Object Localization, Segmentation, and Classification in 3D Images

Allan Zelener

The Graduate Center, City University of New York
Object Localization, Segmentation, and Classification in 3D Images

by

Allan Zelener

A dissertation submitted to the Graduate Faculty in Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York.

2018
This manuscript has been read and accepted for the Graduate Faculty in Computer Science in satisfaction of the dissertation requirements for the degree of Doctor of Philosophy.

(required signature)

Date

Chair of Examining Committee

(required signature)

Date

Executive Officer

Ioannis Stamos

Yingli Tian

Andrew Rosenberg

Philippos Mordohai

Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK
Abstract

Object Localization, Segmentation, and Classification in 3D Images

by

Allan Zelener

Advisor: Ioannis Stamos

We address the problem of identifying objects of interest in 3D images as a set of related tasks involving localization of objects within a scene, segmentation of observed object instances from other scene elements, classifying detected objects into semantic categories, and estimating the 3D pose of detected objects within the scene. The increasing availability of 3D sensors motivates us to leverage large amounts of 3D data to train machine learning models to address these tasks in 3D images. Leveraging recent advances in deep learning has allowed us to develop models capable of addressing these tasks and optimizing these tasks jointly to reduce potential errors propagated when solving these tasks independently.
I am extremely grateful for the support from friends, family, and colleagues throughout the many years necessary to gain the experience to write this dissertation. The continued lifetime of support from my parents, Sofia Davydova and Vladimir Ze- lener, have made this work possible as well as the academic mentorship in my childhood from my brother Yan. I am also grateful for the encouragement of my sister-in-law Adina and the opportunity to see my young nephew Julian grow over the years to come.

My thanks to my advisor Ioannis and committee members Yingli, Philippos, and Andrew for their valuable feedback. To the faculty including Subash, Susan, Olympia, Felisia, Robert, Gabor, Liang, and many others with whom I’ve interacted over the years. To John Kender who helped review my literature sur- vey. To my labmates over the years Juan, Thomas, Agis, Sam, and Xiaoke. To James, Ilona, and the other students who aided my research. To my mentors at Google and Twitter, Martin, Qian, Steve, and Clément. To my friends Elijah,
Miky, Fan Yi, Alex, Hernissa, Ali Syed, Ali Ahmed, Xing, Alexey, Raj, Alex, Valia, Michael, Xi, Morgan, Kenny, JP, Andrew, Kati, and many more I’ve surely forgot to mention. Finally to my new colleagues at Zoox including Drago, Subhasis, Balazs, Dominic, David, Nitesh, Sabeek, Ashesh, Carden, and Yushi who have introduced me to many great new challenges to work on and inspire me to succeed going forward.
Contents

1 Introduction .................................................. 1
  1.1 Motivation .................................................. 1
  1.2 Object Identification Tasks ............................... 2
  1.3 Overview ................................................... 5

2 Related Work ................................................ 8
  2.1 3D Point Clustering ........................................ 9
  2.2 3D Deep Learning .......................................... 15

3 Part-based Object Classification ......................... 21
  3.1 Local Feature Extraction .................................. 22
  3.2 Part Segmentation ......................................... 24
  3.3 Part-Level Feature Extraction ............................ 25
  3.4 Structured Part Modeling .................................. 27
  3.5 Experimental Evaluation ................................... 30
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.6</td>
<td>Discussion</td>
<td>33</td>
</tr>
<tr>
<td>4</td>
<td>CNN-based Object Segmentation</td>
<td>36</td>
</tr>
<tr>
<td>4.1</td>
<td>Labeling Procedure</td>
<td>38</td>
</tr>
<tr>
<td>4.2</td>
<td>Patch Sampling</td>
<td>39</td>
</tr>
<tr>
<td>4.3</td>
<td>Input Features</td>
<td>41</td>
</tr>
<tr>
<td>4.4</td>
<td>CNN Model</td>
<td>43</td>
</tr>
<tr>
<td>4.5</td>
<td>Experimental Evaluation</td>
<td>45</td>
</tr>
<tr>
<td>4.6</td>
<td>Discussion</td>
<td>51</td>
</tr>
<tr>
<td>5</td>
<td>Depth-conditioned Object Localization</td>
<td>53</td>
</tr>
<tr>
<td>5.1</td>
<td>Lidar Preprocessing</td>
<td>55</td>
</tr>
<tr>
<td>5.2</td>
<td>Depth-conditioned Anchors</td>
<td>57</td>
</tr>
<tr>
<td>5.3</td>
<td>CNN Model</td>
<td>62</td>
</tr>
<tr>
<td>5.4</td>
<td>Experimental Evaluation</td>
<td>65</td>
</tr>
<tr>
<td>5.5</td>
<td>Discussion</td>
<td>68</td>
</tr>
<tr>
<td>6</td>
<td>Conclusion</td>
<td>71</td>
</tr>
<tr>
<td>6.1</td>
<td>Discussion</td>
<td>71</td>
</tr>
<tr>
<td>6.2</td>
<td>Open Problems</td>
<td>73</td>
</tr>
</tbody>
</table>

Bibliography 76
List of Tables

2.1 Summary of related work on 3D object identification. . . . . . . . 9

3.1 Object part classification accuracy. . . . . . . . . . . . . . . . 32

3.2 Object classification accuracy for Sedan vs SUV . . . . . . . 32

4.1 Average precision for different input feature combinations. . . . 46

4.2 Average precision on non-missing labeled points. . . . . . . . 48

5.1 Localization performance with rescore confidence loss. . . . . . 66

5.2 Localization performance without rescore confidence loss. . . . 68
# List of Figures

1.1 Sample ground truth from ObjectNet3D ............................. 2  
1.2 Multi-task cascade network. ........................................... 4  

2.1 Computation of the spin image 3D local feature. ............. 10  
2.2 3D point clustering on a K-NN graph using graph-cuts. ........ 12  
2.3 Hierarchical semantic segmentation of an RGB-D image. ...... 14  
2.4 Overview of a system using CNN depth image features. ....... 16  
2.5 A 3D spatial convolutional neural network for object classification. 18  

3.1 Planar segmentation of a sedan. .................................. 24  
3.2 Generalized HMM for part-based object classification. ....... 27  

4.1 Overview of lidar CNN system. ................................. 37  
4.2 Example lidar scene containing missing points on cars. ....... 39  
4.3 Detail of labeling procedure for missing points. ............... 40  
4.4 Definition of signed angle feature. ............................... 42
LIST OF FIGURES

4.5 Input initial features for lidar CNN. .......................... 44
4.6 Precision-recall curves for input feature comparison. ............. 47
4.7 Precision-recall curves for comparing efficacy of missing point labels. .................................................. 49
4.8 Comparison of models trained with and without missing labels. . 50
4.9 Sample results of lidar CNN segmentation on busy NYC street. . 51
5.1 Ground truth crops for localization from Street View training set. 56
5.2 Clustering of box anchor parameters from training set. .......... 60
5.3 Visualization of anchor boxes with mean depths. .............. 61
5.4 Localization results on Street View test set. .................. 67
5.5 Object AP and depth errors over training epochs for localization. 69
Chapter 1

Introduction

1.1 Motivation

Our world is a three-dimensional environment and in order for our automated systems to effectively interact with this environment they need to model and reason about the objects of interest that inhabit the world which are necessary to solve a given task. For example these could be vehicles and pedestrians that a self-driving car must avoid colliding with or products stored in a warehouse that a robot must collect for shipping. These systems would employ visual sensors, such as RGB and lidar cameras, that typically acquire 2D image data of the 3D world. It is from these images that we must recover the inherent 3D properties of objects in the world to enable higher-level tasks.
CHAPTER 1. INTRODUCTION

1.1 Object Identification Tasks

Identifying objects of interest in images involves solving a set of related tasks. Given an image of a scene it is first necessary to find the general location of each object within the image, for example by estimating a bounding box for each possible object. We define this task as object localization and it is also often referred to as object detection in the literature. Next this localization may be
CHAPTER 1. INTRODUCTION

refined by segmenting the image pixels corresponding to the localized objects from other parts of the scene. Finally given an accurate segmentation mask of each object it is possible to predict higher level properties such as its semantic class or 3D pose. Figure 1.1 contains a visualization of the ground truth annotations for these tasks on a 2D image. While these tasks are listed here as a sequence of steps, it can be beneficial to share information between these tasks. For example the image features used to localize vehicles are likely different from those used for street signs, which means that localization may be conditionally dependent on semantic class. Furthermore, errors earlier in the process may be propagated to later tasks. It is not possible to correctly classify an object if it was never detected as an object of interest within the scene.

Accurately estimating an object’s 3D shape and pose from a single 2D image using a traditional camera is a difficult task, in fact if no simplifying assumptions about visual cues are used then it is an underdetermined problem with infinitely many solutions due to scale ambiguity in perspective projection. Fortunately in recent years there has been a steady increase in the availability of 3D sensors capable of accurate pointwise depth measurements such as lidar scanners for outdoor and aerial sensing or RGB-D cameras for short-range indoor use, including consumer level sensors like the Microsoft Kinect or Google Tango. This 3D data introduces its own set of challenges. The density of 3D point measurements may
vary throughout a scene depending on the distance of scanned surfaces from the sensor. It is also possible to have missing data due to incompatibility between a surface’s reflectance properties and the scanning technology, for example glass windows often refract a lidar scanner’s laser and glossy paint on cars can reflect it. There will also still be unobserved parts of any given object due to self-occlusion or other occluding scene elements so these 3D scans would only partially match full 3D reference models. However despite all these issues there are inherent advantages to using these 3D sensors and combining them with traditional cameras. The depth measurements directly connect the 2D projections of an environment perceived by a sensor with the environment’s 3D shape, constraining the problems found in color images such as scale ambiguity or camouflage-like textures.

Figure 1.2: Multi-task cascade network [Dai et al., 2016]. Object localization, segmentation, and classification are solved in sequence using jointly learned features in a deep neural network.
By leveraging the large amounts of 3D data that can be collected with 3D sensors, far more than could be easily annotated in color camera images, we are able to train machine learning models that solve the object identification tasks using low-level depth measurements. Earlier works in this area treated this data as a 3D point cloud and used point clustering to estimate local geometric properties of surfaces as features for machine learning models. More recently deep learning models such as convolutional neural networks have become state-of-the-art on a variety of 2D vision tasks including image classification [Krizhevsky et al., 2012, He et al., 2016] and segmentation [Long et al., 2015] and are increasingly being adapted for 3D image analysis. These deep artificial neural networks provide a general framework for optimization-based feature extraction on the target task that outperforms previous manually designed feature extractors. The modeling flexibility provided by deep learning also allows tasks to be solved jointly and the entire model trained end-to-end, for example [Dai et al., 2016] uses a multi-task cascade for object localization, segmentation, and classification as shown in Figure 1.2.

1.3 Overview

The following chapters describe our approaches to addressing the object identification tasks in the context of 3D images. In particular we highlight the shift from
Methods using unsupervised 3D point clustering for feature extraction to deep learning directly on 3D image data in both the literature and our own approaches to these tasks. In Chapter 2 we present a review of the literature on identifying objects in 3D images including the foundational 3D point clustering methods and more recent state-of-the-art deep learning based approaches.

Chapter 3 describes our first approach towards object classification in lidar scans using a 3D point clustering based on RANSAC plane fitting and structured prediction for modeling relations between clusters. Here we consider pre-segmented object point clouds decomposed as piecewise planar parts and perform part-based classification. We show in this work that adding more sophisticated relations between sensed surface regions, as opposed to aggregating all features globally, has the potential for increased classification performance. However in practice the performance is limited due to errors introduced in the unsupervised clustering step. In the discussion of this work we describe how this motivated us to pursue deep learning based methods for the following work described in this dissertation.

Chapter 4 details our initial work on object segmentation using a convolutional neural network approach. Whereas in the previous chapter we assumed a coarse segmentation of objects from the scene using some point clustering method, in this work we aim to solve this initial problem using a CNN on the lidar acquisition.
grid. We adapt the CNN approach to handle the missing point problem found in lidar and produce a high quality vehicle segmentation mask for urban scenes.

Chapter 5 describes an extension of our work on CNN-based lidar processing to object localization. This allows us to generate bounding box proposals within the lidar acquisition grid for each object of interest as well as separate object instances. Because we have direct 3D measurements, in addition to the 2D bounding box dimensions within the scanning grid we also estimate the 3D properties of the object instance within each bounding box such as mean distance from the lidar sensor.

Finally we conclude in Chapter 6 by placing our contributions in context of the stated object identification tasks for 3D images. We also consider potential future research directions in improving performance on each task and natural future extensions of our work to new problem statements and applications.
Chapter 2

Related Work

The object identification tasks that we consider in this work have been extensively studied in the computer vision literature. These tasks have been investigated both independent of each other and in combinations, in domains including both RGB camera images and 3D range sensor data, and using a wide array of techniques from pixel clustering to convolutional neural networks. In this chapter we will focus on works related to object detection in 3D image data and how trends in this research area have shifted from applying geometry based clustering and feature extraction to approaches based on the recent success of deep learning and in particular convolutional neural networks for solving the same tasks on 2D color images. Table 2.1 summarizes the approaches to 3D object identification discussed in this chapter. Our own work parallels the trajectory of this research trend and in the following chapters we will discuss how our contributions, which focus on the domain of urban lidar scans, are influenced by the related work.


<table>
<thead>
<tr>
<th>Methodology</th>
<th>Data</th>
<th>Domain</th>
<th>Primary Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local 3D feature matching.</td>
<td>Point cloud</td>
<td>Synthetic and lab</td>
<td>Instance detection &amp; retrieval.</td>
</tr>
<tr>
<td>Point-based clustering and structured models.</td>
<td>Point cloud</td>
<td>Urban and indoor</td>
<td>Semantic segmentation.</td>
</tr>
<tr>
<td>Single 2D spatial conv nets.</td>
<td>Depth Image</td>
<td>Urban and indoor</td>
<td>All tasks.</td>
</tr>
<tr>
<td>Multiple image sensor fusion neural networks.</td>
<td>Depth Image</td>
<td>Urban</td>
<td>All tasks &amp; 3D reconstruction.</td>
</tr>
<tr>
<td>3D spatial neural networks.</td>
<td>Voxel grid</td>
<td>Indoor</td>
<td>All tasks &amp; 3D reconstruction.</td>
</tr>
</tbody>
</table>

Table 2.1: Summary of related work on 3D object identification.

### 2.1 3D Point Clustering

Early work in 3D sensing has been based on data collected from lidar sensors that are either stationary or mounted on a moving car or airplane. These early sensors provide a coarser resolution compared to traditional camera images and sensor fusion between lidar sensors and RGB cameras remains a challenging task. However they do provide accurate 3D measurements from a single sensor and this prompted early research to focus on geometric features computed on 3D point clouds for detecting objects in these scans rather than color based features. The general approach paralleled similar work in 2D object detection with the basic
steps of each system involving feature extraction, clustering, and applying machine learning techniques to features of extracted clusters for segmentation and classification.

Figure 2.1: Computation of the spin image 3D local feature from [Patterson et al., 2008]. Every point within the volume of a cylinder centered on a keypoint is projected onto the keypoint’s surface normal and another vector orthogonal to the normal to form a 2D image feature. Conceptually this is like spinning an image plane around the surface normal and accumulating the number of intersections of each 3D point at each pixel.

These early works typically regard data acquired from 3D sensors as a 3D point cloud, which can be defined as a set \( \{ x \mid x \in \mathbb{R}^3 \} \). This is because a common application goal was mapping large scale environments which involves the registration of multiple point clouds acquired from different viewpoints into a global coordinate system. The design of feature extraction methods was influenced by this perspective and encouraged the design of features to be independent of the basis chosen from the global coordinate system with properties like trans-
CHAPTER 2. RELATED WORK

lation and rotation invariance. In order to achieve translation invariance, features are computed on a local supporting region such as a sphere around a sampled key-point. For rotation invariance, a new local basis can be created using the surface normal estimate for the local surface defined by the points in the selected region. Feature extraction methods such as spin images and 3D shape contexts [Johnson and Hebert, 1999, Frome et al., 2004] then proceed by accumulating statistics of the selected region, for example 3D shape contexts subdivides a spherical region into bins that represent the number of 3D points contained in each bin, effectively voxelizing the support region. An illustration of how the spin image is computed is shown in Figure 2.1. Other methods such as [Rusu et al., 2010] utilize higher order geometric properties like the surface normals of other points within the support region to compute features. These fixed size features can then be utilized with contemporary machine learning techniques such as support vector machines and random forests to classify the local region. However these features have primarily been designed for the task of exact object matching and involve selection of keypoints whose features are unique to a given object. For dense segmentation simply classifying local surface patches independently tends to ignore larger scale surface structure with many surface patches from different objects appearing identical in feature space, partially due to the invariances built into the feature design. This has prompted research into clustering of multiple surface patches and their
features and designing additional features that are better suited to densely classify the objects in a scene.

Figure 2.2: 3D point clustering on a K-NN graph using graph cuts from [Golovinskiy et al., 2009]. The segmentation algorithm takes as input a K-NN graph constructed from a point cloud, as shown on the left, as well as a hypothesis foreground point. Using graph-cuts the red edges in the K-NN graph are eliminated and the street sign is segmented from the rest of the scene as shown on the right.

There have been several approaches to integrating local features to solve the higher level object classification task. The work of [Patterson et al., 2008] utilizes features computed at both local keypoints and at the level of the candidate segment. A dense set of spin image features are used to generate candidate seg-
ments for vehicles in urban scenes and then the quality of each candidate is evaluated using the extended Gaussian image feature, essentially a histogram of the surface normals for each point in the candidate segment. In [Huber et al., 2004] a candidate object is segmented into coarse parts and an aggregation of the features computed for each part. Features for a part segment are formed by matching locally extracted features to a discretized feature codebook given by $k$-means clustering of all local features found in the training set. Instead of clusters, sequences of points along a given scanline are considered in [Stamos et al., 2012], segmentation into categories such as vertical, horizontal, and vegetation are then determined by changes in the local features along the sequence. Unsupervised graph-cut methods are used in [Golovinskiy et al., 2009] to generate large scale candidate clusters over which a number of features are computed, as shown in Figure 2.2. Many works at this time begin to utilize an increasing number of segment-level features such as average height, segment volume, or variation in locally estimated principal component directions.

A challenge observed in the previous works that decompose a scene into point clusters or segments is that errors introduced at one level of the segmentation can propagate and become difficult to correct later in the pipeline. The most recent methods in this direction have focused on establishing a hierarchy or more general graphical model of segments and their relations to each other rather than
CHAPTER 2. RELATED WORK

Figure 2.3: Hierarchical semantic segmentation of an RGB-D image from [Wu et al., 2014]. The fridge handle segment, lower in the hierarchy, is predicted only in an image where the system can confidently estimate it, otherwise it is considered a component of the higher level fridge door.

using a single predetermined segmentation strategy. Multiple rounds of classification are used by [Xiong et al., 2011] to generate contextual features based on the preliminary classifications of neighboring segments in space and the segmentation hierarchy. More sophisticated inference procedures based on probabilistic graphical models [Anguelov et al., 2005, Savinov et al., 2016] have also been used to better utilize local connectivity between points. The work of [Anand et al.,
2013, Wu et al., 2014] uses a hierarchical segmentation tree model in order to find an optimal cut in the hierarchy to produce high confidence segments, as shown in Figure 2.3. Rather than producing a hierarchy through unsupervised segmentation [Dohan et al., 2015] learns a scoring function to merge neighboring clusters to find a hierarchy that will lead to a good final segmentation.

More recently there has been an increased availability of affordable RGB-D cameras that also systems similar to those for pixel images to be applied to pixel depth images as well such as the CRF-based work of [Silberman and Fergus, 2011]. The increased availability of RGB-D has also helped the adoption of deep learning methods which have become state-of-the-art on many 2D computer vision tasks since [Krizhevsky et al., 2012] won the Imagenet image classification challenge in 2012 by a significant margin and prompted a huge increase in neural network research across the entire field. In the following section we discuss in more detail recent work at the intersection of 3D computer vision and deep learning.

### 2.2 3D Deep Learning

Initial work within the recent wave of deep learning in 3D images utilized RGB-D sensors and treated depth as simply an additional input modality for semantic segmentation with 2D convolutional neural networks [Couprie et al., 2013]. However
Figure 2.4: Overview of the system of [Gupta et al., 2014] using CNN depth image features. This work uses CNN-based feature extraction as one component of a multi-stage pipeline for instance and semantic segmentation.

depth alone does not entirely capture all the geometric properties of the image. For example a pair of adjacent pixels in a depth image may have the same value but may be further apart in space than another pair of identical pixels closer to the sensor. In this case determining the actual 3D positions of these points requires knowledge of the sensor’s spatial resolution. The work of [Gupta et al., 2014] addresses this by computing additional features during preprocessing which include height from an estimated ground plane and angle between estimated surface normals and the up direction to generate CNN features for object detection, although like many other works from this period the CNN is used primarily as a feature extractor rather than for end-to-end learning. An overview of this system is shown in Figure 2.4.

In terms of estimating 3D object properties from single 3D images one ear-
lier work on object pose estimation [Papon and Schoeler, 2015] utilized known surface normals themselves as additional input channels from synthetic RGB-D images which were likely used because large datasets with pose annotations were not yet available. More recently the work of [Li et al., 2016] estimates 3D bounding boxes from a single lidar image. A related line of work in 2D vision has used RGB-D images as ground truth for estimating depth and surface normals as well as semantic labels in RGB images [Eigen and Fergus, 2015, Mousavian et al., 2016], and has also been extended to use these estimates for predicting object pose and visual similarity between objects [Bansal et al., 2016]. Higher level object pose from 3D bounding boxes are predicted in [Mousavian et al., 2017] by imposing geometric constraints using 2D bounding boxes in image space. One unifying theme in all of these works is that low-level geometric properties like depth and surface normals are related to higher level tasks like object pose estimation and semantic segmentation and can be utilized either as pre-calculated inputs or auxiliary outputs to improve performance on these tasks.

Another branch of 3D deep learning for object recognition considers objects as existing in a 3D space rather than lying on a 3D image and generates feature representations based on this perspective. For example, given a 3D object model the work of [Shi et al., 2015] generates a 2D convolutional feature map by projecting points from the object onto an enclosing cylinder. This is related to a multi-view
Figure 2.5: A 3D spatial convolutional neural network for object classification from [Wu et al., 2015]. On the right are averaged activations for particular filters. Similar to 2D conv nets, low level filters at L1 activate on simple surfaces and corners, mid-level filters at L2-L3 on object parts, and higher level filters at L4-L5 on whole objects.

approach like that of [Su et al., 2015] which generates a representation by pooling 2D convolutional features from multiple viewpoints surrounding the object. Most recently this approach has been used by [Chen et al., 2017] to combine images from RGB, lidar, and a bird’s eye reprojected view of the lidar in order to estimate 3D object bounding boxes.

An alternative approach is to represent the objects using a 3D voxel grid, this is used by [Wu et al., 2015] as input to a 3D convolutional neural network for shape completion and object recognition as well as view planning for active recognition. A diagram of this 3D CNN is shown in 2.5. A similar 3D convolutional framework
is used by [Song and Xiao, 2016] for 3D region proposal and combined with 2D image features for object classification and 3D bounding box refinement. Both volumetric and multi-view approaches are examined by [Qi et al., 2016] where they note a surprising performance shortfall of 3D voxel methods. These methods are sensitive to the choice of grid orientation and are more constrained in terms of the spatial resolution that can be represented since memory requirements grow cubically rather than quadratically in the size of the representation. They propose several solutions such as multiple volumetric inputs with various orientations of the 3D input. They also utilize probing kernels which are $1 \times 1 \times N$ convolutional kernels, where $N$ is the full volume extent, that transform the input volume into an image representation which is then processed by 2D convolutions. More recent work by [Riegler et al., 2017] utilizes an octtree data structure to make the computation of 3D convolutions more efficient by omitting operations where there are only zero-activations. Additional work that also addresses the sensitivity to rotation would make this approach even more promising.

Finally, one very recent line of work attempts to utilize unstructured point clouds directly by utilizing neural network architectures that are not based on spatial convolutions but instead use alternate connection schemes. A graph-convolution approach is utilized by [Ravanbakhsh et al., 2017] on a K-NN graph constructed from the point cloud for classification. A variant on a fully connected network
is used by [Qi et al., 2017] where dense layers are applied to features for each point and then the features for all points are realigned using a spatial transformer layer [Jaderberg et al., 2015] for classification and semantic segmentation. These approaches utilize the 3D structure of the data, like the voxel based approach, while also maintaining robustness to rotation and other artifacts introduced by discretization.
Chapter 3

Part-based Object Classification

Initial work on object classification for localized object candidates in 3D scenes [Golovinskiy et al., 2009] has utilized aggregations of simple local features like spin images [Johnson and Hebert, 1999] to generate global feature descriptors for candidate objects. We observe however that this approach does not capture the fine-grained variations in shape which are needed to discriminate between similar semantic categories. For example different classes of vehicles like sedans and SUVs have similar global shapes and it is necessary to utilize specific local properties, such as curvature of the sides or the angle at which the car trunk is joined to other parts. Furthermore, in 3D range scans the object is often partially observed and so an aggregation of local features may be more indicative of the sensor’s relative viewpoint rather than the object category. To address these challenges we adopt a parts-based approach using planar clustering inspired by earlier work that used a simple three-part front/middle/back segmentation on syn-
thetic models [Huber et al., 2004]. By associating local features to object parts and computing additional features between adjacent parts we are able to build a structured global representation for the entire object that captures its observed 3D shape using a piecewise planar approximation [Zelener et al., 2014].

The model consists of a four stage pipeline composed of local feature extraction, RANSAC-based part segmentation, part-level feature extraction, and structured part modeling. We evaluate our model on a collection of vehicle point clouds that have been manually extracted from the Wright State Ottawa dataset which consists of unstructured point clouds that have been registered together from both ground and aerial lidar scans of Ottawa. We show that our structured prediction model achieves superior classification accuracy for object parts and can improve overall object classification.

3.1 Local Feature Extraction

We define local features as statistics computed with respect to a reference point using neighboring points within a fixed radius as support. For 3D feature descriptors these are typically histograms of neighboring point positions or surface normal orientations parameterized within the support space. For this work we selected the spin image [Johnson and Hebert, 1999] feature descriptor which utilizes an estimated surface normal at the reference point to parameterize the support space.
resulting in a rotationally invariant descriptor.

In order to ensure only those reference points with well-populated supports are used we use a statistical outlier filter to remove points whose nearest neighbors have an average distance beyond one standard deviation of the mean average distance for all points within a given object. For the remaining points we estimate surface normals using PCA and orient them away from the centroid of the object’s footprint on the ground. Spin images are computed on a dense subsampling of these points using a fine-grained voxel grid. In order to adjust for variable density in our scans we weight the contribution of each point to a spin image by its inverse density, which is the inverse of the number of neighbors within a fixed radius.

We use a large support radius for computing spin images so that the local features can capture global object shape and the relative position of the reference point. This parameterization makes the features more amenable to the task of object classification and for use in a visual bag-of-words descriptor rather than finding locally unique points when doing keypoint detection for exact matching. This descriptor will be used as our baseline global object descriptor and as a component of the part-level object descriptor.
Figure 3.1: Planar segmentation of a sedan. Dark blue points correspond to unsegmented and unlabeled points, typically interior points. Here the manual ground truth labels for each segment in the order the segments were automatically extracted are light blue roof, cyan lateral-side, lime green front-bumper, yellow trunk, and red hood. Our method is robust to some interior points being included in these segments.

3.2 Part Segmentation

For part segmentation we assume that our objects of interest have roughly piece-wise planar exteriors which is a reasonable assumption for man-made objects at the level of detail found in range scans. Our segmentation method is unsupervised and can be done in parallel to local feature extraction. The planar segments will then be combined with the coinciding local features to form part-level features which are expected to vary significantly between different parts.
Planar segments are extracted iteratively using an adaptive RANSAC approach as described in [Hartley and Zisserman, 2004], essentially accepting a random candidate plane with the most inlier points after an adaptive number of random trials. A typical approach to generating candidate planar models is to randomly sample three points that are not colinear. However due to occlusions and transparent surfaces that expose an object’s interior, such as windows on a car, it is possible to fit planes that intersect through the object interior and do not correspond to semantically identifiable surface components. We avoid these undesirable candidate planes by estimating the convex hull of the object point cloud using the QHull algorithm [Barber et al., 1996] and sampling candidate planes from the faces of the convex hull. Due to noise in the sensor measurements, outliers can bias the planes given by the convex hull so we robustly reestimate each selected plane through expectation-maximization using PCA. We assume the observed surface of our object can be explained with a small number of large planar components and so limit the total number of planar segments to five or stop when at least 90% of points are segmented. An example of the resulting segmentation can be seen in Figure 3.1.

### 3.3 Part-Level Feature Extraction

The densely sampled local descriptors are combined with their corresponding part segments to produce a visual bag-of-words representation. We apply the $k$-means
algorithm to all spin images in the training set to generate a codebook of features for a visual bag-of-words descriptor, where any given test spin image corresponds to the closest mean spin image in the codebook. The descriptor for each part is a $L_2$-normalized count vector of the number of local descriptors matching each element of the codebook. Since the codebook was generated from the training set the matches for each local feature are given by the result of the $k$-means clustering. To efficiently match test examples we construct a $kd$-tree to perform efficient search through the codebook. For our experiments we chose a codebook of size 50 since larger codebook sizes did not significantly change classification performance in preliminary testing.

Additional part-level features that give a more global description of each part’s shape and its place in the scene are also computed and concatenated to the visual bag-of-words descriptor. This includes the average height of all the points in the part assuming the up direction and height of the origin in the registered coordinate system is reliable across scenes. We also include a binary indicator variable for whether the part has a mostly horizontal or vertical alignment. We test the angle between the planar part’s estimated surface normal and the axis corresponding to the up direction and if it is less than 45 degrees then we assume the part is vertical, otherwise it is horizontal. Finally we include the mean, median, and max of the plane fit errors for the points in each part, the three eigenvalues from the plane
CHAPTER 3. PART-BASED OBJECT CLASSIFICATION

Figure 3.2: Generalized HMM for jointly classifying a sequence of object parts and object class. Part labels depend only upon part features and joint features with the previously predicted part. Class labels depend on the classification of all parts and their features.

estimation ($\lambda_1, \lambda_2, \lambda_3$, in descending order), and the differences between adjacent eigenvalues which are referred to as linearity ($\lambda_1 - \lambda_2$) and planarity ($\lambda_2 - \lambda_3$) which have been used in previous work [Anand et al., 2013, Kahler and Reid, 2013]. These measures are based on geometric interpretations of the PCA-based planar estimation.

3.4 Structured Part Modeling

Traditional structured prediction models typically exploit the natural structure of a target domain to simplify their graphical models and avoid the hardness of inference on general Markov random fields. For example the linear structure of natural language sentences or the grid structure of camera images. In an un-
structured point cloud registered from multiple scans there is no simple natural structure to exploit, so we instead impose a linear structure over our small number of high level parts. We adopt a generalized sequential Hidden Markov Model which can be trained online and discriminatively by an averaged structured perceptron [Collins, 2002]. Each observed variable in the HMM $x_i$ corresponds to a part-level feature and the hidden variables correspond to part class labels $a_i$. The HMM is generalized to include a final hidden variable $c$ corresponding to the overall object class that depends on all previous observations. A graph depicting this model can be seen in Figure 3.2.

Our linear approximation to a more general MRF requires a sequential ordering of the object parts. While the iterative RANSAC procedure used to generate the parts gives such an ordering that we found to be superior to random permutations, it is too heavily influenced by variations in occlusions and variable point density determined by the scanner location. Again we utilize the known geometric properties of the scene and order the parts such that horizontal parts appear before vertical parts and within descending order of average height within each part. This gives an approximate sequential ordering that is more consistent across all possible objects and allows us to more easily fit our model on a small number of likely observation sequences.

We also exploit structure by computing additional joint features $x_{i,i-1}$ between
CHAPTER 3. PART-BASED OBJECT CLASSIFICATION

adjacent parts in the sequential ordering that will be used to learn the pairwise potentials in the HMM. The features we use here describe the geometric relationships between the two parts and include the dot product between their normals, the absolute difference in average heights, the distance between part centroids, the closest distance between points from each part, and a measure of coplanarity as defined by the mean, median, and max of the cross-fit errors between the points in one part and the planar estimate of the other.

Part labels for each part in the sequence are determined by finding the labeling that maximizes the recursive scoring function

\[
s(a_i) = \max_{a_{i-1}} s(a_{i-1}) + p(x_i|a_i) + p(x_{i-1},i|a_{i-1}, a_i).
\]

(3.1)

Where here \(p(x|Y) = x^T w_Y\), the dot product of the observed features with the learned model weights for the set of labels \(Y\). Here \(x\) may be either the unary part features or the pairwise features between parts. This recursive function is maximized by the Viterbi algorithm over the HMM.

The objective to determine the overall object class label \(c\) is

\[
\max_c \sum_i p(x_i|a_i, c) + \sum_{i,j} p(x_{i-1},i|a_{i-1}, a_i, a_j, c).
\]

(3.2)

Note here that terms in this expression include both part and object class labels
and so the estimated weights here are distinct from those used to determine the part class labels. During training the weight vectors for determining class are updated only if the corresponding part was correctly classified, otherwise we may be penalizing the wrong weight vector and convergence of perceptron training relies on updates only on correctly identified errors. For example, weight $w_{a_i,c}$ is updated only if object class $c$ is incorrect but the $i$th part was correctly classified as having label $a_i$ using weight vector $w_{a_i}$ and the preceding structure.

3.5 Experimental Evaluation

We evaluated our structured prediction model on vehicle point clouds extracted from the Wright State Ottawa dataset. A total of 222 sedans and SUVs, the two most commonly occurring vehicle categories, were used in our experiments and were partitioned into training, development, and testing splits with two-thirds of the data in training and the remaining equally split between development and test sets. Two sets of ground truth part labels were generated for this dataset to evaluate the unsupervised part segmentation and part level classification. One for the automatically generated planar part proposals from the RANSAC segmentation and another large subset with a manual segmentation of the vehicle point clouds using a 3D labeling tool in order to evaluate the performance of the automatic segmentation. The manual labels include 90 sedans and all 67 SUVs in the dataset.
of 222 vehicles. The labels using the unsupervised segmentation include merged labels like roof-hood and roof-trunk caused by errors in the automatic segmentation. These segmentation errors are generally caused by inclined surfaces with curved transitions or occlusions that limit the number of points that can be fit. Although generally not planar, interior segments are often extracted for particularly occluded objects with few visible planar parts.

For our baseline we trained support vector machine and random forest classifiers for part and object classification as well as a simple perceptron for object classification. When training for part classification these non-structured classifiers used the same part-level feature descriptors as our model but did not use any of the pairwise features between parts. For object classification we use a similar set of features defined over the local features of the entire object but not including any PCA estimation features since our overall objects are not assumed to be planar and these would vary greatly with occlusion.

Overall part classification results are presented in Table 3.1. By leveraging the HMM structure and our proposed set of pairwise part-features the structured perceptron classifier is able to consistently outperform the SVM and random forest classifiers. Even though the structured perceptron is not known to have max-margin or non-linearity properties like the SVM and random forest, the additional structural information provides an advantage over theoretically more powerful
CHAPTER 3. PART-BASED OBJECT CLASSIFICATION

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Part Acc</th>
<th>All Acc</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>76.10</td>
<td>41.50</td>
</tr>
<tr>
<td>RF</td>
<td>82.44</td>
<td>54.72</td>
</tr>
<tr>
<td>SP</td>
<td>88.29</td>
<td>56.60</td>
</tr>
<tr>
<td>Manual SVM</td>
<td>82.18</td>
<td>40.00</td>
</tr>
<tr>
<td>Manual RF</td>
<td>86.14</td>
<td>50.00</td>
</tr>
<tr>
<td>Manual SP</td>
<td>93.56</td>
<td>65.00</td>
</tr>
</tbody>
</table>

Table 3.1: Overall part classification results. Part Acc corresponds to the percentage of correctly classified parts. All Acc is the percentage of vehicles for which all parts are correctly classified. The top rows use the automatic segmentation while the bottom rows use the manually segmented data set.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Unstructured</th>
<th>Automatic</th>
<th>Manual</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>83.02</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>RF</td>
<td>79.25</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Perceptron</td>
<td>62.26</td>
<td>77.36</td>
<td>87.5</td>
</tr>
</tbody>
</table>

Table 3.2: Classification accuracy for Sedan vs SUV. Without parts the SVM achieves good accuracy and the unstructured perceptron is significantly less powerful. Using part structure the perceptron can compete with and exceed the unstructured classifiers depending on segmentation quality.

Furthermore we see a large increase in performance for the structured perceptron on completely correct classification for all parts in one object when using the manually segmented labels, showing how the structured model can better utilize a high quality part-based segmentation.

Table 3.2 shows that as expected without any structure the SVM and random forest outperform a baseline perceptron. However when a part-based segmentation is available the structured perceptron is able to significantly close the gap with
baseline methods. When using the higher quality manual segmentation without segmentation errors we are able to exceed the global descriptor baseline performance using a part-based classification approach.

### 3.6 Discussion

In this work we presented a part-based structured prediction approach for classifying objects and their semantic parts in unstructured 3D point clouds. Our segmentation algorithm is robust to many of the complexities found in point clouds and avoids non-surface segments that would be produced by a naive RANSAC segmentation. We evaluated our model on a challenging dataset of partially observed vehicles from real world lidar scans and demonstrated superior performance over the baseline methods. Additionally we note that due to some semantic part categories being orientation dependent that this work can also be interpreted as a form of object pose estimation. However we have also identified several challenges for the model in this work that have motivated us to investigate deep learning approaches for these tasks.

First, when performing a supervised parts-based classification it is necessary to generate ground truth labels for every part of every possible object of interest. This is a significant multiplicative increase in labeling efforts which may not be unique for different choices of part categories or segmentation strategies.
For example here we used approximately planar parts but the labeling may have to be regenerated if we revised our algorithm to fit curved surfaces. Secondly, the learned structure is an explicit linear approximation to a more general set of possible relations between parts that may need to be considered. An informative pairwise feature may not be found because it does not occur in the predefined expected ordering. Third, the feature representation has been manually engineered for extracting geometric information about the parts and their relations in order to determine overall object class but this does not seem to yield as significant a gain in performance on the object classification task as the part classification task. Finally, errors introduced in the unsupervised segmentation impact the classification performance and there is no mechanism to adjust the segmentation once it has been performed.

Deep learning techniques provide a framework to address these challenges in several ways, both implicitly and explicitly. A deep neural network addresses the first two challenges by implicitly learning a hierarchical representation of its inputs [Zeiler and Fergus, 2014], effectively learning features for parts and combinations of parts automatically based on the network structure. The challenges of learning feature representations for solving the target task and correcting errors introduced earlier in model are also explicitly addressed by end-to-end learning through the backpropagation algorithm. These considerations led us to move away
from a point cloud representation of our data and develop a convolutional neural
network model that can segment objects in lidar range scans.
Chapter 4

CNN-based Object Segmentation

Object segmentation in lidar scenes has previously been studied in point clustering and graph cut based frameworks [Golovinskiy et al., 2009, Dohan et al., 2015]. Based on the conclusions of our previous work, we take inspiration from recent work in RGB-D semantic segmentation [Couprie et al., 2013] and apply a similar convolutional neural network based framework adapted for lidar scenes. In particular we address a relative abundance of missing lidar data found in urban scenes caused by vehicles having reflective paint and refracting glass windows. We show that by labeling missing points in the scanning acquisition grid we can train our model to achieve a more accurate and complete segmentation mask for the scene. Additionally, we show that a lightweight set of low-level features, based on those introduced by [Gupta et al., 2014], that encapsulate the 3D scene structure computed from the raw lidar have a significant effect on performance. We evaluate our model on a lidar dataset collected by Google Street View cars over large areas.
Figure 4.1: System Overview. During training we sample positive and negative locations in large pieces of the lidar scene. For each sampled position we extract an input patch of low-level features and using our CNN model predict labels for a target patch centered on the same location. Note that the gray windows on the car are likely to be missing points and are labeled with the positive class. At test time we use a sliding window to densely segment a scene.

of New York City that we have annotated with vehicle labels for both sensed 3D points and missing lidar ray directions [Zelener and Stamos, 2016].

In the following sections we describe the procedure for generating labels in 3D images, our preprocessing pipeline for extracting input crops from large lidar scenes, the low-level input features generated for each crop, and the structure of our convolutional neural network model. An overview of the entire system can be seen in Figure 4.1. In our experiments we show that a combination of all the described low-level features provides superior segmentation performance and that missing point labels significantly improve segmentation precision.
4.1 Labeling Procedure

Previous works on object segmentation have interpreted lidar data as a 3D point cloud since each scene is constructed as a registration of scans from multiple sensor positions into one global coordinate system. However, in this perspective it is difficult to consider missing points where there is a known scanning ray direction from a particular sensor position but no distance measurement along the ray. For this reason we reframe the object segmentation problem as acting on the grid of sensor data acquisitions, allowing us to establish adjacency relations between missing and non-missing data points for a 2D convolutional neural network model.

Accurately labeling these 3D images is a challenging task since a one pixel difference on the 2D grid may correspond to a large distance in the 3D space and so labeling on the grid alone may be error prone. We have developed a labeling tool that allows us to first label the measured points in a 3D point cloud representation. The labeling software implements several tools such as allowing the selection of a volume above a plane fit, as shown in Figure 4.2, that allows us to efficiently label a large dataset for our model. We then reproject all points onto a 2D manifold where we can represent missing points based on the known resolution and motion of the sensor. Based on the 3D point cloud labels we can fill in
CHAPTER 4. CNN-BASED OBJECT SEGMENTATION

Figure 4.2: Part of a 3D scene containing two cars. While missing data due to occlusions and sensor range are obvious, it is not entirely clear from this view where missing points are located in relation to 3D points. We also show how selecting all points above a fit ground plane makes it possible to quickly and accurately label the 3D object points.

the missing point labels, as in Figure 4.3, and then verify that no labeling errors are introduced by again visualizing the point cloud.

4.2 Patch Sampling

The lidar scenes in the Google Street View dataset consist of long runs of continuous driving by the vehicle the sensors are mounted on resulting in 3D images that
Figure 4.3: Labeling missing points. Left: 2D reprojection with missing points on cars and above buildings visualized in gray. Note that some cars only have missing points on windows while others are more heavily effected. Right: Missing points within boundaries of the car are labeled.

are effectively thousands of scanlines long. These types of images are too large for a single convolutional neural network. The standard solution for 2D images of resizing down to a smaller resolution may distort the accurate 3D measurements given by the lidar sensor at depth edges and missing point positions. Rather than simply subdivide each image of our dataset we instead use a random cropping strategy to generate patches of appropriate size for a CNN that also acts as data augmentation for training the model.

We first divide each full lidar run into smaller pieces of $2 - 4k$ scanlines, avoiding segmenting target objects when possible, in order to efficiently label and preprocess the entire run. During training, for each scene piece we sample $\frac{N}{2}$ unla-
beled background positions and up to $\frac{N}{2}$ labeled object positions depending on the number of valid positions that yield a full sized patch. This biased sampling helps approximate a uniform distribution of positive and negative samples for training a standard classifier, which is necessary in our case since labeled object points are a minority of scene points.

Centered on each sampled position we generate an $M \times M$ patch of input features and a $K \times K$ patch of labels where $K \leq M$. We typically set $K$ less than $M$ so that there is sufficient support for features used to predict the object label and avoid errors due to edge effect. At test time we densely generate patches with a step size of $K$ to label the entire scene. For training we consider $T$ scene pieces and define the size of one epoch as $NT$. We continuously generate new random patches throughout training, effectively augmenting the size of our dataset without explicitly storing all possible crops. In order to reduce preprocessing computation and memory usage we reuse one set of $NT$ samples for a fixed number of training epochs before generating new samples.

4.3 Input Features

Since 3D point positions vary throughout a scene depending on the global coordinate system, it becomes necessary to generate normalized features for each patch independent of the sampled position. Similar to [Gupta et al., 2014] we generate
Figure 4.4: Signed angle feature. The signed angle for $p_2$ is $\text{acos}(\mathbf{\hat{z}} \cdot \mathbf{\hat{v}}_2) \cdot \text{sgn}(v_1 \cdot v_2)$. The yellow arc gives the angle and the dashed blue arc determines the sign.

a set of features that encode 3D scene structure and properties of the lidar sensor. We consider the depth from the sensor and height along the sensor-up direction as reliable measures and for each patch generate relative depth and height maps with respect to the centroid of all points within the patch which gives similar features for different patches robust to variation in distance from the sensor. These feature maps are then normalized based on the standard deviations within each patch and truncated to a fixed range to control for outliers such as very distance points in the background. For missing point positions we assign the maximum possible value in the fixed truncation range, allowing our classifier to learn distinctive features for these positions.

We replace the surface normal based angle feature used by [Gupta et al., 2014]
with the more lightweight signed angle feature introduced in [Stamos et al., 2012] that uses only three points for support and encodes similar local curvature properties. The signed angle feature measures the angle of elevation formed by two consecutive points which describes the orientation of the local surface. The sign is given by the dot produce of the vectors formed by three consecutive points and indicates sharp changes in local shape. Figure 4.4 gives a diagram of the signed angle definition.

Finally we also introduce another angle feature which measures the angle of elevation for each scanned point, effectively embedding the sensor orientation, and a 0/1 mask indicating which scanning grid locations correspond to missing points. Combining all of these features results in a $M \times M \times 5$ patch of low-level features for input to the CNN. An example set of features for a given patch is shown in Figure 4.5.

4.4 CNN Model

Our model follows a commonly used architecture for convolutional neural networks that consists of a sequence of convolutional layers with the ReLU activation function and max-pooling followed by a sequence of fully connected linear layers. We set the number of layers to two $5 \times 5$ convolutional with $2 \times 2$ max-pooling and two linear layers. This model is relatively shallow compared to modern state-
of-the-art 2D image models, but this design was useful in establishing a baseline for lidar data and serving as a testbed for our preprocessing pipeline and different combinations of low-level input features.

In order to accomplish single class segmentation our model predicts a $K \times K$ block of labels for a window of points centered on the $M \times M$ input patch. We parameterize this as $K^2$ independent binary classification tasks utilizing logistic regression on the representation for the entire patch produced by the final layer of the CNN. The total loss of the model is the sum of the binary cross entropy losses for each logistic regression plus an L2-regularization penalty on the weights of

![Image](image_url)

**Figure 4.5:** Input low-level features. Color values from navy (low) to yellow (high) follow the *viridis* color map shown on the far left. Top row: Relative depth, relative height, and signed angle. Bottom row: Sensor angle, missing mask, and ground truth labels in black and white.
the fully connected layers,

\[- \sum_{k=1}^{K^2} y_k \log(p_k) + (1 - y_k) \log(1 - p_k) + \frac{\lambda}{2} \sum_{l=1}^{L} ||W_l||_2^2, \tag{4.1}\]

where \(y_k\) is 1 if the \(k\)th point in the target grid is positive and 0 otherwise, \(p_k\) is the probability of the \(k\)th point being the positive class, and \(W_l\) are the weights of the \(l\)th linear layer.

For additional regularization we also apply dropout with 0.5 probability on the final layer weights. The weights of the layers with ReLU activations are initialized using the method of [He et al., 2015] and the weights for the final layer with sigmoid activation use the initialization of [Glorot and Bengio, 2010]. The model is trained by stochastic gradient descent with momentum of 0.9 and initial learning rate of 0.01. The learning rate is decayed using an exponential schedule every 350 epochs by a rate of 0.95.

4.5 Experimental Evaluation

We evaluated our model on a labeled subset of the Google R5 Street View dataset which includes a collection of 20 runs through lower Manhattan covering approximately 100 city blocks. We have annotated four of the largest runs in this collection with labels for vehicles, which are one of the most common objects in urban scenes and are a common source of missing points. The dataset was acquired
Table 4.1: Average precision of different feature combinations. D denotes depth, H denotes height, A denotes sensor angle, S denotes signed angle, and M denotes the missing mask. The model containing all feature maps gives the best overall performance.

<table>
<thead>
<tr>
<th>Features</th>
<th>Test AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>77.49</td>
</tr>
<tr>
<td>DHA</td>
<td>86.40</td>
</tr>
<tr>
<td>DHS</td>
<td>84.54</td>
</tr>
<tr>
<td>DHAM</td>
<td>84.72</td>
</tr>
<tr>
<td>DHSM</td>
<td>86.58</td>
</tr>
<tr>
<td>DHASM</td>
<td><strong>86.74</strong></td>
</tr>
</tbody>
</table>

by Street View cars with two side-mounted lidar sensors that measure 180 point scanlines in 1 degree increments on either side of the car. The labeled portion of the dataset contains over 1000 labeled vehicle instances across over 225,000 total scanlines.

For training we use the majority of the largest run that also contains over half of the labeled objects. We reserved two pieces of this run for in-sample testing. For these experiments the patch size was set to $M = 64$ with a target window of size $K = 8$. Each model was trained for 10,000 epochs which took approximately 28 hours per model on a workstation with a single Titan X GPU.

A new model was trained for a select number of combinations of the low-level input features. Average precision for each of the models on the out-of-sample test set can be found in Table 4.1 and precision-recall curves in Figure 4.6. We observe
a large increase in performance over depth alone as the input modality and best performance is generally obtained using a combination of all features. We note that there is a degradation of performance in the DHAM model over the DHA model and we suspect this is because both the sensor angle (A) and missing mask (M) feature channels are not informative about the scene geometry, indicating the importance of balancing between appearance-based features and those of other

![Feature Map Comparison Precision-Recall Curves.](image)

Figure 4.6: Precision-recall curves for input feature comparison. The top performing combinations of features throughout all possible sensitivity settings are DHSM and DHASM, which utilize our proposed signed angle and missing mask feature maps.
Features | Test AP
--- | ---
DHSM-NML | 82.71
DHSM | 84.80
DHASM-NML | 83.85
DHASM | **84.92**

Table 4.2: Average precision on non-missing labeled points only. NML denotes a model trained with no missing point labels for the vehicle class.

Scene properties. The size of our CNN model is also fixed across experiments and it is possible that those with more input features may see more benefit with expanded model capacity. Although not directly comparable with [Dohan et al., 2015] because we evaluated our work using independently labeled versions of the Street View dataset, we note that our pointwise CNN segmentation easily exceeds their local point feature baseline and appears to be competitive with their higher-level engineered features for point clusters without explicitly generating segment clusters.

Additionally, we tested the efficacy of labeling missing points for overall segmentation performance by comparing our two top models against equivalent versions trained without missing point labels. To have a fair comparison we considered only the predictions for non-missing points in our evaluation. Table 4.2 shows that the models trained with missing point labels have a significant increase in average precision even on those points that are not missing themselves. A visu-
Figure 4.7: Precision-recall curves for comparing efficacy of missing point labels. Here we see that models trained with missing point labels generally outperform those models without those labels, even on the non-missing points.

The full precision-recall curves in Figure 4.7 generally show the same result but there is a dip in performance for the DHASM model at certain tolerance levels, showing that further work is needed to understand how the selection of these features interact with the CNN model.

In order to generate visualization for qualitative evaluation we selected the DHASM model and selected a confidence threshold corresponding to 0.85 recall.
CHAPTER 4. CNN-BASED OBJECT SEGMENTATION

Figure 4.8: Comparison of models trained with and without missing labels. On the left is the DHASM model trained with missing points labeled and on the right is the same model trained without missing points labeled. For the model without missing points labeled we of course expect to see the model to disagree on missing points inside objects, for example the car on the far left. Also in order to achieve the same level of recall, the model trained without missing points must use a lower threshold and achieves lower precision.

on the test set, corresponding to a confidence threshold of 0.46 and test precision 0.73. We observed high quality segmentation on the relatively simple in-sample test scenes. General segmentation quality of common vehicles like sedans and SUVs was preserved on the out-of-sample test set, as seen in Figure 4.9, but additional errors were introduced due to more challenging vehicles like trucks with large facade-like planar regions and previously unobserved background elements such as more varied types of facades and vegetation.
Figure 4.9: Results on NYC 1 out-of-sample test scene. Colors correspond to True Positives - Yellow, True Negatives - Dark Blue, False Positives - Cyan, False Negatives - Orange. Green denotes boundary points that were not classified. Relatively high accuracy is still maintained on this challenging high traffic out-of-sample test scene. Notable mistakes in this scene include parts of large vehicles, like trucks and buses, with mostly planar surfaces that may look locally similar to facades, as well as impatient pedestrians crossing the street through traffic.

4.6 Discussion

In this work we presented a convolutional neural network model and training pipeline for segmentation of large-scale urban lidar scenes acquired by vehicle-mounted sensors. In our evaluation we show that by explicitly labeling missing lidar data points we are able to achieve a superior segmentation mask both in terms improved precision on non-missing points and coverage of probable missing points. Furthermore we have shown that the choice of input features is a sig-
nificant factor in this task and the additional input features we present like signed angle and missing mask can improve performance.

For future work on segmentation it may also be possible to impute expected depth values for missing points in the same we predict semantic labels. However in order to train this model it would require measuring ground truth values for missing points in a controlled environment or utilizing synthetic data from a 3D scanning simulator.
Chapter 5

Depth-conditioned Object Localization

In the previous chapter we developed a deep learning approach for generating a semantic segmentation mask for a lidar scene. While this mask was relatively high quality it still contained segmentation errors at boundaries and small patches of false detections at various unexpected locations within the scene. We suspect these errors are partly due to the shallow low-resolution CNN architecture used in our previous experiments but also partly due to how the segmentation task itself is formulated. In [Luo et al., 2016] they show that the effective receptive field of convolution activations is smaller than the maximum possible receptive field, which makes modeling of long range relationships reliant only on the densely connected layers which only have access to the coarse features of the final convolutional layer. The assigned label is then effectively the result of a relatively local set of features that does not utilize larger scale structure.
We tackle these challenges by first addressing the task of object localization, determining where in an image objects are located. This task requires a significantly more sophisticated objective function for good performance but ideally will give tighter boundaries and reduce locally plausible false detections by requiring global structure as determined by the width and height of an object’s bounding box. For this task we adopt the state-of-the-art YOLOv2 localization model [Redmon and Farhadi, 2017] which is a significantly larger CNN, containing over twenty layers compared to the four layers used in our previous work.

Additionally, rather than simply relying on a localization model designed for 2D images we experiment with a variation of the model objective that requires it to leverage the 3D properties of lidar images. To that end we require the model to estimate the mean depth from the lidar scanner for every object instance point. This makes the localization model directly estimate a 3D property of an object instance that requires some distinction between foreground and background within the bounding box crop. There is also an empirical correlation between an object’s width and height in image space and its physical distance from the sensor due to the angle of projection which we would expect to regularize the model’s bounding box predictions. This auxiliary task of depth estimation combined with localization also provides the necessary minimal information for the task of collision avoidance for mobile robotics applications.
In the following sections we describe modifications to our preprocessing procedure to adapt it to the YOLOv2 model as well as how we generate anchor boxes [Ren et al., 2015] which are priors for bounding box parameters. We generate these priors for both bounding box width and height in image space as well as depth in 3D space. We then briefly detail the differences between the YOLOv2 model architecture and the architecture of our previous work and the details of our extension to the YOLO objective function to regress mean depth. Finally in our experiments we show that we are able to estimate the mean depth of an object instance during localization with small error, that we achieve similar localization performance to our baseline while performing this additional regression with the same model architecture, and that for a more sophisticated form of the localization loss function we observed a regularization effect leading to faster model convergence.

5.1 Lidar Preprocessing

Our basic approach to preprocessing lidar images is essentially the same as our previous work. However we have made several simplifications and adaptations to make it similar to the preprocessing steps performed for the YOLOv2 framework.

For simplicity the initial features we utilize are only the depth, height, and signed angle which contributed most to the performance of our previous work.
Figure 5.1: Ground truth crops for localization from Street View training set. Colors represent a combination of the depth, height, and signed angle features. Missing points take the maximum value for these features and are shown in white. The $13 \times 13$ black grid represents positions corresponding to activations of the final convolutional layer. The red highlighted grid cells contain the center point of a ground truth box. Note that the far left sedan is not part of the ground truth due to the majority of its bounding box being clipped out of this crop.

over the simple depth-only baseline. Rather than normalize the features using a unit normal assumption for each crop, we adopt the simple strategy YOLOv2 uses for color images of simply scaling down values to the range $[0, 1]$. To do this we estimate from the training data the maximum depth and height values observed and select a threshold that retains a majority of the observed data, this threshold is roughly $40m$. For signed angles we simply scale based on the range of valid signed angle degrees from $[-180, 180]$. The feature values for missing points are still set to the maximum value in the range which is now 1 for all features.

For the localization task we selected a larger crop size, $160 \times 160$, so that it is more likely for whole objects to be contained within the crop while still main-
taining a 1 : 1 aspect ratio. We selected a crop size slightly smaller than the full 180 points per scanline in our dataset to avoid low quality noisy measurements at the top and scans of the sensor platform itself at the bottom of the lidar images. Because the YOLOv2 architecture is designed for higher resolution color images we use bilinear interpolation to rescale the feature crops to $416 \times 416$, the default size for YOLOv2. Examples of these sampled crops can be seen in Figure 5.1.

When sampling crops for training we consider a crop to be positive if at least one ground truth box clipped to the crop window contains over half of the area of the original ground truth box. For training the localization model we only sample positive crops. Unlike photographs used in most 2D image benchmarks our scans come from continuous mapping with no camera operator to focus on specific objects. This means that there typically exists ample negative space even in positive crops and mining negatives is likely unnecessary and may even slow down training for this task.

## 5.2 Depth-conditioned Anchors

While earlier neural networks for localization like Overfeat [Sermanet et al., 2014] and YOLOv1 [Redmon et al., 2016] attempt to directly regress parameters of a bounding box like the position of each edge within an image, more recently better performance has been found using anchor boxes in Multbox [Erhan et al.,
2014] and Faster R-CNN [Ren et al., 2015] that act as a prior on the bounding box parameters. In this case it is only necessary to regress a residual to the closest prior rather than to directly regress the target value from the space of all possible values. Since an object can appear anywhere in an image the anchors are only estimated for bounding box width and height while its exact position are determined using a prior-free method.

However in 3D images we have access to the depth dimension over which we can estimate a parameter. This was done in [Song and Xiao, 2016] by voxelizing the scene space and having a depth prior for each anchor box, but this is sensitive to the orientation voxel tesselation of the scene and their model used computationally expensive and coarse resolution 3D spatial convolution operations. We chose instead to estimate the mean depth of measured 3D points on an object instance which is more robust to variation in object orientation. This allows us to avoid using 3D convolutions by selecting priors for mean depths rather than explicitly computing features densely for spatial position in depth that are mostly empty.

The use of anchor boxes can be thought of as a discrete-continuous hybrid combination of classification and regression. In general for each anchor $k \in [1, K]$ and parameter $p \in [1, P]$ with prior $a_{kp}$ and a corresponding regressed target $t_{kp}$ the corresponding predicted value is defined as $v_{kp} = f_p(a_{kp}, t_{kp})$ for some transfer function $f_p(\cdot)$. The objective for the regressed target is determined by the
inverse of the transfer function. Let \( v_k \) be the vector of all predicted values for anchor \( k \). Then given a valuation function \( Q(\cdot) \) of the predicted values for each anchor, the final prediction is determined by a discrete maximization,

\[
v = \arg \max_{k \in K} Q(v_k).
\] (5.1)

In YOLOv2 the bounding box width and height are determined by a transfer function with the exponential form \( f_p(a_{kp}, t_{kp}) = a_{kp} e^{t_{kp}} \). In our experiments we use the same form for the additional mean depth parameter that we estimate. Note that in our formulation each anchor box contains a width, height, and mean depth prior rather than a separate set of anchors for mean depth alone. This allows us to utilize the correlation between bounding box scale and depth, adding only a linear number of parameters for the mean depth regression rather than the multiplicative growth caused by an additional set of anchors, i.e. the number of parameters would be \( K_{wh} \times K_{\text{depth}} \). We set the number of anchors \( K \) to be 5 which is the value used in most experiments for YOLOv2. Here the valuation function for bounding boxes involves non-maximum suppression among all predicted bounding boxes based on their predicted confidence, predicted class probability, and pairwise IOU. The depth estimate does not impact the valuation function in our model.

There are two common methods for selecting anchor box priors, either design them manually to provide a broad coverage of possible boxes at test time or
unsupervised clustering on the training set with the assumption that priors computed on the training set will be representative to boxes found at test time. In YOLOv2 they use $k$-means clustering on the training set with a distance metric of $1 - \text{IOU(box, centroid)}$. Because our anchor boxes also contain a depth prior we formulated an affinity function as,
Figure 5.3: Visualization of anchor boxes with mean depths. From near to far: (green) almost square at 3.3m, (purple) very wide rectangle at 4.1m, (blue) smaller square at 5.1m, (red) smaller wide rectangle at 5.2m, (yellow) smallest rectangle at 9.2m. Compared to YOLOv2 anchors on the COCO dataset our boxes have smaller height, due to the $180^\circ$ vertical field of view of the lidar sensor, but are generally larger and wider in image space which reflects the typical dimensions of vehicles versus more general objects found in COCO.

\[
\text{affinity}(a, b) = \alpha \text{IOU}(a, b) + (1 - \alpha) \frac{\min(a_{\text{depth}}, b_{\text{depth}})}{\max(a_{\text{depth}}, b_{\text{depth}})},
\]

We set $\alpha = 0.75$ in order to give more weight to a clustering that supports the localization objective. Instead of using $k$-means clustering directly we perform spectral clustering using the affinity function with some tuning of the number of components used for the spectral embedding in order to avoid clusters with outliers. We visualize the result of this clustering in Figure 5.2 and the anchor boxes themselves in Figure 5.3.
5.3 CNN Model

The YOLOv2 architecture incorporates several recent innovations in the design of convolutional neural networks compared to the baseline model used in our previous work. The design of the network is fully convolutional, meaning that there are only convolutional layers and no densely connected layers. This allows the network to take inputs of varying spatial dimensions, however for our experiments we only use one image size. Fully convolutional networks can also be thought of as applying a model designed for a smaller image size at many locations of a larger image but with the benefit of parallelizing the operations on the GPU running the model. Fully convolutional networks typically feature a $1 \times 1$ convolution as the final layer that is functionally equivalent to a dense layer for a small enough input image.

The model itself is also significantly larger than our previous baseline containing 23 layers versus the 4 in our previous model as well as 5 pooling layers instead of just 2 pooling layers. This is due in part to the fully convolutional design, which eliminates dense layers with many parameters, and partly due to the use of a bottleneck design between convolutions. Here we describe convolutions as a tuple of width and height dimensions, input feature dimension, and output feature dimension which correspond to the shape of the convolutional kernel. The
bottleneck design replaces large convolutions of the form \((3, 3, D, D)\) at large feature dimension \(D\) with a sequence of three convolutions: first one with kernel \((3, 3, D, D/2)\), then a \((1, 1, D/2, D/2)\), and finally a \((3, 3, D/2, D)\). This uses slightly more parameters than the original convolution but introduces a sequence of separate layers that produce an identically shaped output which allows for an equivalent yet deeper network.

Instead of a simple ReLU activation the YOLOv2 model uses leaky ReLU activations [Maas et al., 2013]. Rather than outputing zero for negative values like the original ReLU, the leaky ReLU introduces a nonlinearity by assigning a slope \(\alpha \neq 1\) to negative inputs. This allows some information from negative activations to be utilized by the model while still introducing a nonlinearity. The original ReLU can be thought of as a leaky ReLU with \(\alpha = 0\). We adopt the setting of \(\alpha = 0.1\) from YOLOv2.

While most traditional CNNs have been entirely feed forward with one convolutional layer after another, YOLOv2 includes in its design what is commonly referred to as a skip connection which combines the output of layers that do not directly follow each other in sequence. Specifically there is a skip connection from the convolutional layer before the final pooling layer to the end of the network that works by concatenating these activations with the ones before the final layers. This allows the very final layers of the network to consider features at two
scales. The higher resolution features are reorganized using a method similar to the periodic shuffling operation described by [Shi et al., 2016]. Every $2 \times 2$ spatial block of features is reorganized into a single vector. Because of the structure of this skip connection the network is restricted to images whose spatial dimensions are a factor of 32, the downscaling factor due to the five $2 \times 2$ max pooling layers.

For regularization instead of dropout, YOLOv2 uses batch normalization [Ioffe and Szegedy, 2015] after each convolutional layer. Batch normalization essentially renormalizes the activations of each layer using batch statistics similar to the normalization done during preprocessing on the initial input features. This operation tends to be more computationally expensive than dropout but allows for deeper models without zeroing out as many gradients. Additionally we now use L2 regularization on the weights of every convolutional layer rather than just the dense layers in our previous model.

Finally for our experiments with depth estimation we simply add to the objective function an extra squared error term for the mean depth estimation, $(t_{kd} - \hat{t}_{kd})^2$. The reference target $\hat{t}_{kd}$ is determined by the inverse transfer function $\hat{t}_{kd} = f^{-1}(v_{kd}; a_{kd}) = \log(v_{kd}/a_{kd})$ where $v_{kd}$ is the actual mean depth value we would like to predict using anchor $a_{kd}$.
5.4 Experimental Evaluation

We evaluate our localization model on the Google Street View dataset used in our previous work, however we have changed the training and test splits. We found that because localization is more sensitive to global structure we required more example instances with more variation between examples. We now use an additional run from a different location for training to add diversity and have labeled an additional run to have a sufficient number of instances for testing. Each vehicle instance is now coarsely categorized into one of five categories that generally correlate with vehicle scale: sedan-suv, bike, mid-vehicle, large-vehicle, and moving. The mid-vehicle category includes vehicles like vans, minibuses, and pickup trucks while the large-vehicle category includes full size buses and delivery trucks. The moving category includes vehicles whose scans were distorted by the scanning processes due to relative motion of those vehicles to the scanning platform, however we observed confusion in our models between this category and the sedan-suv category since the majority of these are locally identifiable as one of those majority vehicle categories. The bike category includes bicycles and motorcycles but these are relatively rare in our dataset and the handful in our test set are generally not detected by our models.

For training we defined a single epoch as 8192 random crops, that is about
Table 5.1: Performance with rescore confidence loss. We found that with this option enabled, our proposed localization + mean depth estimation model was similar to baseline performance however it yielded the lowest error on mean depth estimation. The average prior model provided better localization at the cost of increased depth error.

256 crops per scene piece. Based on training several models for extended periods, we found that for our dataset our models would converge within 64 epochs, approximately two days of training on a single Nvidia Titan X Maxwell GPU. For evaluation on the test set we densely crop along the middle 160 of the 180 pixels in height with a step size of 80 across the scene, producing a total of 1559 ground truth boxes across all crops. We report performance in terms of Object AP, which is the average precision of object detections independent of class, where a true positive detection has an IOU with a ground truth box greater than 0.5 and that box has not already been matched by a higher confidence prediction. We also report Class AP where the predicted class must also match the ground truth class and the mean IOU score of true positive detections. Qualitative results of our best performing models in terms of Object AP are shown in Figure 5.4.

We found that performance of the model was significantly impacted by the
Figure 5.4: Localization results on street view test set. Left: Predicted boxes with class label and confidence. Right: Evaluation of predicts with TP-C (red) denoting true positive prediction with correct class, FP (cyan) denoting false positive predictions, and FN (green) denoting false negatives.

rescore confidence option found in YOLOv2, which sets the target confidence value of the model to be the IOU score of the predicted box with the closest ground truth box. With this optional enabled, we found that the model yields better estimates for mean depth however has reduced performance for object detection. We found little difference in the mIOU of true positives between these models so this is likely due to a subtle effect from requiring a more accurate confidence prediction that yields a better signal for depth estimation but makes localization more demanding. We found the largest impact of our proposed modification was when this option was enabled. We summarize the results for the rescore confidence
Table 5.2: Performance without rescore confidence loss. Localization performance is superior to using the rescore confidence loss at the cost of depth estimation error. While average IoU of true positive detections is largely the same across all models we see that our proposed prior has more difficulty modeling less confident detections than the simpler priors, perhaps overfitting on low confidence correlations between class and depth.

<table>
<thead>
<tr>
<th>Model</th>
<th>Object AP</th>
<th>Class AP</th>
<th>TP mIoU</th>
<th>Depth MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>77.57</td>
<td>60.67</td>
<td>76.92</td>
<td>N/A</td>
</tr>
<tr>
<td>MeanDepth Cluster Prior</td>
<td>77.21</td>
<td>49.67</td>
<td>76.40</td>
<td>$\mu = 82.01 \text{ cm}^2$</td>
</tr>
<tr>
<td>MeanDepth Unit Prior</td>
<td>77.63</td>
<td>61.27</td>
<td>75.07</td>
<td>$\mu = 52.72 \text{ cm}^2$</td>
</tr>
<tr>
<td>MeanDepth Average Prior</td>
<td>78.49</td>
<td>62.99</td>
<td>77.13</td>
<td>$\mu = 68.15 \text{ cm}^2$</td>
</tr>
</tbody>
</table>

With the rescore confidence option disabled we are able to achieve significantly better performance for localization and there is less of a difference between our proposed model and the baseline while also achieving the task of object depth estimation. As we see in Table 5.2, our proposed model achieves performance close to the baseline while jointly solving the object depth estimation problem using a minimal number of extra parameters. Furthermore, in Figure 5.5 we show how the key metrics of Object AP and MSE depth error evolved over training and discuss general trends between models.

### 5.5 Discussion

We have presented a system for simultaneous localization and depth estimation of object instances in 3D lidar images. Our experiments show that the addition
Figure 5.5: Object AP and depth errors over training epochs for localization. Our proposed model with rescore confidence achieves lower mean depth estimation error while tending to have slightly lower object AP. For depth estimation we found that our proposed cluster prior consistently yielded low error throughout training. We note that for object AP the trends are not as consistent and there is more variance in performance across models and across model checkpoints.

A depth estimation regression target can be combined with an existing localization objective, and can improve localization performance when using appropriate priors.

Further research in this direction may investigate using a larger number of anchor boxes as the restriction of 5 was arbitrarily chosen to match the number used by YOLOv2. We suspect that some of the performance shortfalls of our method are due to object detections at test time exhibiting more variance in terms of bounding box dimensions and depths than expected and additional anchor boxes may compensate for this discrepancy. Additional 3D object properties can also be estimated like the minimum and maximum object instance depth along
the viewing direction. The selection of predicted boxes can be modified to allow boxes that overlap in image space but are separated by a large distance in depth. Finally, performing this kind of preliminary 3D object property estimation when using this model as a region proposal network for other object identification tasks like segmentation and 3D pose estimation should be investigated.
Chapter 6

Conclusion

6.1 Discussion

In this dissertation, we have defined the fundamental object identification tasks required for basic applications related to object understanding in images.

- Object localization - Detecting and locating an object within a scene image, typically by regressing an object bounding box.

- Object segmentation - Separating the points or pixels of an object from other background elements, usually by assigning a label to each element.

- Object classification - Determining an object’s high level semantic category.

- Object pose estimation - Estimating the location and orientation of an object within the 3D world, for example with an oriented 3D bounding box.

We have reviewed the literature on how these tasks are addressed specifically
in 3D images that are acquired by lidar sensors and RGB-D cameras, including a discussion of the transition from point clustering based methods based on the traditional machine learning with feature engineering pipeline to a deep learning approach using neural networks. Our own contributions to the literature include,

- A part-based segmentation and object classification system with semantic pose estimation using a point clustering approach.
- A convolutional neural network approach to dense semantic segmentation of a lidar image with modeling of the semantic class of missing points.
- A CNN for localization of objects in a lidar image that is conditioned on the object’s estimated 3D distance from the sensor position.

In our work we have observed that point clustering systems designed without the ability to correct errors in the clustering phase are not able to perform to their hypothetical potential. Recent work has addressed this by performing structured prediction over more fine-grained point clusters [Wu et al., 2014] or by utilizing a multi-objective deep learning model that can propagate an error signal for all tasks simultaneously [Dai et al., 2016].

Additionally we have observed that by utilizing the 3D properties of the sensed objects we can improve performance on the object identification tasks. This in-
includes utilizing initial feature representations that allow a model to better understand the relationship between sensed data, the sensor, and the environment such as orientation with respect to the gravity direction or where the sensor image has missing data. Furthermore, estimating properties of an object’s 3D geometry and pose as an additional objective can support tasks like localization and segmentation that are solved within image space.

### 6.2 Open Problems

Based on the framework for object identification that we have established and the analysis of our works detailed in this dissertation, we will make recommendations for future work on object identification within a single 3D image. We will also discuss extensions of this topic to new research directions in the broader area of 3D scene understanding and going beyond a single 3D image.

One natural extension to our work is to jointly solve all of the object identification tasks within a single deep learning framework. By utilizing our localization system as a region proposal network we can perform dense instance segmentation, classification, and 3D pose estimation on the proposed regions. Systems that solve several subsets of these tasks have already been applied successfully to 2D images [Dai et al., 2016, Poirson et al., 2016] but we are not aware of any system that solves all these tasks jointly for a single 3D image where it should
be possible to retrieve accurate 3D object properties from the data. One reason such a system has not yet been proposed is due to only the recent availability of large scale datasets that contain both per point segmentation labels as well as oriented 3D bounding boxes like SUN RGB-D and SceneNN [Song et al., 2015, Hua et al., 2016]. Unfortunately for the urban setting there does not yet exist a large scale publicly available dataset with ground truth annotations for all of these tasks, KITTI [Geiger et al., 2013] comes closest but does not contain dense segmentation labels. We see the development of such a benchmark dataset as essential for measuring progress in this area.

There are also several additional tasks that can be performed on a single image that we did not include as fundamental object identification tasks but may also lead to natural extensions. These include part-based segmentation, 3D object reconstruction, semantic segmentation of the entire scene, and more fine-grained classification beyond high-level categories such as a taxonomy based classification or a short text description. This set of tasks may only be required for certain applications and also require even more ground truth annotation than is available in existing datasets. However our research suggests that addressing each additional task will likely lead to a more complete scene understanding and better overall performance. Some of these tasks have been addressed for special cases such as human pose estimation. However we suspect that it is infeasible to densely
annotate all these properties for any arbitrary object of a given dataset and that for these tasks a significantly different framework incorporating unsupervised or semi-supervised learning may be necessary.

Finally, for many applications such as real-time robotic navigation or analysis of densely scanned scenes it is necessary to use methods that go beyond a single 3D image. We may either consider a video sequence of 3D images that contain scans of overlapping regions, or a single registered 3D scene where many views of the same scene regions that have been registered together. In these settings the application of a 2D spatial convolutional neural network like those we have used in our work is not as straightforward. Solving additional tasks like object tracking over time may be necessary as well as investigating alternate models like recurrent neural networks, 3D spatial convolutions, or graph-based convolutions.
Bibliography


