Using Faster-RCNN to Improve Shape Detection in LIDAR

TJ Melanson
Stanford University
Stanford, CA 94305
melanson@stanford.edu

Abstract

In this paper, I propose a method for extracting objects from unordered point cloud data by using 2D semantic segmentation on a camera image of the corresponding data. This added constraint, which exploits a common feature in datasets such as KITTI, would collapse the relatively unsolved problem of categorizing unorganized 3D data into a 2D problem plus also solved issues of sensor fusion and projection. The KITTI dataset was used as input to a Faster RCNN and then reprojected into 3D data to determine the regions of foreground objects, namely vehicles and pedestrians. Overall, there seemed to be an plausible application for using the projection matrices to determine the foreground points in the point cloud.

1. Introduction

In traditional structure from motion mapping, a camera uses a series of images to calculate disparity and depth to directly compute a point cloud for scene reconstruction. However, in cases such as autonomous vehicles, the 3D reconstruction alone is not enough to gather the necessary scene information. In the case of autonomous vehicles, vision is needed to distinguish obstacles as well as recognize intent. My project is about using object recognition to augment reconstruction by replacing partially reconstructed meshes of objects with full models of a generic model type recognized by CNNs. The results are evaluated visually, by comparison with the original scene, as well as numerically, by comparing depth, size, and object type to real collected data.

1.1. Background

Recent developments in computer vision, particularly in object recognition and sensor fusion, have accelerated the autonomous vehicle industry. However, although there are guidelines for so-called "Level 3" autonomy (vehicle assisting the driver), little has been done in visualizing the data the vehicle collects and displaying it to a user [1]. However, this project attempts to combine vehicle LIDAR / stereo reconstruction data with image recognition and segmentation via Faster-RCNN to provide a clear virtual reconstruction of the surrounding environment.

2. Related Work

2.1. Scene Segmentation before Neural Networks

In a paper in 2012. Kim *et al.* researched ways to segment objects in a scene by fitting the scene to sets of convex geometry [2]. It would first learn to recognize an object by a series of feature extractions. First, the points are fitted to sets of simple geometry, i.e. boxes and cylinders. Then, the sets of geometry are categorized together by the amount of overlap between them. Finally, the sets of primitives are ordered by the arrangement of objects within the hierarchy.

For the runtime part, the dataset was first parsed into primitive sets as part of preprocessing. Then, the primitive sets were turned into objects through a fast (~0.2 s) recognition phase.

Although this is relatively slow compared to some 3D object recognition algoirthms today, it was a major step in scene recognition, and laid out a potential framework for a neural implementation. For example, given that the dataset mainly consisted of fitting geometry, followed by linear recognition, it made the idea of a more affine network like PointNet possible. Also, while it was not a neural network, it was still a form of machine learning, as there was a learning and recognition phase.

2.2. FusionNet: Combining 2D and 3D data to improve object recognition

For volumetric networks, the lecturer sought to work with ShapeNet, a Princeton-based network of CAD models. In order to do well, the lecturer came up with three different types of networks. The first combined multiple 2-D views of a 3D image (20 sides, all equidistant from each other), sent them across 20 CNNs to

output the class. This is called a Multiple View CNN. The second was to do a series of 3D convolutions of the 3D voxel image rather than 2D convolutions (i.e. the kernel now has 3 dimensions). This was the iteration of the first Volume CNN (V-CNN). The second V-CNN improved upon the first by both using region sampling and taking advantage of the Inception Module style recently discovered by Google to try and piece together smaller components in the image. All of these elements, by themselves, showed significant improvement over the previous networks that entered ShapeNet40. The biggest leap, though, came from how the combination of these elements into a single network called FusionNet, which once topped the ShapeNet40 and provides the highest accuracy overall. The paper for the structure has been published outline on Matroid's website [3].

2.3. 3D Scene Segmentation

Most of the work for 3D scene segmentation has been for indoor scenes. The most recent major example was the 3DIS dataset [4], a series of pointcloud sets depicting large indoor scenes. These scenes were filled with indoor objects such as tables, chairs, and cabinets, and provided a content-rich set of point models within a scene.

To parse this dataset, PointNet was created to not only recognize objects as point clouds (it is currently on the top set of ModelNet classifiers), but also to distinguish objects in a point cloud scene. The idea behind PointNet was to create a network that can learn on unordered datasets by selecting the "important" points via max pooling. The results of each max pooling function are then connected through a series of affine fully connected layers as a sort of global feature analysis. It is additionally on of the few neural networks out there that use unordered points rather than voxel data [5].

3. Methods

3.1. 2D Image Segmentation

The purpose of using a neural network is to detect the shapes of the vehicle within the 3-D image so it can be processed or segmented out. Because of this, it made sense to use a Faster-RCNN structure to determine the regions of various foreground objects in an image. The input to the network is the camera data, which could be considered the projection of the 3D scene into a 2D image, and the output is the sets of bounding boxes for the relevant foreground objects in the region [8].

The metric for object recognition in the KITTI dataset was taken from the PASCAL Mean Average Precision (MAP) algorithm. This involves taking the number of correctly guessed images for each set, then dividing it by the number of correct images and false negatives in a given set. In the case of the KITTI dataset, the metric measures the number of bounding correct bounding boxes rather than the full image. As said before, a bounding box is considered correct for a vehicle if it has at least 70% intersection with the annotated bounding box, and a person is considered correct if the area of intersection is at least 50% of the full image.

3.2. 3D Segmentation from 2D Segmentation

Once these foreground obstacles are detected, they then could be tracked by the vehicle over time, and can then be either recorded or acted upon by a vehicle decision. The output of the Faster-RCNN is then combined with the 3D scene for the output. Originally, I was going to combine the 2D scene segmentation output with that of a 3D scene segmentation output. However, due to time constraints, and the relative recentness of the PointNet framework (the code for semantic segmentation was uploaded last week), the 3D point cloud was directly converted from the world view into the camera view using camera projection:

$$[wu, wv, w]^T = K\ T\ [x, y, z, 1]^T$$

K is the camera intrinsic matrix; T is the transformation from the Velodyne coordinate frame to the camera view frame (similar to the view matrix). The coordinates x, y, and z are the point cloud coordinates, and the resulting [wu,wv,w] vector are the 2D pixel coordinates u, v scaled by the homogenous coordinate w.

3.3. Data Conversion

For each of the experiments, a major component of the work was formatting the dataset to resemble the input for each of the interim steps. The formats in detail of each of the data structures is listed below. In particular, the KITTI dataset was translated to the VOC and ModelNet dataset formats, and the output of the Faster-RCNN was converted to the KITTI annotated format for benchmarking tests.

3.4. 3D Segmentation from 2D Segmentation

Originally, the final step was to gather some of the geometric properties of the shape could either be averaging the points to find the center, then using standard deviations to approximate the outer bounds of the shape. As for the certainty, using such an approximation would affect the original score, so the standard deviation would

likely have to be taken into account in addition to the original uncertainty score of the 2D region.

4. Dataset and Features

4.1. Baseline Data: KITTI

Originally, for the purpose of integrating this project with a previous project, the Oxford dataset was used. However, because of a lack of benchmarking, as well as the misaligned captures of the Velodyne LIDAR data and the camera image, the KITTI dataset was used in this project. (Note: existing code from the other project was not used in this project, and neither funds nor human resources were used in the making of this project, although the CPU we use was briefly used for creating the datasets). Ideally, the Faster-RCNN could be partially trained on the KITTI dataset; the KITTI dataset, as it contains only a few thousand images and 10 classes, would not be able to train data as well as the ImageNet data.

The purpose of this project is to detect objects in 3D space by properly segmenting the portions of the Point Cloud related to three classes: Person, Car, and Cyclist. Although other classes, such as Plant and Train/Tram, are included in the detection suite, there is not a significant enough set of objects outside of the three major class types to warrant a full training/testing dataset. Additionally, when an object is detected, but the data is either inconclusive or irrelevant to the class objects above, it is marked as a "Dontcare" object. Although the project could be generalized by including only a foreground and background object, it would be better to keep disparate items such as vehicles and pedestrians separate.

For training, however, in order to preserve the model already implemented in the system, the network used was the default network trained on ImageNet. The class output of the data, which included a class for vehicles, pedestrians, buses, trains, and bikes, could then be trimmed and converted to the class mapping of the KITTI dataset. Trimming was done by labeling the extraneous classes as "Dontcare."

4.2. 3D Benchmarking Suites

The KITTI site has both a 2D and 3D competition for both object detection and tracking. The competitions are ranked separately depending on if it detects people, cars, or cyclists. For the 3D object detection and tracking, the data must know the object position as well as the orientation in terms of the Y rotation (i.e. the direction a vehicle would be moving) [7]. For the object detection, it is enough to determine a bounding box that has at least 70% intersection with the correct annotation. The bounding box has both a location and orientation. The tracking dataset additionally has an object ID to distinguish between objects.

Instance level 3D object recognition has been done in a very similar fashion to 2D images, through a method known as a Volumetric CNN. This takes a 3D "image" of an object by viewing it as a series of blocks, or "voxels" [3]. The most significant work in the field was a project at Princeton called ModelNet, which created a database of models with labels as well as benchmarks for object recognition [6]. These models were of single objects, without any background clutter, simplifying 3-D recognition algorithms for recognizing the object.

4.3. Conversion of 2D Features into 3D Point Sets

The second step was to convert the 2D image data into the 3D LIDAR space. For the KITTI object detection dataset, the LIDAR data and the camera data were taken at the same time, so there was no issue of determining the change in vehicle pose between two image captures. Additionally, the Velodyne and camera were aligned such that the view frame for both datasets were the same. However, for the transformation into the camera view, the dataset were calibrated with internal camera parameters. These were used to project the Velodyne data into the 2D camera space. From there, the points within the bounding square of an object were determined and marked.

4.4. Dataset Formats

4.4.1 KITTI Object File:

Layer	**Values**
1. Data segment	Training, test
2. Device/recording type	Velodyne, Image_0[0-2], calib, label_0[0-2]
3. Capture number	0000- (~7900)
4. Data within file	Varies
(Label type)	Class, score, alpha, (bounding-box)

4.4.2 KITTI Raw:

Layer	Values
1. Date of recording	(mm-dd-yyyy)_sync
2. Device/recording type	Velodyne, Image_0[0-2], label_0[0-2]
3. Capture number	%dddddd
4. Parameters/data	Data directory, calib.txt, vo.txt
4. Data within file	Varies

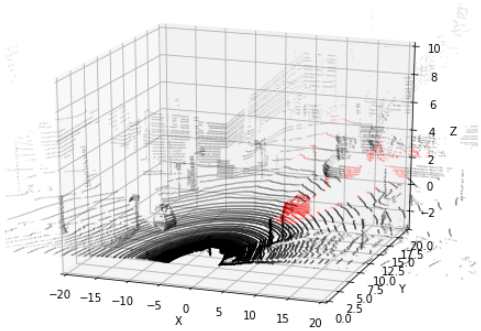
4.4.3 PASCAL_VOC

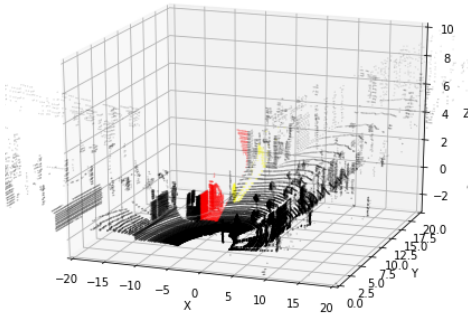
Layer	Values
1. Benchmark Type	Annotation, Layout, JPEGImages, SegmentationClass, SegmentationObject
2. Device/recording type	Velodyne, Image_0[0-2], calib, label_0[0-2]
3. Capture number	0000- (~7900)
4. Data within file	Varies

4.4.4 ModelNet

Layer	Values
1. Area	Area_%dd
2. Device/recording type	Velodyne, Image_0[0-2], calib, label_0[0-2]
3. Capture number	0000- (~7900)
4. Data within file	Varies

5. Experiments/Results/Discussion





The detected objects in the foreground of the image, and their corresponding projection onto the LIDAR view. The red corresponds to vehicle data, while the yellow corresponds to the person data.

The Faster-RCNN was able to detect both vehicles and pedestrians within the LIDAR cloud (see figure above). Although there is reprojection error, and some extraneous point cloud data due to including any point that fell within the bounding box. Even with these sources of noise,

however, the general location of the vehicle could be determined by the location of the highlighted data.

One major takeaway is that, although it may seem easier to reformat the data to build off of others' work, building your own model based on the structure in others' work is easier than trying to retrofit new data into the current codebase. This is because, while much of the dirty work in neural network structure is abstracted away by libraries such as TensorFlow and Pytorch, the input datasets for any network often have unique classes in addition to format. This means that a lot of the lower level work most code bases use to abstract away the work only applies to specific datasets, and are not flexible with their input. For example, both PointNet and Py-Faster-RCNN relied on a specific data structure for their respective default inputs (for PointNet, it was the ModelNet input, and for Py-Faster-RCNN, it was the Pascal VOC dataset). The sample evaluation code also assumed a certain set of classes when training the data, which either overfit the KITTI dataset (in the case of PASCAL) or didn't even apply (in the case of ModelNet). When all these datasets are removed, the result is essentially the base implementation using a CV library, which is high-level enough to implement the data structures in only a few hundred lines of code.

6. Conclusion/Future Work

It is possible to extract the foreground features from the overall mesh, although some work must be done to refine the extracted LIDAR points. Because the KITTI benchmarking did not output any data, and the Pointnet network integration could not be reformatted to train on the LIDAR dataset, there was not enough benchmarking data to conclude if this method improves current 3D tracking or segmentation methods for LIDAR data.

One possible improvement is to refine the bounds on the 2-D vehicle data, thereby removing possible noise from the outliers. However, this method could also create false negatives around the outer boundaries of the objects, and would not account for the noise created by projection error.

Another improvement would be to use 3D CNN to convert the noisy model into a more realistic model. There are several volumetric approaches to 3D recognition. If the original model was taken in as noisy input, gradient ascent could magnify the features the model is missing (for example, a car that looks incomplete due to missing parts could be modified to have a shape more like a vehicle's).

Perhaps the simplest approach is to extract more basic features from the extracted 3D model, and then abstract away the physical shape into a generic placeholder. For example, all the cars in the image would be converted to the same sedan shape, with different sizes and rotations depending on their location and orientation. In the end, this would be useful for tracking a vehicle based on its 3D bounding box. However, without some improvement to either denoise the model or by using learning to predict the center rather than a more naïve averaging method, feature extraction would be very prone to noise (as a few outliers could greatly affect the car shape and center point).

References

[1] Reese, Hope. "Autonomous driving levels 0 to 5: Understanding the differences." *TechRepublic.* 20 Jan 2016.

[2] Young, Kim *et al.* Acquisition of 3D Indoor Environments with Variability and Repetition. Stanford University, 2012. http://hci.stanford.edu/cstr/reports/2012-01.pdf

[3] Hegde Vishakh , Zadeh Reza. FusionNet: 3D Object Classification Using Multiple Data Representations. Matroid, Jul-Nov 2016

[4] Armeni *et al.*, 3D Semantic Parsing of Large-Scale Indoor Spaces, CVPR2016.

[5] Qi, Charles R and Su, Hao and Mo, Kaichun and Guibas, Leonidas J. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. 2016

[6] Princeton ModelNet. http://modelnet.cs.princeton.edu

[7] Geiger, Lenz, and Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. 2012

[1] Shaoqing Ren and Kaiming He and Ross Girshick and Jian Sun, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

[2] Kundu et al. Joint Semantic Segmentation and 3D Reconstruction from Monocular Video. Georgia Institute of Technology, 2014. https://web.engr.oregonstate.edu/~lif/HybridSFM-ECCV2014.pdf

[3] http://www.robots.ox.ac.uk/~mobile/Papers/2016IROS_maddern.pdf

[4] http://graphics.usc.edu/cgit/publications/papers/point_cloud_3dcnn.pdf

[5] http://ai.stanford.edu/~haosu/papers/iccv_renderforcnn.pdf

[6] https://arxiv.org/pdf/1702.04405.pdf