Object Localization, Segmentation, Classification, and Pose Estimation in 3D Images using Deep Learning

by

Allan Zelener

A dissertation proposal 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.

2016
©2016

Allan Zelener

All Rights Reserved
ABSTRACT

Object Localization, Segmentation, Classification, and Pose Estimation in 3D Images Using Deep Learning

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. Recent advances in deep learning lead us to propose a model capable of being optimized for all of these tasks jointly in order to reduce potential errors propagated when solving these tasks independently.
## Contents

1 Introduction ................................................. 1

2 Completed Work ............................................ 6
   2.1 Part-Based Object Classification ....................... 7
   2.2 CNN-Based Object Segmentation ....................... 21
   2.3 Related Work ........................................... 37

3 Proposed Work ............................................. 41
   3.1 Timeline for Completion ............................... 49
Chapter 1

Introduction

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 that typically acquire 2D images 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.

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. Next this localization may be 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. 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 dis-
tance 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 reference models. However despite all these
issues there are inherent advantages to using these sensors. The 3D depth mea-
surements 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., 2016b]. 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 we are able to train machine learning models that solve the object identification tasks. Deep learning models using 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]. 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., 2016b] uses a multi-task cascade for object
localization, segmentation, and classification as shown in Figure 1.2.

Here we propose to extend deep learning methods to the domain of 3D images and develop a model that incorporates the tasks of object localization, segmentation, classification, and pose estimation with a design based on recently proposed techniques for these tasks. Additionally, we would like to experiment with domain adaptation from synthetic data given the limited availability of large-scale labeled 3D datasets and address the challenges posed by missing data in 3D images. In Chapter 2 we describe in detail our completed work on these problems and give a brief review of related work in 3D computer vision that will motivate and inform the design of our proposed model. Chapter 3 describes the details of the proposed work and our proposed experiments for evaluating the model. The timeline for completing the proposed work is described in Chapter 3.1.
Chapter 2
Completed Work

Identifying objects in images is a topic that has been extensively covered in the computer vision literature from a variety of perspectives. Our survey [Zelener, 2014] has examined prior work on object classification and segmentation in 3D range scans which could broadly be categorized into either 3D point clustering methods for outdoor scenes or 2D image based methods for indoor RGB-D scenes. Our two prior works towards the proposed dissertation have together investigated both of these approaches for object classification in urban LIDAR scans. In one approach [Zelener et al., 2014] we utilize planar point clustering to estimate object parts with a structured prediction model to jointly classify the object parts and overall object category. We have also developed a 2D convolutional neural network approach on the scanning acquisition grid of urban LIDAR to perform semantic segmentation over missing data points [Zelener and Stamos, 2016]. In the following sections we will describe our prior work in more detail and then
review some recent related work that has been released since our survey that will inform our proposed work.

2.1 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 synthetic 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.

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.

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 neigh-
bors 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.

**Part Segmentation**

For part segmentation we assume that our objects of interest have roughly piecewise 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
Figure 2.1: 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.

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 collinear. However due to occlusions and transpar-
ent 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 don’t 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 2.1.

**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 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.
Figure 2.2: 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.

**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 unstructured 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 2.2.

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 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
Chapter 2. Completed Work

15

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 parts in the sequence are determined by finding the labeling that maximizes the recursive scoring function

\[ s(a_i) = \max_{a_{i-1}} s(a_{i-1}) + p(x_i|a_i) + p(x_{i-1,i}|a_{i-1}, a_i). \] (2.1)

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

\[ \max_c \sum_i p(x_i|a_i, c) + \sum_{i,j} p(x_{i-1,i}|a_{i-1}, a_i, c). \] (2.2)

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.

**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
CHAPTER 2. COMPLETED WORK

<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><strong>88.29</strong></td>
<td><strong>56.60</strong></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><strong>93.56</strong></td>
<td><strong>65.00</strong></td>
</tr>
</tbody>
</table>

Table 2.1: 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.

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 proposed 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 2.1. 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
Table 2.2: 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.

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 classifiers. 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 2.2 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.
CHAPTER 2. COMPLETED WORK

Conclusion

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. 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 ex-
CHAPTER 2. COMPLETED WORK

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.
2.2 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 of New York City that we have annotated with vehicle labels for both sensed 3D points and missing LIDAR ray directions.

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
CHAPTER 2. COMPLETED WORK

22

Figure 2.3: 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.

our convolutional neural network model. An overview of the entire system can be seen in Figure 2.3. 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.

Labeling Procedure

Previous works on object segmentation has 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
Figure 2.4: 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.

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
Figure 2.5: 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.

so labeling on the grid alone may be error prone. We’ve 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 2.4, that allows us to efficient label a large dataset for our model. We then reproject all points on to 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 the missing point labels, as in Figure 2.5, and then verify that no labeling errors are introduced by again visualizing the point cloud.
CHAPTER 2. COMPLETED WORK

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 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 $\frac{N}{2}$ unlabeled 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.

**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 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
Figure 2.6: Signed angle feature. The signed angle for $p_2$ is $\cos(\hat{\mathbf{z}} \cdot \hat{\mathbf{v}}_2) \cdot \text{sign}(\mathbf{v}_1 \cdot \mathbf{v}_2)$. The yellow arc gives the angle and the dashed blue arc determines the sign.

Truncated to a fixed range to control for outliers such as very distant 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 product of the vectors formed by three consecutive points and indicates sharp changes in local shape. Figure 2.6 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 2.7.

![Figure 2.7: 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.](image)
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 the fully connected layers,

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

(2.3)

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.

**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 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.
Table 2.3: 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>

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 2.3 and precision-recall curves in Figure 2.8. 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 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

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

Figure 2.8: Precision-Recall curves for feature map 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.
CHAPTER 2. COMPLETED WORK

<table>
<thead>
<tr>
<th>Features</th>
<th>Test AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>DHSM-NML</td>
<td>82.71</td>
</tr>
<tr>
<td>DHSM</td>
<td>84.80</td>
</tr>
<tr>
<td>DHASM-NML</td>
<td>83.85</td>
</tr>
<tr>
<td>DHASM</td>
<td>84.92</td>
</tr>
</tbody>
</table>

Table 2.4: Average precision on non-missing labeled points only. NML denotes a model trained with no missing point labels for the vehicle class.

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 2.4 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 visualization of this difference is shown in Figure 2.10. The full precision-recall curves in Figure 2.9 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
CHAPTER 2. COMPLETED WORK

34

Figure 2.9: 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.

In order to generate visualization for qualitative evaluation we selected the DHASM model and selected a confidence threshold corresponding to 0.85 recall 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 2.11, but ad-
Figure 2.10: 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.

ditional 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.

**Conclusion**

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
Figure 2.11: 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.

points. Furthermore we’ve shown that the choice of input features is a significant factor in this task and the additional input features we present like signed angle and missing mask can improve performance.

This work has described the first steps towards applying a deep learning framework to LIDAR data. In our proposed work we seek to extend this framework to additional object identification tasks and further incorporate the 3D properties of our data in the design and structure of a CNN model. It may also be possible
to impute expected depth values for missing points in the same way we predict their semantic labels, however this would require measuring ground truth values in controlled scans or the use of synthetic data.

2.3 Related Work

While there has been some additional work in the direction of 3D point clustering methods for object segmentation and classification [Dohan et al., 2015], the body of work that has received more attention and is most related to our proposed work lies at the intersection of 3D computer vision and deep machine learning. Not all of these works focus on the object identification tasks or utilize 3D sensors for input, but they share common deep learning methodologies and relate their given task to the 3D world and as such may influence our proposed work. We shall primarily describe recent work on object identification in 3D images which is most closely related to our proposal and also briefly survey work on other 3D vision tasks including estimation of 3D properties in 2D images.

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 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 identi-
cical 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 be-
tween 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
earlier 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 used because large datasets with pose annotations were not
yet available. 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 sim-
ilarity between objects [Bansal et al., 2016]. One unifying theme in all of these
works is that low-level geometric properties like depth and surface normals are re-
lated 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 approach like that of [Su et al., 2015] which generates a representation by pooling 2D convolutional features from multiple viewpoints surrounding the object. 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 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. Overall this line of work is promising for its ability to process more complete 3D data and learn more fine-grained 3D relations in densely packed 3D scenes, but further work is needed to enable efficient high resolution representations and robustness to variations in object pose.
Chapter 3

Proposed Work

Motivated by earlier work on multitask learning [Caruana, 1998, Collobert and Weston, 2008] and the recent success of joint localization and segmentation systems, we propose a model for joint object localization, segmentation, classification, and pose estimation in 3D images. We identify these as the set of basic tasks necessary for higher level applications involving objects of interest in a 3D environment. Our proposed model will be based on several recent innovations in neural network component design including fully convolutional networks [Long et al., 2015], region proposal networks [Ren et al., 2015], spatial transformers [Jaderberg et al., 2015], and sub-pixel convolutions [Shi et al., 2016].

To reasonably limit the scope of this proposed work we impose the following restrictions which will be reserved for future work, note however that we consider the proposed work as a necessary prerequisite for the tasks we exclude. In this proposal we limit ourselves to single 3D image data such as a RGB-D camera
frame or a single sweep of LIDAR scanlines. This excludes both video sequences of 3D images and densely registered 3D scenes from multiple views as possible data sources. We also exclude the task of complete shape reconstruction since it typically requires multiple views, reference database matching, or a generative model and it may be a significantly more resource intensive task that would limit the practical design of our model. Although, we may still consider the task of reconstructing missing data points that should have been visible by the sensor but were not measured due to limitations of the active sensing technology.

We intend to continue using the Google Street View dataset that was used in our previous work and will further extend it with oriented bounding boxes that capture the pose for each object. Additionally, we’ve investigated publicly available 3D datasets in the urban LIDAR and indoor RGB-D settings. The KITTI dataset [Geiger et al., 2013] is a benchmark dataset for autonomous driving and contains oriented 3D bounding boxes for objects on the road such as cars, trucks, and pedestrians. Unfortunately the KITTI dataset does not contain an official semantic segmentation benchmark but there are some annotated subsets of the data that we may use. Synthia [Ros et al., 2016] is a large scale synthetic dataset for semantic segmentation of urban scenes, however it does not appear to include object pose annotations. Because of these limitations if we were to use these datasets for training we would consider combining them with domain adaptation [Ganin
et al., 2016] and pretrain certain tasks separately. The large-scale indoor RGB-D datasets like SUN RGB-D [Song et al., 2015] and SceneNN [Hua et al., 2016] contain all of the necessary ground truth labels and can be used to train and evaluate our proposed model without additional modification. SceneNN also provides a mesh reconstruction of its RGB-D scenes which may be utilized as an approxi-
mate ground truth for missing point estimation.

For our baselines we will build independent convolutional neural network models for each of the tasks based on efficient model architectures that compete with the state-of-the-art, for example our previous work for segmentation or the YOLO localization and classification network [Redmon et al., 2016] for which we have already implemented a fully convolutional variant for 2D bounding box estimation with preliminary results shown in Figure 3.1. For 3D images we will extend this localization to predict axis-aligned 3D bounding boxes. Either the output of the baseline localization network or random crops based on ground truth labels may be used as the input for classification, segmentation, and object pose estimation baselines.

Although we expect to see a benefit in the shared representation of a network that jointly solves the object identification tasks, we note that the state-of-the-art networks for these tasks have architectures that have been specialized in significantly different ways. For example, classification networks typically contain many pooling layers for translation invariance and produce a low dimensional representation for the likelihood of each class whereas the performance of segmentation networks degrade with excessive pooling and the output needs to have the same spatial dimensions as the input image. Our strategy to address these concerns is to design our network to prioritize an accurate instance-level segmentation
which may be most useful for upstream tasks while mitigating potential shortfalls for other tasks with specialized branches from the main computational path. To that end we’ve identified several recent innovations in neural network design that can aid in this goal and also have interesting implications for adaptation to 3D images.

section*{Localization in Depth}

One of the main tasks in the pipeline is localization since some scenes are sparsely populated with objects and so it is beneficial to further process only those regions where an object has been localized for the remaining tasks. Previously localization has been performed in two stages, region proposal where candidate locations are generated either through an external method or a region proposal network, and then object detection where the features from the proposed region are used to predict confidence in an object’s presence in the region as well as a more refined bounding box containing the object. Such an approach can be found in the Deep Sliding Shapes architecture [Song and Xiao, 2016] which has a region proposal network with the property that proposals of different scales are generated based on representations from different layers of the work. Architectures like YOLO and Single Shot Detector [Liu et al., 2016] attempt to avoid region proposal entirely, however SSD uses the same idea as Deep Sliding Shapes and generates multiple localizations after each downsampling step in the CNN. For this proposal
we would like to investigate a localization method that adapts this idea to 3D images by predicting proposals at multiple depths as well as scales throughout the network. Rather than adopt a full 3D volumetric approach we will instead partition the observed viewing volume into slices along the depth direction and associate each ground truth location with overlapping slices. The network will still perform efficient 2D convolutions but will be expected to predict a confidence and location for each depth slice as well as each spatial grid cell. We expect this depth conditional approach to separate objects that overlap in the 2D projection and better estimate 3D locations.

**Spatial Transformers for Pose Estimation**

There has been some prior work in 3D pose estimation, including a recent extension of the Single Shot Detector [Poirson et al., 2016], and one of the most promising approaches is that of the Spatial Transformer Network [Jaderberg et al., 2015]. Unlike other approaches that take the resulting features from a localization network and directly predict the pose of the localized object, the STN generates an affine transformation for the coordinates of a sampling grid on the input feature map. When applied to an input image this transformation tends to both localize an object of interest and transform the object to a pose that is helpful for optimizing the network objective. For classification networks this tends to be a canonical
Figure 3.2: Spatial Transformer Network on 3D MNIST input. The STN generates a transformation visualized as a bounding box in the 3D voxel input which defines where a 3D sampling grid is applied. The sampled features are then flattened along one dimension to produce a 2D image that can be classified by the rest of the network which is a conventional CNN.

object pose. An example of a STN on 3D voxel input is shown in Figure 3.2. Not only does this component perform further localization but the inverse transformation is likely to yield something very close to the object pose, and even if it does not it may make it easier to find a more fine-grained adjustment. We note however that it is not clear how the STN interacts with the features in the later layers of a CNN, that is whether the geometric interpretation still applies, but we would still like to investigate its suitable for this task. We are also interested in a recent variant that applies spatial transformers in a convolutional fashion [Choy et al.,
2016], allowing for many transformations on a single feature map.

**Subpixel Methods for Segmentation and Detection**

Fully convolutional neural networks with up-sampling convolutions [Long et al., 2015] have had a big impact on the segmentation literature, allowing architectures that effectively reverse the pooling operations of a traditional CNN and allow for outputs as large as the input or even larger. More recently it has been established in work on image super-resolution [Shi et al., 2016] and depth estimation [Laina et al., 2016] that the 2D up-sampling convolution operation is equivalent to a regular convolution with \( cr^2 \) feature channels where \( c \) is the original number of feature channels and \( r \) is the up-sampling ratio. The resulting \( cr^2 \) feature map can be efficiently reshuffled to have the same size as the target output. Effectively each channel contains information on surrounding subpixels in the spatial feature map. A similar idea has been applied to instance segmentation and detection [Dai et al., 2016a, Dai et al., 2016c] that associates certain features in each grid cell of the final convolutional layer with a higher resolution score map for each possible location. In our work we would like to adopt this approach for segmentation and investigate whether we can establish a similar relationship in 3D images between depth slices without explicitly representing our data in a 3D volumetric form.
3.1 Timeline for Completion

Table 3.1 contains a schedule for the completion of the tasks outlined in this proposal. We are targeting the March deadline for ICCV for our initial conference submission of this work. This submission should include a complete model for joint localization, segmentation, classification, and pose estimation as well as evaluation on the Google Street View and at least one other dataset. Realistically there is probably not enough time to implement and train models for all of the three proposed experimental directions for this deadline but ideally at least one of these will be part of the submission.

Optimistically we will be able to revisit the tasks of domain adaptation and missing point reconstruction for a follow-up paper that will be submitted to 3DV in Summer 2017. Otherwise we will continue work on our earlier proposed experiments for the 3DV submission and the dissertation.
<table>
<thead>
<tr>
<th>Date</th>
<th>Milestones</th>
</tr>
</thead>
<tbody>
<tr>
<td>January 2017</td>
<td>Complete implementation of baseline models and begin training models for evaluation. Implement joint localization, segmentation, classification, and pose estimation model.</td>
</tr>
<tr>
<td>February 2017</td>
<td>Experiment with architectures using spatial partitioning of viewing volume, spatial transformers, and sub-pixel shuffle techniques.</td>
</tr>
<tr>
<td>March 2017</td>
<td>Prepare paper for submission to ICCV 2017. Additional experiments on domain adaptation and missing point reconstruction.</td>
</tr>
<tr>
<td>May 2017</td>
<td>Dissertation defense.</td>
</tr>
</tbody>
</table>

Table 3.1: Schedule for completion of tasks.
Bibliography


