Introduction to Robotics

Lab Instructor: Miss Sadaf Sheikh
Group Members: Afsah Hyder, Ailiya Fatima, Maleeha Hussain
Course Instructor: Dr Basit Memon

LAB 08- Complete Vision-based Pick & Place Workflow

May 3, 2024

Contents

1 8.1: Camera Datasheet

IR resolution: 640x480
Color camera resolution: 640x480

Frame rate of IR camera: 10,30,60
Frame rate of color camera: 30,60,120,200

Table 4-8. Depth Field of View

Format	SR300/SR305
Horizontal FOV (deg)	69±3
Vertical FOV (deg)	54±2

Figure 1: Depth field of view

Table 4-11. Depth Module Depth Start Point

Depth Module	Front of Lens (Z')	Back of Module (Z")
SR300	0.9mm	3.0mm

Figure 2: Depth Start point

2 8.2: Real-world coordinates

```
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');
    depth_stream = profile.get_stream(realsense.stream.depth).as('video_stream_profile');

    % Get and display the intrinsics
    color_intrinsics = color_stream.get_intrinsics()
    depth_intrinsics = depth_stream.get_intrinsics()
end
```

Figure 3: Color intrinsic

```
color_intrinsics =

  struct with fields:

     width: 1920
    height: 1080
       ppx: 953.8414
       ppy: 527.4162
        fx: 1.4000e+03
        fy: 1.4000e+03
     model: 0
    coeffs: [0 0 0 0 0]


depth_intrinsics =

  struct with fields:

     width: 640
    height: 480
       ppx: 312.5867
       ppy: 243.7870
        fx: 481.0156
        fy: 481.0156
     model: 2
    coeffs: [0.1199 0.2006 0.0036 1.1447e-05 -0.1258]
```

Figure 4: Result

We measure the distance between the base frame/ which is also the world frame, and the camera frame. This turned out to be 69cm.

The u and in the code below, are the coordinates of the point in image frame- obtained by running code of Lab 3.

```
z=69;

u= bottom_left(1); v= bottom_left(2);      %  we only took bottom right , bcs the masks were for different lighting conditions, we weren't able to change it later on
u0=312.5867 ; v0=243.7;
fx=481 ; fy=481;

x=((u-u0)/fx)*z
y=((v-v0)/fy)*z


th=-pi
TcW=[1,0,0,0;
    0,cos(th),-sin(th),0;
    0,sin(th),cos(th),z;
```

```
    0,0,0,1;]
inv(TcW)
pc=[x;y;z-3;1]
pw=inv(TcW)*pc
```

pw = 4×1 single column vector
-32.7915, 12.2737, 3.0000, 1.0000

3 8.4: Vision-based pick and placer

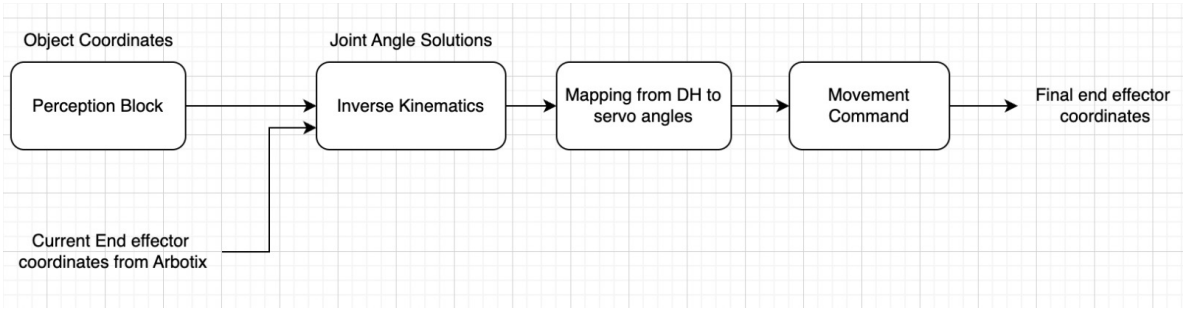

Figure 5: Outline

3.1 Working explained

The process can be broken down into several stages:

Image Acquisition: The initial step involves capturing a digital image of the workspace using a camera.

Object Segmentation: Pre-defined algorithms (developed in Lab 2) are employed to generate a mask. This mask isolates the specific object of interest (colored block) within the captured image.

Feature Extraction: Another set of algorithms (from Lab 3) is applied to the generated mask. These algorithms identify a corner point of the object within the image frame. The coordinates of this corner point are denoted as u and v.

Camera Coordinate System Localization: The pixel coordinates (u, v) obtained from the previous step are utilized to determine the object's location (x, y) within the camera's reference frame.

Transformation to World Coordinates: The object's location currently exists relative to the camera's perspective. To instruct the robot arm, we need its position in the world frame (which coincides with the object's frame). A pre-defined transformation matrix is employed to convert the camera frame coordinates (x, y) into world frame coordinates.

Motion Planning and Execution: Finally, the object's world frame coordinates are fed into the pick and place function developed in Lab 7. This function utilizes these coordinates to generate the robot arm's motion plan, enabling it to grasp and place the object at the desired location.

3.2 Code

```
z=69;

u= bottom_left(1); v= bottom_left(2);      %  we only took bottom right
    , bcs the masks were for different lighting conditions, we weren't
    able to change it later on
u0=312.5867 ; v0=243.7;
```

```
fx=481 ; fy=481;

x=((u-u0)/fx)*z
y=((v-v0)/fy)*z


th=-pi
TcW=[1,0,0,0;
    0,cos(th),-sin(th),0;
    0,sin(th),cos(th),z;
    0,0,0,1;]
inv(TcW)
pc=[x;y;z-3;1]
pw=inv(TcW)*pc

PickandPlace2(pw(1),pw(2),-3,-14,-14,-3)
```

This code is the same as task 8-2, we just called the Pick and Place function from lab 7. Note: we initially used u0,v0, fx,fy of the RGB camera but that resulted in incorrect real world coordinates. We then used the depth camera, which gave more accurate results.

We also tried using additional lighting to improve the image captured, we tried different saturation/ intensity levels for the image obtained, to get accurate corner points of the block in image frame.

P.S: the details of our grasping algorithm are already explained in lab 7.

3.3 Results

The videos of multiple successful test runs are attached:

Video 1

Video 2