MCEN90032 Sensor Systems

Workshop 3(a): LIDAR Data Processing

Prepared by: Nandakishor Desai and Marimuthu Palaniswami

1 Introduction

Lidar is one of the most common sensors used on self-driving cars and many other kinds of mobile robots. It comes in many different shapes and sizes and can measure the distances to a single point, a 2D slice of the world, or perform a full 3D scan. Its job is to provide detailed 3D scans of the environment around the vehicle. It determines ranges (variable distance) by targeting an object with a laser and measuring the time for the reflected light to return to the receiver. The LIDAR data is subsequently used by the vehicle in its perception and localization modules to perform further porcessing and understand its immediate surroundings.

The project is separated in three parts. In part 1, you will write simple MATLAB code to construct and visualise basic LIDAR data structure. Part 2 involves performing spatial operations to understand pointcloud transformations and implement the iterative closest point algorithm to solve the point set registration problem. In part 3, you will develop a simple machine learning LIDAR object recognition module that operates on LIDAR pointclouds and classifies them to one of the predefined object categories.

- The workshop assignment is worth **9 marks**.
- You will need to complete and submit this work individually.
- The report is due on Saturday (14/05/2022) at 11:59 PM and is **5 marks**. Report submissions are made through LMS.
- The demonstrations are due by (16/05/2022 during week 11). You need to explain the working of your code to demonstrators and answer their questions. It is worth **4 marks**.
- Note that you cannot modify reports after 14/05. You may just demonstrate the code during the sessions on 16/05 and then move on to project 3(b).

1.1 Learning Objectives

- Understand LIDAR pointcloud data structure and storage management.
- Determining transformations between two LIDAR pointclouds to understand the vehicle movement.
- Using LIDAR pointclouds to recognise the objects surrounding the vehicle.

1.2 Component Required (minimum)

1. A MATLAB account, if you don't have one please go to https://matlab.mathworks.com/

2 Project Instructions

Part 1: LIDAR Data Structure

LIDARs measure the position of points in the three-dimensional world using spherical coordinates. Range or radial distance from the centre origin to the 3D point, elevation angle measured up from the sensors XY plane, and azimuth angle, measured counterclockwise from the x-axis of the sensor. The spherical coordinates can be converted to cartesian coordinates x, y, and z and stored as a 3 by 1 column vector. Subsequently, the points can be stacked into a matrix that stores the scanned points.

1.1 A vehicle equipped with a LIDAR sensor is starting its travel from a location A. The LIDAR sensor scans the 3D world and captures the information according to the following distribution:

$$z = \begin{cases} x^2 + y^3 & \text{if } 0 \le x \le 1 \text{ and } 0 \le y \le 2 \\ 0 & \text{otherwise} \end{cases}$$

MATLAB stores 3D LIDAR scan points in a data structure **pointCloud** consisting of the cartesian coordinates of the scanned points and associated metadata. Write a MATLAB script to construct the **pointCloud** data structure by sampling 1000 values of each of x and y. Describe its properties and visualise the point clouds.

Part 2: Spatial Transformations and Pointcloud Registration

Objects in the world mostly stay put while the reference frame attached to the vehicle moves and observes the world from different perspectives. When the vehicle moves, it will affect the perception of all the scanned points in the point cloud. The vehicluar movement between the points can be described by a combination of translation and rotation operations. Given two point clouds in two different coordinate frames, and with the knowledge that they correspond to or contain the same object in the world, optimal translation and rotation between the two reference frames can be computed to minimise the distance between the two point clouds. A popular algorithm to solve the point registration problem is the iterative closest point (ICP) procedure. The basic intuition behind ICP is that, when we find the optimal translation and rotation and we use them to transform one point cloud into the coordinate frame of the other, the pairs of points that truly correspond to each other will be the ones that are closest to each other in a Euclidean sense.

2.1 Transform the LIDAR point cloud from 1.1 using a transformation matrix with a 60° rotation about z-axis and translation of 2 and 4 units along x- and y-directions, respectively. Visualise the original and transformed pointclouds.

2.2 Apply ICP to the transformed point cloud in 2.1 to determine the optimal translation and rotation matrices to align the two point clouds. This alignment best describes the movement of the vehicle between the two points. You can assume that the LIDAR sensor is ideal and measures the entire point cloud all at once.

2.3 Compare the translation/ rotation values obtained in 2.2 with the values in 2.1 and comment.

Part 3: LIDAR-based Perception Module: Object Recognition

Autonomous vehicles rely on their perception systems to obtain information about their immediate surroundings. The perception system is primarily tasked with detecting and identifying the objects around the vehicle. Once the presence of an object is detected from the LIDAR pointcloud, the subsequent step involves recognising the content of pointcloud by classifying it into one of several predefined categories such as pedestrians, bicycles, trees, buildings, and others. A widely used approach to object recognition involves developing a supervised machine learning algorithm that can learn from the input (pointcloud) and output (category name) pairs. A typical machine learning recognition process employed comprises a feature-extraction step, calculating compact object descriptors, and a classification step, where pretrained classifiers predict the categories of objects based on the extracted features.

In this task, you will design and implement a simple machine learning-based object recognition algorithm that can operate on LIDAR pointclouds to classify them into one of the predefined categories. You will use a dataset consisting of 523 individual LIDAR scans of objects across classes of buildings, cars, pedestrians, etc.. This dataset was acquired using a Velodyne HDL-64E LIDAR, collected in the CBD of Sydney, Australia. The datasets are available in two '.mat' files: *lidarData.mat* and *lidarLabel.mat*. The *lidarData.mat* is a cell array consisting of 523 elements corresponding to 523 LIDAR scans of different objects. Each element of the cell array is a 2D matrix with 4 columns (x,y,z coordinates of a world object and the fourth column represents intensity A of the returned laser signal). The number of rows in each element varies and represents the number of 3D pointsclouds obtained for the given scan. The *lidarLabel.mat* is a cell array consisting of 523 class names corresponding to the 523 LIDAR scans. These 523 scans belong to a total of 10 classes: *building, car, pedestrian, pillar, pole, traffic_lights, traffic_sign, tree, trunk and van.* Load the data and labels and inspect the content and dimensions before proceeding further.

Feature Extraction

The first step in developing a machine learning object recognition algorithm involves extracting compact feature descriptors that effectively represent the LIDAR pointclouds. The pointcloud features available in literature can be roughly divided into two classes: the global features for the whole object or the local features for each point inside the object pointcloud. The global features that do not involve any local computations can be called *object level features*.

One kind of object level features can be based solely on the intensity of the returned laser signal. Different objects with different characteristics and views reflect the laser signal differently. The intensity can provide rich information about the type of the scanned object. The intensity values (4th column in the LIDAR pointcloud matrix) can be used to extract simple features for our recognition task. One simple intensity feature could be the maximum of all the intensity values from a given pointcloud. It can be calculated as $A_{max} = max\{A_1, A_2, ..., A_N\}$, where N represents the number of points in the pointcloud. For example, the first entry in the dataset is a pointcloud (with dimensions 2294×4) of a building. The maximum intensity feature for this can be calculated as: $A_{max} = max\{A_1, ..., A_{2294}\} = 176$.

3.1 Extract three scalar features from the intensity values for all the pointclouds in the given data and store them into a 2D matrix.

Check
Your feature matrix after [3.1] should be 523×3

The intensity features alone cannot capture all the properties of the objects from the pointclouds. Features that can be extracted from the 3D coordinates(x,y,z) of the object can provide information on the physical structure of the object. A simple shape feature can be to compute the spread of pointcloud data in different directions. The amount of spread in different directions can be calculated using an unsupervised machine learning technique called principal component analysis. The spread directions are given by the eigenvectors of the covariance matrix after computing PCA and the amount of spread along each direction is given by the eigenvalues.

3.2 Compute a three dimensional shape feature that can describe the three largest amounts of spread along the three orthogonal directions.

Check
Your feature matrix after combining the shape features from [3.2] with the intensity features from [3.1] should be 523×6.

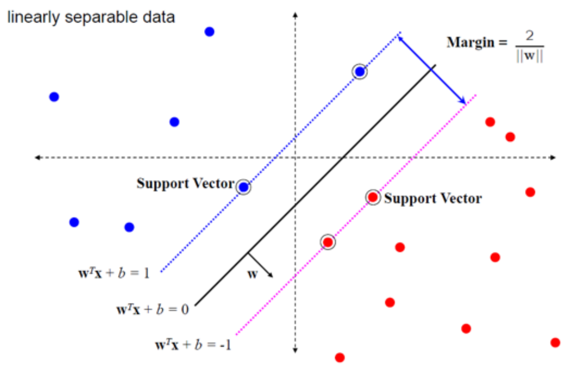
Classification

An important step before training a machine learning classifier is to split the dataset into training and test subsets. The training subset data and labels are used to train the classifier, which is then evaluated using the test subset data and labels to assess its classification performance. The split into training and test subsets is usually done using stratified sampling to ensure all the classes in the dataset are represented in both the subsets.

3.3 After 3.2, you have feature matrix of dimensions 523×6 and corresponding 523 labels represented as a cell array. Performa a stratified split to sample 70% of the data points into your training subset and remaining into test subsets. Verify that training and test subsets each contain LIDAR scans from all the 10 classes.

Check
Your training subset should have a 367×6 feature matrix and 367×1 class-label cell array. The test subset should have 156×6 feature matrix and 156×1 class-label cell array.

Final step in the object recognition task is to train the classifier using the training data and labels. A widely used classifier is support vector machine (SVM). The problem of classification using SVM is usually formulated for 2-class (binary) classification problem. An SVM classifies data by finding the best hyperplane that separates all data points of one class from those of the other class. The best hyperplane for an SVM means the one with the largest margin between the two classes. Margin is defined as the maximal width of the slab parallel to the hyperplane that has no interior data points. Binary SVM can then be converted to multi-class SVM by training K separate SVMs (one-versus-the-rest) for a K-class classification.


linearly separable data
Margin = $\frac{2}{\|\mathbf{w}\|}$
Support Vector
Support Vector
$\mathbf{w}^T\mathbf{x} + b = 1$
$\mathbf{w}$
$\mathbf{w}^T\mathbf{x} + b = 0$
$\mathbf{w}^T\mathbf{x} + b = -1$

3.4 Use the training feature matrix and corresponding class-label array from 3.3 to train a linear multi-class SVM. Evaluate this trained classifier using the test data and labels and determine the classification accuracy.

3.5 Retrain the classifier in 3.4 and determine test accuracy for different combinations of features as following (i) using only intensity features (drop the last 3 columns), (ii) now incrementally add one PCA feature column at a time. You will have 4 values of test accuracies after this trial. Inspect the test accuracies and explain the results and any specific observations that you can make.

Notes and tips

- You do not have to implement any of the algorithms from scratch and can use existing MATLAB commands.
- Pay attention to arguments inside a MATLAB command to configure your code for the task. For example, the SVM command would have options for you to choose a linear model.
- You may have to standardise your feature data.
- For Part-3, set a random generator using your student ID as the seed to ensure the code is reproducible.
- Refer to lecture 16 for LIDAR data structure, spatial transformations, and iterative closest point algorithm.
- Refer to tutorial 7 for relevant machine learning basics and developing a classifier for your LIDAR data.
- Please talk to your demonstrators if you are unclear on anything.

3 Project Report

Prepare a report as per the task given as a Portable Document (.pdf). Use an appropriate engineering report format. Your report should be no more than 12 pages with 12 pt font and it should clearly outline your methodology, results and discussion related to the tasks given in the project. You will be assessed on the following marking scheme for the report:

- Format: Is it in an appropriate engineering report format (include proper introduction, has figures and equations properly labelled, has references properly managed, etc)?
- Methods: Is the design clearly explained?
- Results: Did the implementation work? Is the functionality well assessed? How did you assess the functionality?
- Discussion: Is the performance and limitations clearly analysed and described? Are the results and analyses presented in a way that clearly support the effectiveness of the proposed scheme?

4 Submission

Submit your report as Portable Document (.pdf) and a zip file containing the raw data and two '.m' files. Name the report and zip file with your student ID and submit to Canvas before the due date. Late penalties apply at a rate of -10% (of the whole assignment marks) per day. Assignments submitted later than 5 working days after the due date will not be accepted and will receive zero marks. Only the last submission will be assessed.

5 Academic Integrity

We take academic integrity seriously. Please note that while students may discuss and help each other in the thinking process, you should work on your assignments separately. Details about academic integrity can be found at http://academicintegrity.unimelb.edu.au/. Please check with the tutors or the lecturer if you are in doubt. Ignorance is not a valid reason for academic misconducts.

References

[1] Himmelsbach, M., Mueller, A., Lüttel, T., & Wünsche, H. J. (2008, October). LIDAR-based 3D object perception. In Proceedings of 1st international workshop on cognition for technical systems (Vol. 1).

[2] Chen, T., Dai, B., Liu, D., & Song, J. (2014, June). Performance of global descriptors for velodyne-based urban object recognition. In 2014 IEEE Intelligent Vehicles Symposium Proceedings (pp. 667-673). IEEE.

[3] Wang, D. Z., Posner, I., & Newman, P. (2012, May). What could move? finding cars, pedestrians and bicyclists in 3d laser data. In 2012 IEEE International Conference on Robotics and Automation (pp. 4038-4044). IEEE.

[4] De Deuge, M., Quadros, A., Hung, C., & Douillard, B. (2013, December). Unsupervised feature learning for classification of outdoor 3d scans. In Australasian Conference on Robitics and Automation (Vol. 2, p. 1). Kensington, Australia: University of New South Wales.

[5] Quadros, A. J. (2013). Representing 3D shape in sparse range images for urban object classification.