense TM Depth Camera SR300 Series Product Family Datasheet Intel ® RealSenseTM Depth Camera SR305, Intel ® RealSenseTM Depth Module SR300 Revision 002," 2019.

[2] IntelRealSense, "Projection in RealSense SDK 2.0," GitHub, Aug. 02, 2018.