Problem Statement:

Given a rosbag containing the output from a intel RealSense camera, process the data from the ros topics in the rosbag to detect the sidewalk, and output the following.

1. RGB image with sidewalk highlighted,
2. PointCloud with points on the sidewalk
3. PointCloud with points outside the sidewalk

Environment Setup and Learning Technologies:

Installed ROS in my Fedora 21 system (ROS really hates Fedora). Installed the required libraries (OpenCV, PCL). Learned the basics of ROS, and PCL. Developed an understanding of the RealSense data (found out its stereo and not structured light). Started working on an algorithm.

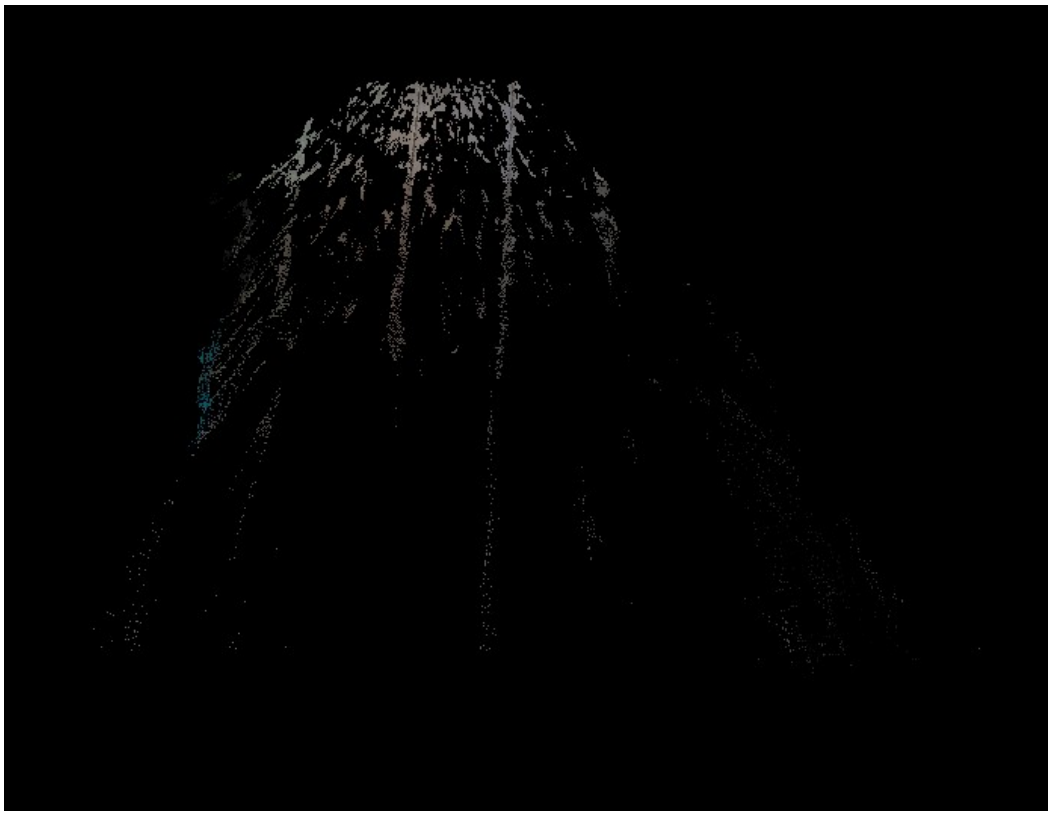
Implementation Details:

The detection algorithm is implemented in c++. The python script in the scripts folder is for testing purposes and does not constitute the node.

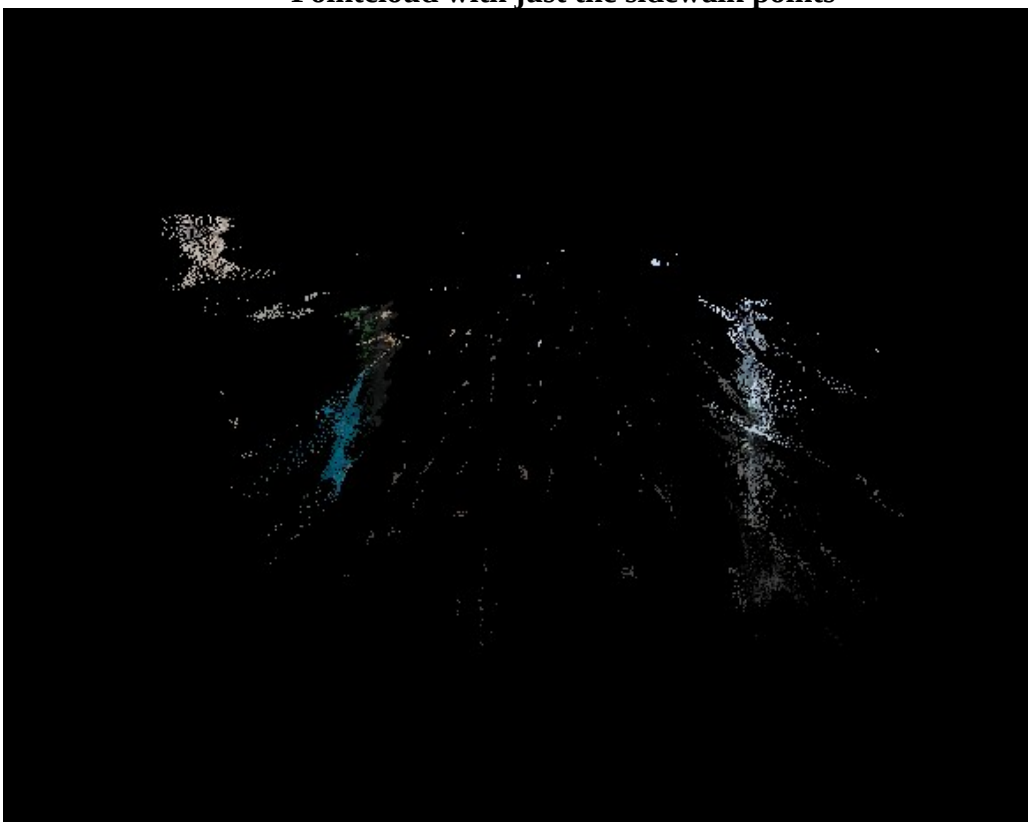
First the node is initialized and is set as a subscriber to the following topics.
/camera/depth/points
/camera/color/image_raw

A message time synchronizer is used to get the messages from the two topics and provide it to a common callback. The callback function gets the PointCloud and the raw image. The pointcloud message is converted to an PointXYZRGB type and the image message is converted to a opencv MAT pointer. The detection algorithm has the following phases.

1. **Extract the plane containing the sidewalk:**
   A sequential RANSAC algorithm is used to fit planes to the points in the point cloud. For each frame I got 2 – 4 planes that fit the data. The plane of the sidewalk is selected by picking the plane that has the smallest angle to the camera y-axis. This is because the camera y-axis is facing the sky and the normal to the sidewalk plane also faces the sky. Once this plane is found, the inliers indices in the original pointcloud are extracted. The points are separated into two different pointclouds, one containing the sidewalk the one without the sidewalk. The pointclouds the shown below.

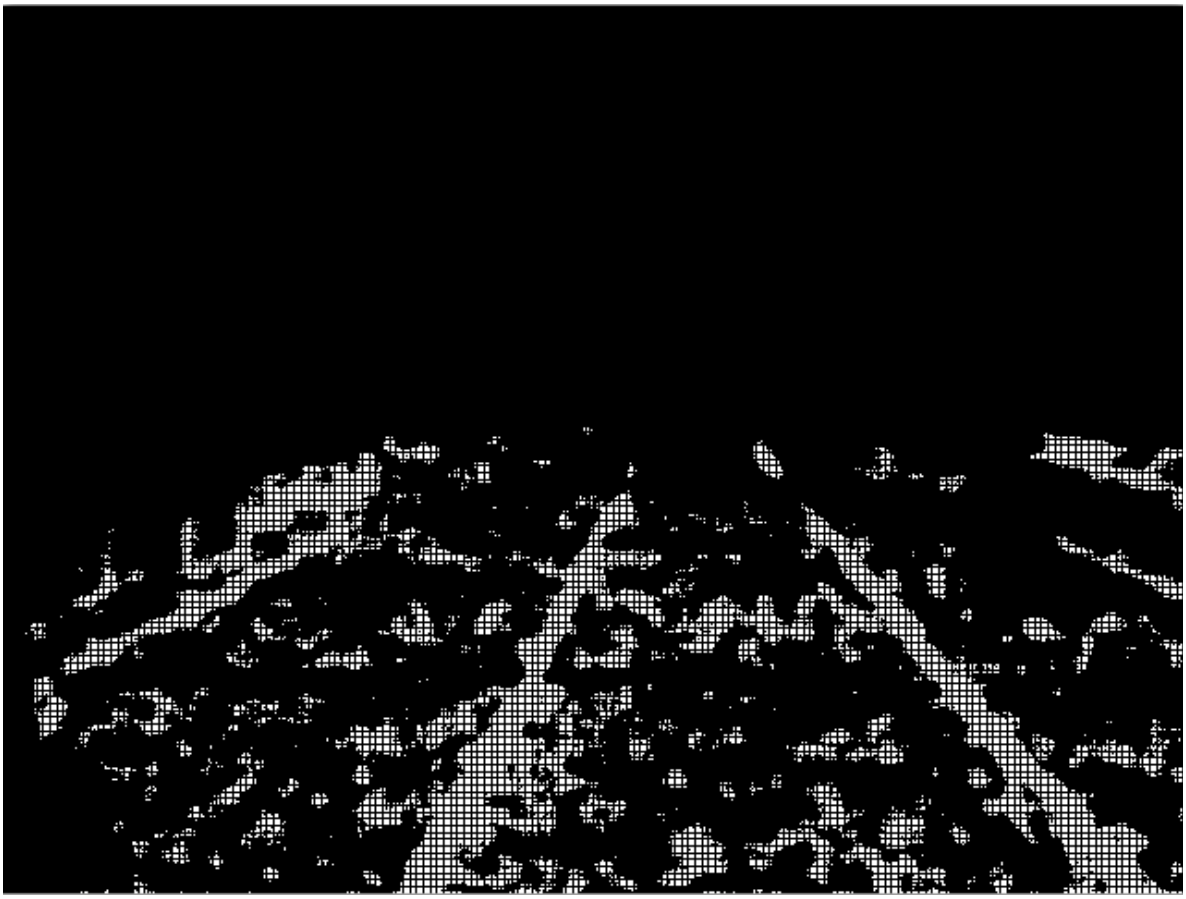


Pointcloud with just the sidewalk points



PointCloud without the sidewalk

2. **Highlighting the sidewalk in the RGB image:**
   Once I had the points belonging to the sidewal, I projected the 3D points onto a flat image using the color camera projection matrix. The color camera projection matrix is present in the "/camera/color/camera_info" topic. I used this image with the 3D point projections as a mask image to build the highlighted area. The Mask Image is shown below.



Projection of 3D sidewalk points onto image frame

The projection matrix is hard coded in this implementation, and the value is found by echoing the messages from the camera info topic. The grid like appearance in this image is due to the rounding error when the 3d world coordinated which are in float are converted to pixel coordinates which are integers.

This Mask is then filled by using a fill technique that is similar to a convex hull. The outer most white pixels are connected in both the x direction and the white direction. This results in a mask with small gaps and full of lines in the interested regions. A flood fill technique is then used to fill in the gaps. This mask is then used to highlight the corresponding pixel in the RGB image. The result is as shown below.

3. Conclusion and Possible Improvements.

My algorithm finds the sidewalk (within some error) as long as there is accurate depth information. In the data provided there are some frames in the middle where the depth information is absent and my technique fails during those frames.

The detection can be improved by doing the following.

- The current implementation is state less. Each frame is processed as independent without the knowledge of previous frames. By making this a statefull algorithm and using a Kalman type filter in the time domain to predict the ground plane will give better results.
- The detection is only dependent on the depth information. A hybrid technique using the color and edge data in the RGB image will greatly improve the detection.
- Filtering the pointcloud to remove unwanted speckles will improve plane fitting by RANSAC as we can use a tighter distance threshold.