Active 3D Reconstruction using Baxter

Dr. Qiang Ji

Sabbir Rashid & Andrew Cunningham

October 7, 2016

Contents

1 Abstract

Active 3D Reconstruction is performed using the Baxter, a two-armed Robot. The camera on the left end effector is used to record the scene, while a laser line projector is attached to the right end effector. Applying a thresold to reduce noise, the red channel of the image is used to find the image coordinates of the laser scan. Using full perspective projection in conjecture with the laser plane equation, the 3D coordinates of the object points are computed.

2 Problem Statement

The primary goal is to do an estimated 3D reconstruction of the environment facing a Baxter robot. Using a camera and laser line projector system, we attempted to do 3D reconstruction on the band of points that the laser illuminates. In attaching the camera and laser projector to the manipulators of a dual armed robot, we are able to dynamically position the system to obtain multiple scans.

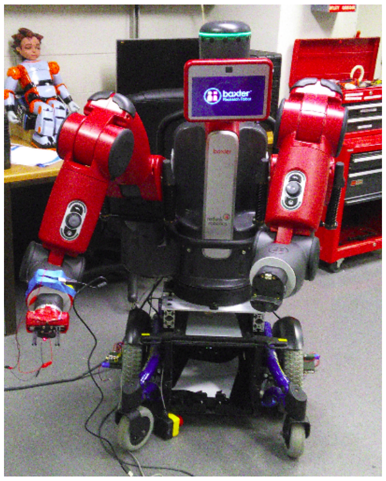


Figure 1: Experimental setup

By using ROS to perform forward kinematics to get the position and orientation for each end effector, we were able to determine both the external parameters of the camera, as well as the orientation of the laser, and thus the components of the laser plane. The intrinsic parameters were also obtained using ROS support for Baxter. Multiple laser scans and images were used to create a point cloud, which was visualized using RViz.

3 Theory

3.1 3D Reconstruction

The projection of a 3D point on a 2D image is given by

$$\lambda \begin{bmatrix} c \\ r \\ 1 \end{bmatrix} = P \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} . \tag{1}$$

If the projected 3D point lies on a plane, the equation of which is given by

$$Ax + By + Cz + D = 0 , \tag{2}$$

then a linear system of four equations can be formed to solve for the four unknown variables, λ, x, y, and z. 3D reconstruction is done by assuming the 3D points lie on the laser plane. With some rearrangement, the linear system of equations can be formed to be:

$$\begin{bmatrix} -P_{11} + \frac{A*P_{13}}{C} & -P_{12} + \frac{B*P_{13}}{C} & c \\ -P_{21} + \frac{A*P_{23}}{C} & -P_{22} + \frac{B*P_{23}}{C} & r \\ -P_{31} + \frac{A*P_{33}}{C} & -P_{32} + \frac{B*P_{33}}{C} & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ \lambda \end{bmatrix} = \begin{bmatrix} \frac{D*P_{13}}{C} + P_{14} \\ \frac{D*P_{23}}{C} + P_{34} \\ \frac{D*P_{33}}{C} + P_{24} \end{bmatrix} \tag{3}$$

Which can be solved by:

$$\begin{bmatrix} -P_{11} + \frac{A*P_{13}}{C} & -P_{12} + \frac{B*P_{13}}{C} & c \\ -P_{21} + \frac{A*P_{23}}{C} & -P_{22} + \frac{B*P_{23}}{C} & r \\ -P_{31} + \frac{A*P_{33}}{C} & -P_{32} + \frac{B*P_{33}}{C} & 1 \end{bmatrix}^{-1} \begin{bmatrix} \frac{D*P_{13}}{C} + P_{14} \\ \frac{D*P_{23}}{C} + P_{34} \\ \frac{D*P_{33}}{C} + P_{24} \end{bmatrix} = \begin{bmatrix} X \\ Y \\ \lambda \end{bmatrix} \tag{4}$$

4 Methodology

One way to perform 3D reconstruction of a scene with a laser plane and a camera is to sweep the laser plane through the scene. We opted to use the Baxter robot as our platform to do this because it has cameras embedded at the ends of each of its arms. Further, the use of forward kinematics provides the extrinsic parameters of the cameras by solving for the position and orientation of the end effectors with respect to the base. A laser line projector was attached to the right hand such that its orientation aligned with the orientations of the z-y axis of the end effector. To perform 3D reconstruction with a set configuration of the arms, it is necessary to calculate the plane parameters, detect the laser line in the image and then reconstruct the points in 3D space.

4.1 Laser Line Detection

By turning off any other lights when taking the laser scan, the image appears dark everywhere other than where the laser is hitting, marked as a red curve. Therefore, in order to do line detection, only the red channel of the image is used. As the laser disperses off the object, or hits an area further from the camera, the intensity of the light decreases. In order to reduce noise, but still detect less intense light, a threshold parameter is adjusted. Therefore, any pixel intensity in the red channel greater than a threshold value is considered a points on the object. For all such points, the column and row values are stored.

4.2 Obtaining Plane Parameters

The equation of a plane is:

$$Ax + By + Cz + D = 0\,, \qquad (5)$$

In general, a plane can be defined by a point on the plane and a vector that is normal to the plane. Equation 5 can be re-arranged to match this form:

$$N^t \begin{bmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{bmatrix} = 0 \qquad (6)$$

Where $A = N_1$, $B = N_2$, $C = N_3$ and $D = N_1x_0 + N_2y_0 + N_3z_0$. Using forward kinematics, it is possible to obtain a point on the plane (the laser pointer is attached to the end effector) and a normal vector. The normal vector is formed by representing the x-axis of the end effector in the base frame. This is accomplished by a single rotation matrix. The implementation of this follows:

```
# Get plane parameters for a given
# orientation of end effector and
#translation to end effector
def getPlaneParams(R,T):
    # we know that the x direction
    # in the manipulator frame is the
    # direction perpendicular to the
    # laser plane through inspection
    # of the baxter model
    # and the transform tree
    ex = np.array ([[1],[0],[0]])
    normal = np.dot(R, ex)
    planeA = normal[0]
    planeB = normal[1]
    planeC= normal[2]
    planeD = −np.dot(normal.reshape(1,3),T)
    return (planeA,planeB,planeC,planeD)
```

4.3 Construct Perspective Projection Matrix

Once the plane parameters have been found, it is necessary to find the perspective projection matrix for the left hand camera. Thankfully, the SDK provided by Rethink Robotics for the Baxter contains information on the intrinsic camera parameters. Thus, to form the extrinsic parameters, one need only find the transform from the base frame to the camera frame. This is easy to do in ROS:

```
Rcamera, Tcamera = getTransform(listener,'
    left_hand_camera','base')
Rright,Tright = getTransform(listener,'base','
    right_hand')
planeA,planeB,planeC,planeD = getPlaneParams(
    Rright,Tright)
P = getPFull(Rcamera,Tcamera)
```

Once the T and R are found, the perspective projection matrix can be constructed:

```
# Given the inrinsic parameters of a camera and
    the arm it is attached to
# find the projcetion matrix
# default W is for left camera frame
def getPFull(R,T, W = np.array
    ([[406.00431,0,590.6386],[0,406.0743,420.9455],[0,0,1]])
   ):
    M = np.append(R,T.reshape(3,1),1)
    print M
    P = np.dot(W,M)
    return P
```

4.4 3D Reconstruction

Equation 4 is then applied using the newly acquired perspective projection matrix, plane parameters and laser line pixels to find X,Y and Z in the base frame.

$$\begin{bmatrix} -P_{11} + \frac{A*P_{13}}{C} & -P_{12} + \frac{B*P_{13}}{C} & c \\ -P_{21} + \frac{A*P_{23}}{C} & -P_{22} + \frac{B*P_{23}}{C} & r \\ -P_{31} + \frac{A*P_{33}}{C} & -P_{32} + \frac{B*P_{33}}{C} & 1 \end{bmatrix}^{-1} \begin{bmatrix} \frac{D*P_{13}}{C} + P_{14} \\ \frac{D*P_{23}}{C} + P_{34} \\ \frac{D*P_{33}}{C} + P_{24} \end{bmatrix} = \begin{bmatrix} X \\ Y \\ \lambda \end{bmatrix} \qquad (4)$$

where Z is solved for using the plane parameters:

$$Z = \frac{D + AX + BY}{-C} \tag{7}$$

5 Results

By running the code with the lights on, we are able to visualize the laser plane that is calculated. An image of the scene with the lights on, with no particular configuration set up, is shown in the figure below.

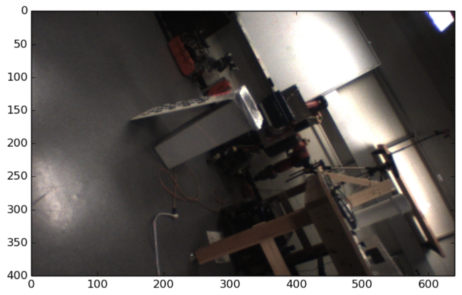


Figure 2: Camera View with Lights on

The resulting mask while the lights are on results in all the bright locations in the scene as white, while dark locations remain black, as shown in the image below.

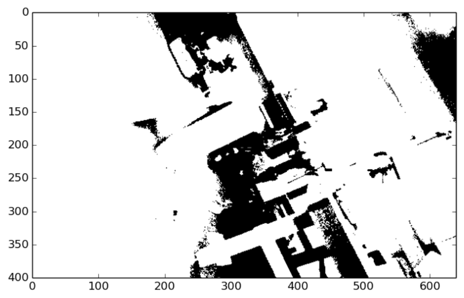
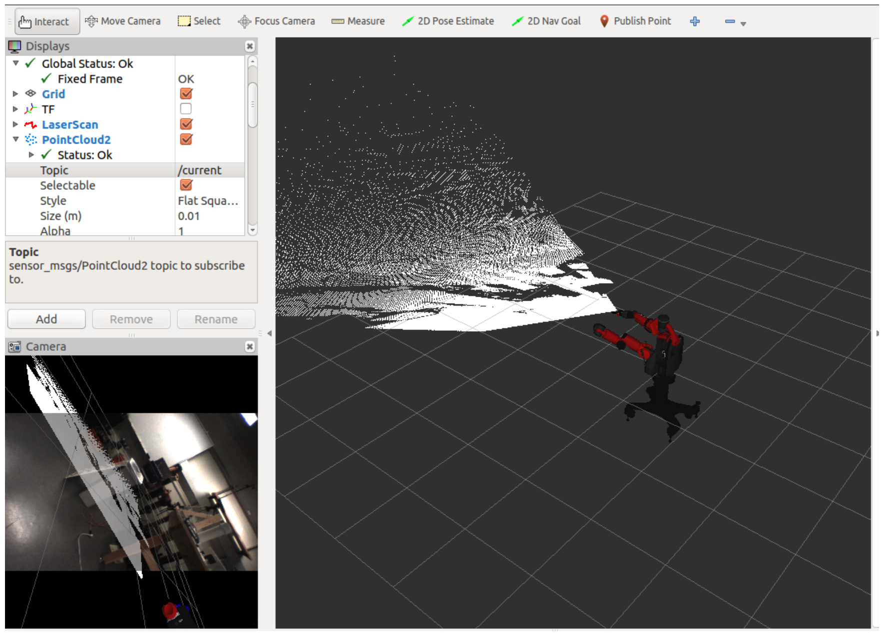


Figure 3: Mask Created with Lights on

Although the mask highlights points in the image which does not correspond to the laser projection, by creating a point cloud using the above mask, the laser plane can be visualized, as shown in the image below.



Global Status: Ok
Fixed Frame OK
Grid
TF
LaserScan
PointCloud2
Status: Ok
Topic /current
Selectable
Style Flat Squa...
Size (m) 0.01
Alpha 1

Topic
sensor_msgs/PointCloud2 topic to subscribe to.

Figure 4: Visualization of Laser Plane

By turning off the lights, the only thing visible in the image was the laser line projected onto the object. The resulting laser projection on the intersecting point between the planes (a trash can and a poster board) is shown in the figure below.

By only considering the red channel of the image and applying a threshold, the resulting mask is created. The corresponding row and column values for each white pixel is stored. When doing the 3D reconstruction, a sample of these points are used, depending on an assigned sample rate.

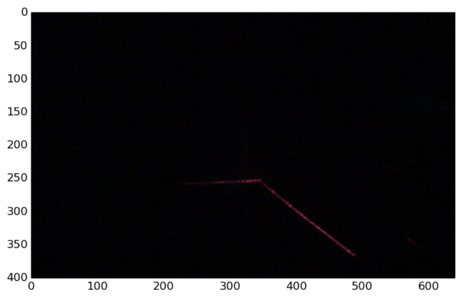


Figure 5: Laser Line projected on Object

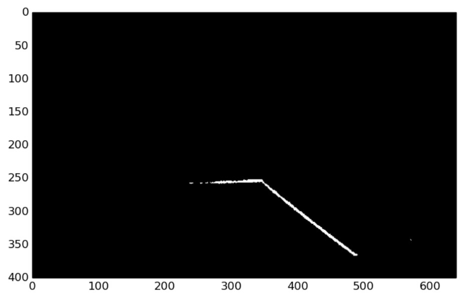
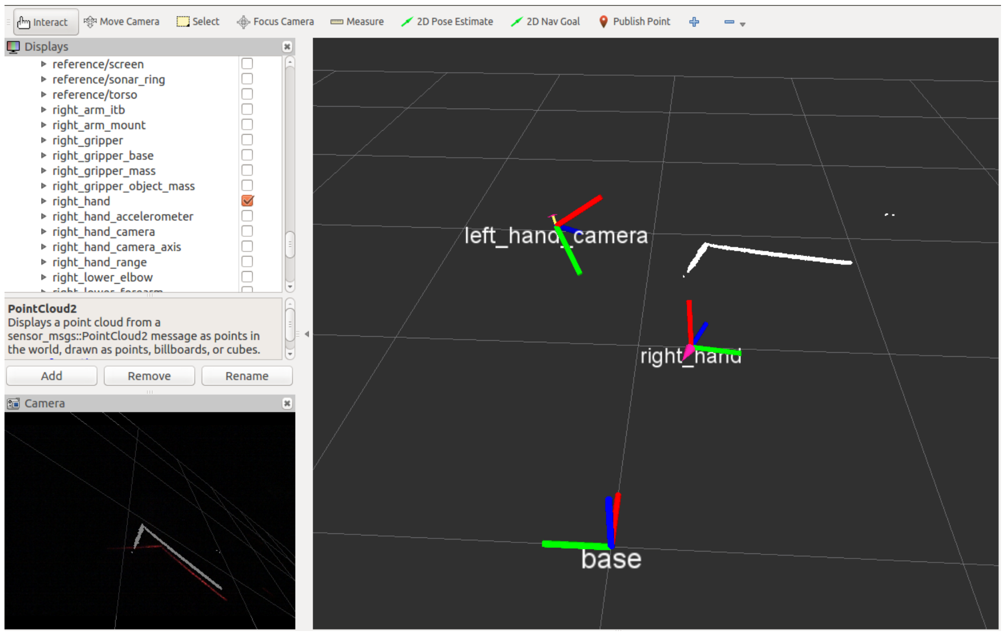


Figure 6: Mask from Threshold on Red Channel

The corresponding visualization of these 3D points is shown using RViz.



PointCloud2
Displays a point cloud from a sensor_msgs::PointCloud2 message as points in the world, drawn as points, billboards, or cubes.

Figure 7: RViz Visualization of Reconstructed points

Lastly, a reconstruction test was run on a simple scene where scans were aggregated. The scene used in this test is pictured in Figure 8 below.

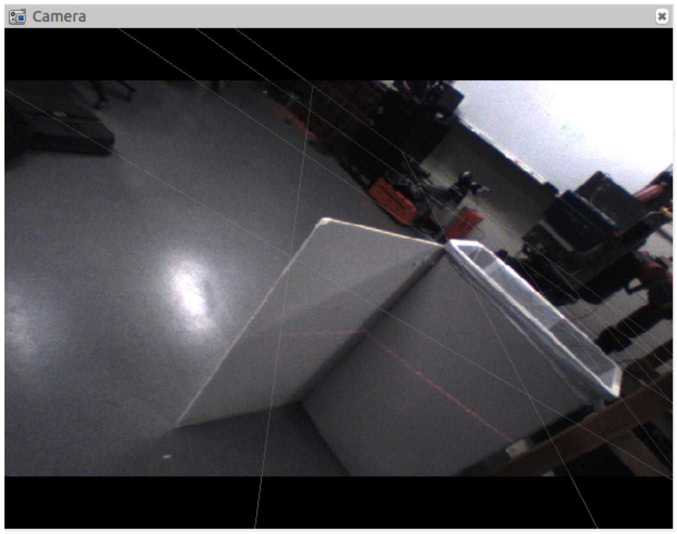


Figure 8: Image of scene to reconstruct

The aggregated point cloud, created by manually moving the laser, is shown in the figure below.

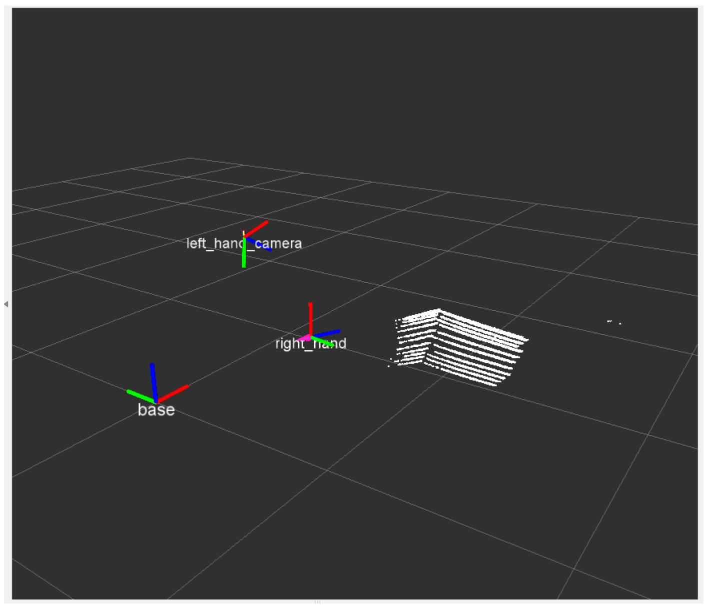


Figure 9: RViz Visualization of Reconstructed scene

6 Conclusion

The 3D reconstruction of the scene seemed to be accurate as the shape and relative orientation/distance was sensible. Additionally, when viewing the results from the camera, the 3D points were close to where the laser line was. This result is significant because the 3D points (dots) in the camera image are RViz's interpretation of what the camera would see if the 3D points were present in the actual scene. Thus, it is good that the 3D reconstructed points are close to overlapping with the laser line. While we were able to obtain decent results when reconstructing the intersection point of two planes, this was not generally the case for most of the objects we had scanned. The 3D reconstruction points were computed relative to the robot base frame, where the ROS tf package was used to transform frames between the end effectors and the base. Technical difficulties arose, however, when rotating the camera, resulting in scale distortion in the projected points.

A possible solution may be to calculate the points relative to the camera frame, or to use calibration to reduce the error in the laser plane equation. A optimization approach could also be attempted to find the best plane parameters. Furthermore, in order to scan multiple laser lines, the laser end effector was move either by hand or using an XBox controller (through the ROS joy node). While the position and orientation of both the camera and the laser are dynamically computed each iteration, this manual shift of the laser resulted in slight errors, which could be reduced by automating the scanning process.

References

[Ahuja and Abbott, 1993] Ahuja, N. and Abbott, A. L. (1993). Active stereo: integrating disparity, vergence, focus, aperture and calibration for surface estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(10):1007–1029.

[Diebel et al., 2004] Diebel, J., Reutersward, K., Thrun, S., Davis, J., and Gupta, R. (2004). Simultaneous localization and mapping with active stereo vision. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 4, pages 3436–3443 vol.4.

[Geiger et al., 2011] Geiger, A., Ziegler, J., and Stiller, C. (2011). Stereoscan: Dense 3d reconstruction in real-time. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 963–968.