A Point Cloud Semantic Segmentation Framework for Embedded Systems in Agricultural Robots

Gary Storey
Department of Computer Science
Loughborough University
G.Storey@lboro.ac.uk

Lei Jiang
Department of Computer Science
Loughborough University
L.Jiang2@lboro.ac.uk

Qinggang Meng
Department of Computer Science
Loughborough University
Q.Meng@lboro.ac.uk

Loughborough University

Introduction

We present a novel multi-step framework for **Point Cloud Semantic Segmentation** from **RGB-D** data.

Aim: To perform accurate sematic segmentation of trees, obstacles and flat terrain to aid agricultural navigation and path planning for wheeled field robots.

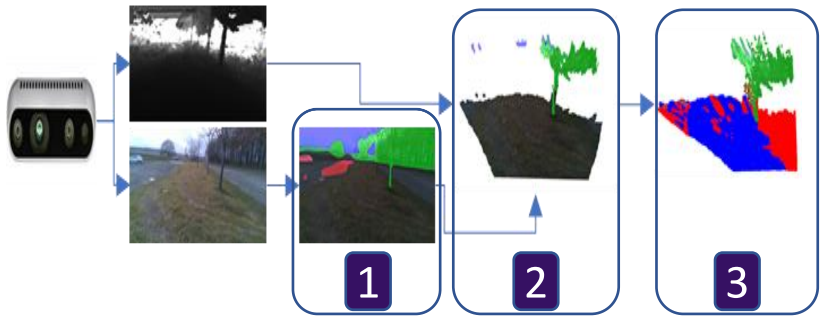
Challenges:

- Capable of running on an embedded device at multiple frames per second.
- Terrain types, i.e. gravel, mud, grass can have uneven surfaces. It is important to identify the ground and also areas which may unsuitable to traverse.
- To detect potential obstacles that are missed when using RGB sematic segmentation alone.

Target Hardware: The Nvidia Jetson TX2, with data capture from an Intel RealSense D435 RGB-D stereo depth camera.

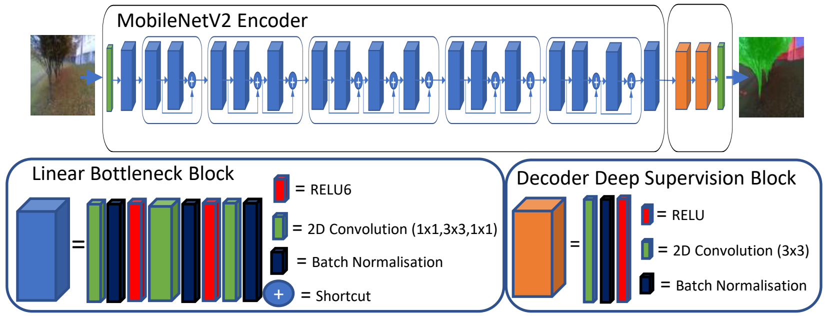
Implemented Using: Python 3.6 with PyTorch, Intel® RealSense™ SDK 2.0 and Open3D software libraries.

Framework Overview



Step 1: RGB Semantic Segmentation

A **MobileNetv2** encoder with dilated convolutions in tandem with a layer decoder is used. Model training used the **ADE20K** dataset.



Step 2: Create Point Cloud

The RGB sematic segmentation output and corresponding depth image create the segmented point cloud as:

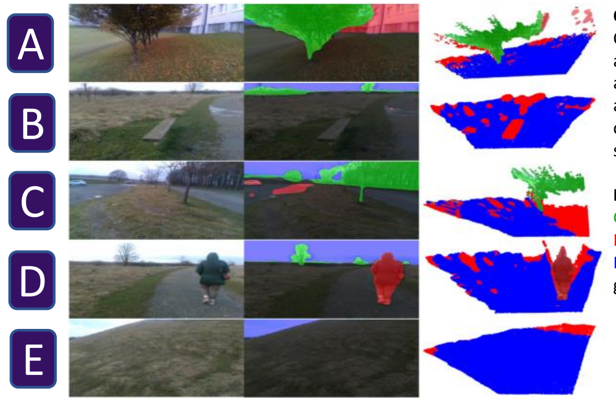
$$x_i = (u_j - c_x) * z/f_x$$
$$y_i = (v_j - c_y) * z/f_y$$
$$z_i = d/\lambda$$

where u_j and v_j are the jth pixel locations in the depth image. c_x, c_y, f_x and f_y are the cameras intrinsic parameters. Each point in the cloud is a vector given by $\{x_i, y_i, z_i\}$ and an associated colour as $\{r_j, g, b_j\}$ taken from the jth pixel locations of the RGB sematic segmentation output.

Step 3: Refine Point Cloud Segmentation

A. **Down-sample point cloud** to reduce computational demands by first removing points that are predicted as trees and sky due to high confidence in these detections from Step 1. Then uniform nearest neighbour down-sampling is then applied.
B. **RANSAC algorithm** is applied to the point cloud to perform plane segmentation to refine flat areas and obstacle detection.

Output Examples



Overview: Qualitative analysis of output at both the RGB and final point cloud sematic segmentation.

Key:
Green = Tree.
Red = Obstacle.
Blue = traversable ground.

Computational Evaluation

Step	1 Meter	5 Meters	10 Meters
1	0.064s	0.064s	0.064s
2	0.017s	0.034s	0.038s
3A	0.018s	0.049s	0.046s
3B	0.008s	0.018s	0.021s
Total	0.11s	0.16s	0.17s

Overview: Point Cloud depth cropped at 3 distances and testing carried out on target hardware, with a total of 50 frames, the mean values are presented in seconds.

Conclusion

Using RGB and point cloud methods leverages the strengths of each domain. Accurate classification for many object types in the RGB domain can be refined using depth data in the point cloud, examples include:

- Detecting missed obstacles from the RGB domain such as a bench in example B.
- Distinguishing between flat and uneven terrain following RGB prediction which can find all potential terrain.

Runs at up to 9 frames per second on the target hardware providing good computational performance on hardware that is relatively cheap at £600.

Acknowledgement: This research was funded through Innovate UK (Project 104016)