A 3D Object Labelling Tool

Conceptual Report

Team Probots: I. Monteiro, J. Piazza, X. Zhang

Problem

There is a lack in the market of an open source tool capable of annotating 3D point clouds. Point cloud annotations serve as ground truth for many applications, including self-driving technology. This project serves to fill this gap.

Market Review

Point cloud annotation tools are generally either proprietary, or if open-source, are intended for applications other than the self-driving realm, such as geomodelling. A variety of commercial applications exist for performing annotations; many of these companies also offer in-house annotation services on provided data. One such popular option is Playment[1]. An interesting open-source solution is provided out of the Stanford Computational Vision and Geometry Lab [1]. Their MATLAB-based tool involves orienting a Google Warehouse model to match the object orientation in the camera image, and then clicking through a series of point correspondences between the model and the object in the image.

Technology

Our application uses the below technologies:

OS	Windows or Ubuntu, with ROS Kinetic
Application	Python 3.5+
3D Visualization	Open3D [2]
Helper Packages	PySimpleGUI, tkinter

Methodology

The first step in the annotation process involves converting `velodyneScan` messages from pre-recorded `.bag` files into a suitable format, the most common of which is `.pcd`, although other formats are supported by the tool. The steps to do so are not obvious and are summarized below, and must be completed before using the annotation tool. Each command must be run in a separate terminal.

	Command	**Output**
1	`roscore`	–
2	`rosrun nodelet nodelet standalone velodyne_pointcloud/CloudNodelet`	–
3	`rosbag record /velodyne_points [-l ⟨num_messages⟩]`	–
4	`rosbag play ⟨bag_file⟩ --topic /velodyne_packets`	New .bag file with ⟨num_messages⟩ of /PointCloud2 messages
5	`rosrun pcl_ros bag_to_pcd ⟨new_bag_file⟩ /velodyne_points /⟨output_folder⟩`	Series of .pcd files in the ⟨output_folder⟩

[1]https://playment.io

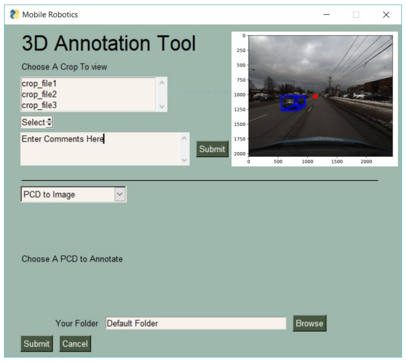


Figure 1: Initialization of the GUI.

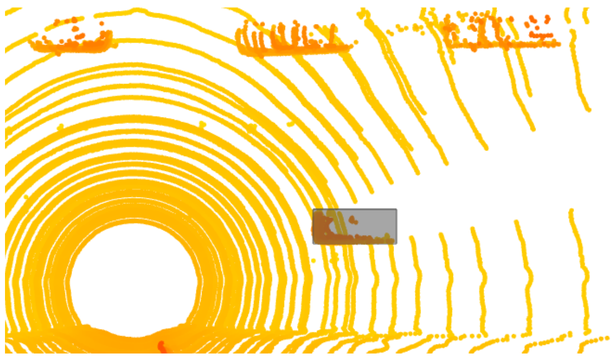
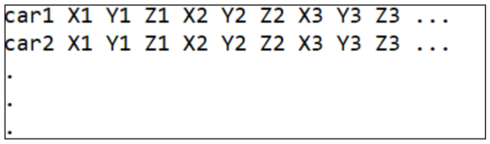


Figure 2: A sample `.pcd` file with bounding box, looking down the Z axis. The forward direction is to the right.

Upon launch of the application, users are greeted with the interface in Figure 1. Users can then browse to select the appropriate `.pcd` file they want to annotate. With the help of Open3D, the `.pcd` file is opened and the scan can be viewed from a variety of angles using the mouse, to get a sense of the forward direction of the vehicle. When ready, the `Z` key is pressed to lock the orientation of the scan facing downwards along the Z axis. Pressing `K` indicates that you are ready for annotating. Using a simple drag and select technique, any car can be bounded by the blue box that appears. Remember that only those vehicles in the field of the view of the mounted camera would be relevant. The bounding box can be redrawn at any time by re-clicking the screen. Pressing `C` saves the current bounding box. Figure 2 shows a typical scan and user-drawn bounding box. Bounding only in the X and Y directions allows the application to assume a standard height for cars.

Annotations are saved in a text file, with the eight vertices of the bounding cube being stored in the 3D coordinates of the camera frame, not velodyne frame, as seen below. These coordinates can easily be projected onto the image plane by pre-multiplying by the camera matrix. Tracking annotations between scans is currently not in scope; each text file represents one `.pcd` file.



```
car1 X1 Y1 Z1 X2 Y2 Z2 X3 Y3 Z3 ...
car2 X1 Y1 Z1 X2 Y2 Z2 X3 Y3 Z3 ...
.
.
.
```

Figure 3: A sample annotation, stored as a text file.

Upcoming Work

The viability of such an annotation tool depends upon the ease and quickness with which a user can create a reliable dataset. The tool will need to be improved so as to allow the user to annotate multiple vehicles in the same frame before saving. Additionally, the GUI will be updated and adequate `help` information will be provided to reduce the learning curve of the tool.

References

[1] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. "Beyond PASCAL: A Benchmark for 3D Object Detection in the Wild". In: *IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2014.

[2] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. "Open3D: A Modern Library for 3D Data Processing". In: *arXiv:1801.09847* (2018).