ReadMe

Steven Spiegel

2025-12-05

Capstone

Introduction

The goal of this project is to segment point cloud using two different methods for comparison. The first method will use a pointwise radial neighborhood method, where neighboring points will be within a radius r of a point, p_0. The second method will use a cylindrical nieghborhood method. This study will apply a random forest classifier to point features derived from these two neighborhood methods. Multiple radii will be used to capture additional geometric context. A total of 6 scales will be used with a voxelized downsampling to maintain computational efficiency.

Neighborhoods

This study focuses on radial and cylindrical neighborhoods on a terrestrial LiDAR (light detection and ranging) scan. Cylindrical neighborhoods are mostly used for aerial scans, while radial neighborhoods and k nearest neighborhoods are primarily used on mobile or terrestrial data. This study will test the two radial neighborhood types on a mobile LiDAR scan. [Thomas et al]((https://ieeexplore.ieee.org/document/8490990) demonstrated that geometrically uniform neighborhoods like radial neighbors give better results than a *k nearest neighbors* approach, while *k nearest neighbors* are less computationally expensive. Let C be a point cloud. Note that

- $C \subset \mathbb{R}^3$ is a finite subset
- A pont $p \in C$ is defined as $p = (x, y, z)$

In other words, p is a point in 3 dimensional space that represents the surface of some real world object. A neighborhood N of point p_0 is defined as the set of all points $p \in C$ that satisfy some neighboring criteria.

Radial Neighbors

Let $r \in \mathbb{R}$ and $p_0 \in C$. Radial neighborhood are then defined as

$$N(p_0) = \{p \mid \|p - p_0\| \leq r\}$$

Cylindrical Neighbors

Cylindrical nieghborhoods are commonly used in aerial scans. Let $h \in \mathbb{R}$ be the height of the cylinder and $r^{(\text{cyl})}$ be the radius of the cylinder. Cylindrical neighbors are then defined as

$$N(p_0) = \left\{p = (x, y, z) \mid z \leq z_0 + \frac{h}{2} \wedge z \geq z_0 - \frac{h}{2} \wedge \sqrt{(x - x_0)^2 + (y - y_0)^2} \leq r^{(\text{cyl})}\right\}$$

NOTE: This assumes that the center line of the cylinder can be parameterized by the unit vector, $e_z = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$.

Point Features

Neighborhoods become important when computing point features. This is done by computing neighborhood points and using Principal Component Analysis (PCA). For a point $p_0 \in C$ and neighborhood $N(p_0)$,

$$\text{Cov}(p_0) = \frac{1}{|N|} \sum_{p \in N} (p - \bar{p})^T \cdot (p - \bar{p})$$

We use the eigenvalues (λ_1, λ_2, λ_3) and corresponding eigenvectors (v_1, v_2, v_3) to compute geometric properties of the point cloud. By convention, we assume

Feature	Equation
Eigen Sum	$\sum_i^3 \lambda_i$
Omnivariance	$(\prod_i^3 \lambda_i)^{\frac{1}{3}}$
Shannon Entropy	$-(\sum_i^3 \lambda_i log(\lambda_i))$
Linearity	$\frac{\lambda_1 - \lambda_2}{\lambda_1}$
Planarity	$\frac{\lambda_2 - \lambda_3}{\lambda_1}$
Sphericity	$\frac{\lambda_3}{\lambda_1}$
Verticality 1	$\|\frac{\pi}{2} - cos^{-1}(v_1 \cdot e_z)\|$
Verticality 2	$\|\frac{\pi}{2} - cos^{-1}(v_3 \cdot e_z)\|$
Height Variance	$\frac{1}{N}\sum(H_i - \bar{H})^2$
Height Range	$H_{\max} - H_{\min}$

In order to capture additional geometric context for point p_0, we use multiple radii and downsampling scales for computational efficiency. We use 6 scales with 12 features at each scale, totalling 72 features per point.

Classifier

We apply a random forest classifier trained on iterative subsets of the training data to keep greater class balance. The classifier is validated on a mutually exclusive subset of the data and a subsample of points are added to the classifier and retrained.

Getting started

Operating System

The code in this repo assumes the use of Ubuntu 22.04.5 LTS operating system.

Python code

We use conda for package management. Requirements can be found in `./environment.yml`. To install with conda from a terminal window, run

```
$: conda env create -f environment.yml
```

Datasets

Download the Lille dataset here. NPM3D is a benchmark LiDAR mobile dataset with annotated labes: unclassified (ignored in classifier and results), ground, building, signage, bollard, trash can, barrier, pedestrian, car, and vegetation. The code base saves it in `~/CapstoneData/Paris`. The file `Lille1_1.ply` is used for training and `Lille2.ply` is used for testing.

Dataset Pathing

Some of the folder paths may need to be changed. Point clouds are assumed to be stored in `~/CapstoneData/Paris/training_10_classes/`. Random forest models are saved in `~/CapstoneData/Paris/RF_models/`. Point features are saved in `~/CapstoneData/Paris/training_10_classes/pickleFiles` with the below folder structure.

```
.
   cylinder
      testing
      training
|
   radial
       testing
       training
```

Python modules

Custom Python functions for computing point features and classes can be found in `./Capstone/PythonScripts`. This is the code that computes point features and the custom `Cylinder` class for computing cylinders.

Point Feature Extraction

Point features are calculated in order in jupyter notebooks. They can be found in `./analysis/radial_Paris` and `./analysis/cylindrical_Paris`.

Random Forest

Training Random forest classifier can be found `./RF`. It's split into `radial` and `cylindrical` folders. To train radial method, use

`./RF/radial/01_Lille_Radial_multiTuning.ipynb`

For testing, use

`./RF/radial/02_Lille_Radial_multiTuning-Testing.ipynb`

To train cylindrical method, use

`./RF/cylindrical/01_Lille_Cylinder_multiTuning.ipynb`

For testing, use

`./RF/cylindrical/02_Lille_Cylinder_multiTuning-Testing.ipynb`

Results can be found in the `results` for both methods.