Machine Learning Engineer Nanodegree Capstone Project Proposal

Classification of Objects in 3D Point Cloud Data

Stephan Roth

March 23, 2017

Domain Background

Images of things are no longer restricted to 2D pictures. Three dimensional images are becoming more available because of low cost commercial and industrial sensors. Sensors like the Microsoft Kinect allow anyone to create a 3D image of anything [Figure 1].

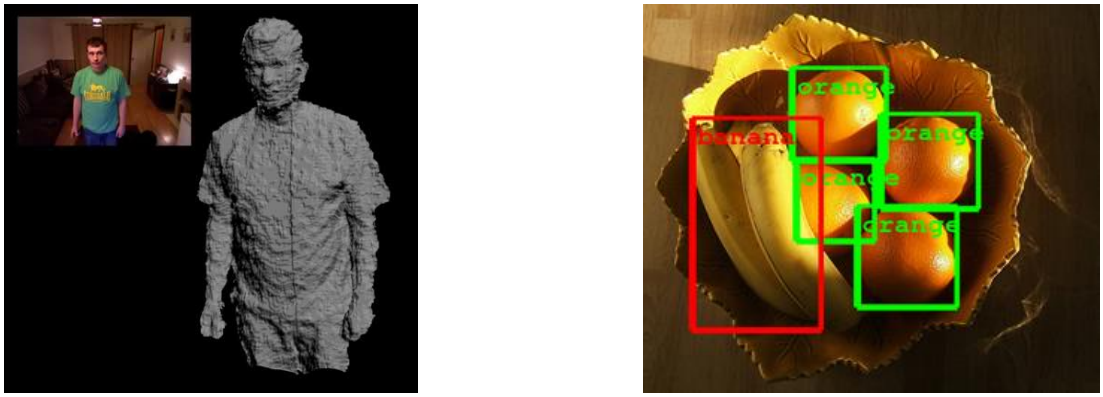

Figure 1: (left)3D Point cloud collected using a Microsoft Kinect (ref. http://123kinect.com/wp-content/uploads/2012/08/Kinect-1-point-cloud-example.jpg) (right) Google's object recognition algorithm (ref. https://www.engadget.com/2014/09/08/google-details-object-recognition-tech/)

Just like with 2D images, it is now possible to do object recognition on 3D structures. Google, Microsoft and other tech companies have long had the ability to take a picture of a scene and identify the objects it contains [Figure 1]. It is now possible to do the same thing with these new 3D images. This has applications to things like gesture recognition for next generation user interfaces or industrial applications such as identifying structures from 3D survey data.

One application of 3D object recognition is the automatic identification of structures in a building survey. Surveys are no longer limited to measuring distances to single points on a job site. It is now possible to capture highly accurate 3D structural data [Figure 2]. The data collected during one of these surveys is what is termed a point cloud. A structure is represented as a dense collection of points in (x,y,z) possibly with brightness and color information. However, just as with images, there is no label attached to each point. The point cloud by itself does not identify walls, chairs, roofs, etc. It would be very useful to have these objects automatically identified within a point cloud. That is the goal of this project, to train a classifier to recognize a 3-dimensional object within a point cloud.

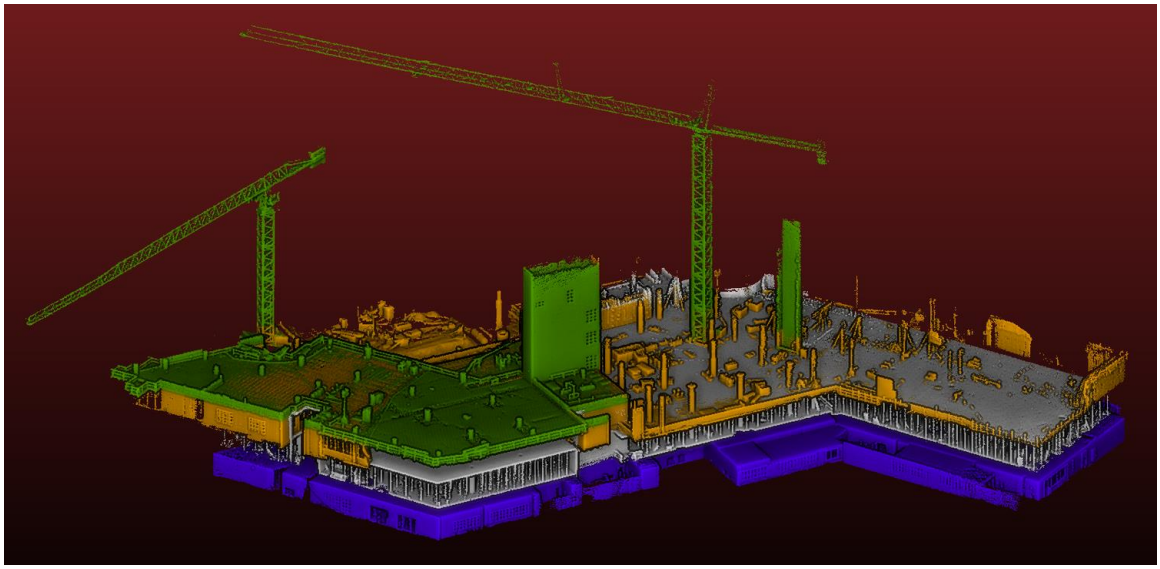

Figure 2: 3D Survey of a building under construction. The data is a 3 dimensional point cloud. In a point cloud, the building is represented by a dense cloud of (x,y,z) points, possibly with color or brightness information (Ref. kaarta.com/gallery)

Problem Statement

The purpose of this project is to create a supervised learning classifier to identify and locate an object within a 3D point cloud. Each point within the point cloud shall be classified as either belonging to the target object or not.

The data in a point cloud is simply (x,y,z,intensity) values. These by themselves are not suitable as inputs to a supervised learning classifier. A feature vector must be defined in such a way that it describes the point cloud geometry local to each point. An example of this kind of feature is the eigen values and eigen vectors of a covariance matrix. This is a standard geometric descriptor used in classifying point clouds in the literature ("Onboard Contextual Classification of 3-D Point Clouds with Learned High-order Markov Random Fields", Daniel Munoz, 2009). Using these features and a hand labeled data set, a supervised learning classifier will be trained to identify the target object. The classifier will be ranked according to its F1 score, and the performance will be verified on additional data sets that are not part of the training set.

Datasets and Inputs

For this project, a 3D point cloud will be collected with a flash LiDAR. The scene will include the target objects, which are 4 different pipes of diameters from 2" to ¾". See Figure 3 for an example scene and point cloud data. The point cloud will contain (x,y,z,intensity) data for the scene. In the scene will be the target object as well as walls, floor, etc. For training, this point cloud will be segmented by hand. This segmentation will define points which belong to the target object. The training data contains 1,427 points belonging to the pipe class and 14,749 points belonging to other type objects.

Each pixel in the 3D point cloud has a value (x ,y, z, intensity). This raw data cannot be used by the classifier because a single point does not contain enough information to describe what it is a point of. Instead, the features describing a point must come from a region of the point cloud. In this project, points in a cube of a given size will be used to calculate features describing the local geometry. The features used are common in analysis of point clouds. They use the eigen values and vectors of the

covariance matrix in the region around a point. Given the eigen values $\lambda_3 < \lambda_2 < \lambda_1$ of the covariance matrix, the geometric features are $\{\sigma_p = \lambda_1, \sigma_s = \lambda_2 - \lambda_1, \sigma_l = \lambda_3 - \lambda_2\}$ These geometric features represent point-ness, surface-ness and linear-ness of the region. In addition, the algorithm contains directional features using the local tangent and normal vectors. The tangent and normal vectors are estimated using the eigen vectors of the largest and smallest eigen values. The sine and cosine of these vectors $\{v_t, v_n\}$ with respect to the horizontal plane are used, giving a total of 4 directional features. To estimate the confidence in these features, the features are scaled according to the strengths of their corresponding eigen values: $scale\{v_t, v_n\} = \frac{\{\sigma_l, \sigma_s\}}{\max(\sigma_l, \sigma_p, \sigma_s)}$. The complete feature vector concatenates the 3 geometric features and 4 directional features for a resulting 7D feature vector. The feature vector, along with the hand labeled class ID for all points will be fed into the supervised learning algorithms to create the classifier.

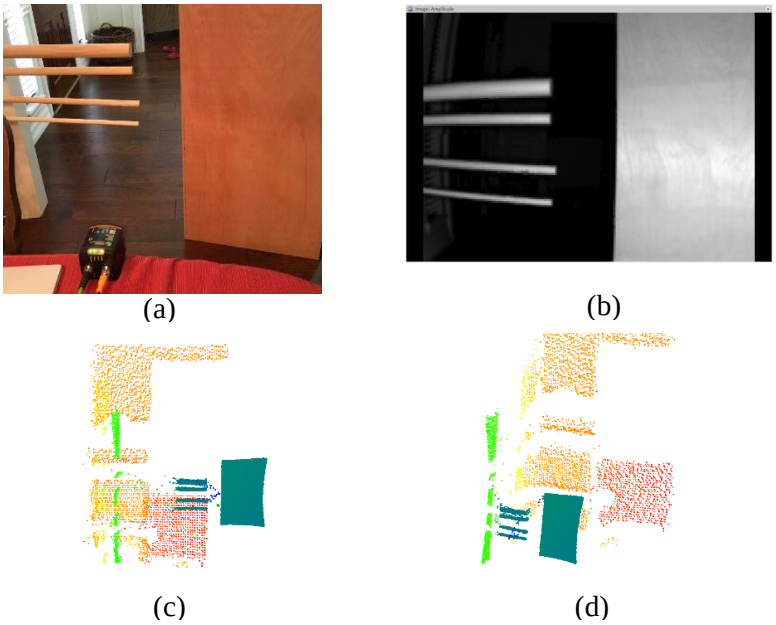

Figure 3: Flash Lidar Output. (a) Scene Setup (b) Flash LiDAR Intensity image, (c), (d) point cloud as viewed from different angles. Points colored by distance from the sensor

Solution Statement

The purpose of this project is to create a supervised learning classifier to identify and locate an object within a 3D point cloud. Because this project uses supervised learning, a hand labeled data set must be created. Each point in the hand labeled point cloud shall be classified as either belonging to the target object or not.

For this project, I propose to evaluate an array of supervised learning algorithms. I have chosen the following supervised learning algorithms: Support Vector Classifier, Ada Boost, K Nearest Neighbors and Gaussian Naïve Bayes.

To improve the efficiency or predictive ability of the learning algorithm, it is sometimes possible to transform the data. Reducing the dimensionality of the feature vector can improve efficiency and other

transforms, to give the data a more gaussian distribution, can improve predictive performance. One can reduce the dimensionality of the feature vector using PCA. PCA can help mitigate the curse of dimensionality. All of the training data is squeezed into a denser, lower dimension space. However, reducing the dimensionality can also reduce the discriminating power of the classifier, so its positive or negative impact must be measured. Some classifiers assume that he data is normally distributed with zero mean and standard deviation of 1. This can give all features equal weight. For instance, if the mean value of 1 feature is 10x larger than another, its importance may seem to be 10x as great. This could force the classifier to consider that feature to have more importance than another even when it does not. Therefore, some classifier performance is improved if the data is transformed to a normal distribution e.g. Again, the positive or negative impact of such a transform needs to be measured.

The overall process used is the following:

1. Create point clouds of a prototypical scene
2. Hand label a training set
3. Define a set of features to train on. The features will describe the point cloud geometry local to each point in the cloud
4. Investigate the data to ensure the defined features have the discriminating power necessary to separate the target object from others
5. Using cross validation, split the data in to a test and train set
6. Evaluate a variety of supervised learning algorithms
7. Investigate the effects of feature vector transforms
8. Select the best performing algorithm, ranking them according to their F1 score
9. Tune the selected algorithm using grid search to optimize its parameters
10. Test the optimized algorithm on other point clouds to see if it can detect the object in other untrained data

Benchmark

To compare the performance of the classifier, I will benchmark it with a simple logistic regression model. Logistic regression tries to fit a linear model to the classified data. With logistic regression, the probability of the classification outcome is modeled using a sigmoid function of the input features. It is a simple model with fast evaluation time.

Evaluation Metrics

To measure the performance of the classifier, the F1 score is used. The classifier we are going to create is binary. A set of features either represents the target object or it does not. The F1 score measures the accuracy of a test with binary results. With a test, there are four possible outcomes: true positive, false positive, true negative and false negative. Given the rate of true positives, true negatives, etc. the precision, recall and F1 score of a test are defined as:

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

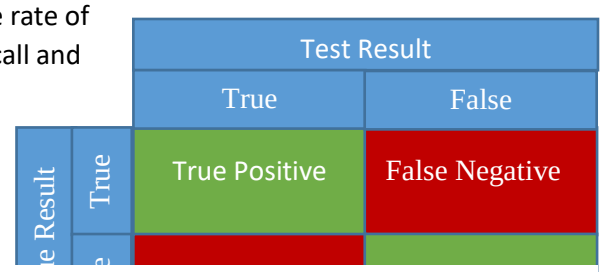

Figure 4: Possible test results are True positive, false positive, false negative and true negative

$$F_1 = 2\frac{precision * recall}{precision + recall}$$

The F1 score has a maximum value of 1 if the test results match the true results and a minimum value of 0 if the test is always wrong.

The training data will be split into a training and testing set. The training set will be used to train the classifier, and the testing set will be used to evaluate its performance. The F1 score on the testing data will be used to evaluate the classifier's performance.

Project Design

The workflow for this project will look something like the following:

1) Collect a sample point cloud data set. The scene should include the target object as well as other objects.
2) Hand segment a test + training data set. This hand segmentation should identify the points within the cloud belonging to the target object
3) Create the software that calculates a feature vector for all points within a point cloud. A point's feature vector will describe the geometry local to that point. The feature vector will use eigen values and vectors of the covariance matrix of points within a neighborhood.
4) Using the hand labeled data, calculate the feature vector for all points. Visualize the data in such a way that it is clear the feature vector can discriminate between points belonging to the target object and those that don't.
5) Investigate the effects of PCA, on the classifier accuracy. PCA can be used to reduce the dimensionality of the input data. This could potentially increase the efficiency of the classifier. However, dimensionality reduction can also have a detrimental effect on classifier performance by eliminating a discriminating dimension.
6) Investigate the effects of log, polynomial and other transformations of the input data. Log and polynomial transformations of the input data can potentially improve the performance of classifiers by making the data appear more Gaussian, implementing a kernel trick to make data linearly separable, etc. Investigate if these transforms increase the classifier's performance.
7) Using cross validation, separate the hand labeled data into a training and testing set. It is important to avoid testing and training on the same set of data to avoid over fitting. However, dividing the data into separate training and testing sets reduces the amount of data that can be used for both. Cross validation is a way to effectively increase the amount of data used by testing and training without needing to collect extra data.
8) Using the F1 score, evaluate a variety of supervised learning algorithms. Not all supervised learning algorithms are created equally. Some perform better on certain problems than others. To find the best candidate, an array of classifiers will be tried. Some potential algorithms to try are: Support Vector Classifier, Ada Boost, K Nearest Neighbors and Gaussian Naïve Bayes.
9) Once the best classifier is chosen, tune its parameters to optimize its performance. Each of the candidate classifiers will have an array of parameters that can be tuned. Using the grid search technique, the optimal set of parameters will be found.

10) Once the classifier has been optimized and an acceptable level of performance has been achieved, the classifier will be tried on other data sets that have not been hand labeled. This will verify that the classifier has reasonable performance on different data sets that it has not been trained on.