Inference Machines

Parsing Scenes via Iterated Predictions

Daniel Muñoz

CMU-RI-TR-13-15

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Robotics.

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

June 6, 2013

Thesis Committee
J. Andrew Bagnell, *Co-chair*
Martial Hebert, *Co-chair*
Takeo Kanade
Yann LeCun, *New York University*

To my family, my dog, and clowns

Abstract

Extracting a rich representation of an environment from visual sensor readings can benefit many tasks in robotics, *e.g.*, path planning, mapping, and object manipulation. While important progress has been made, it remains a difficult problem to effectively *parse* entire scenes, *i.e.*, to recognize semantic objects, man-made structures, and landforms. This process requires not only recognizing individual entities but also understanding the contextual relations among them.

The prevalent approach to encode such relationships is to use a joint probabilistic or energy-based model which enables one to naturally write down these interactions. Unfortunately, performing exact inference over these expressive models is often intractable and instead we can only approximate the solutions. While there exists a set of sophisticated approximate inference techniques to choose from, the combination of learning and approximate inference for these expressive models is still poorly understood in theory and limited in practice. Furthermore, using approximate inference on any learned model often leads to suboptimal predictions due to the inherent approximations.

As we ultimately care about predicting the correct labeling of a scene, and not necessarily learning a joint model of the data, this work proposes to instead view the approximate inference process as a modular procedure that is directly trained in order to produce a correct labeling of the scene. Inspired by early hierarchical models in the computer vision literature for scene parsing, the proposed inference procedure is structured to incorporate both feature descriptors and contextual cues computed at multiple resolutions within the scene. We demonstrate that this *inference machine* framework for parsing scenes via iterated predictions offers the best of both worlds: state-of-the-art classification accuracy and computational efficiency when processing images and/or unorganized 3-D point clouds. Additionally, we address critical problems that arise in practice when parsing scenes on board real-world systems: integrating data from multiple sensor modalities and efficiently processing data that is continuously streaming from the sensors.

Acknowledgements

I would like to thank Drew Bagnell, Martial Hebert, Takeo Kanade, and Yann LeCun, for serving on my committee. I appreciate the time they set aside for me and am thankful for the invaluable discussions and comments regarding this work.

I'm extremely happy to have spent the past n years working with Drew and Martial. Despite their *many* other commitments, both have been there whenever I've needed their help (and often to quickly point why what I was thinking should probably be avoided). At the same time, I'm thankful for the freedom they gave me to pursue collaborations on a variety of topics which have bettered myself as researcher. I've learned a tremendous amount from them during my journey, and I'm eternally grateful and in debt for their guidance, support, and patience with me along the way.

One of the perks of being co-advised is calling home to two lab groups. Thanks to the members of Drew's LAIR lab and Martial's VMR lab for teaching me about their research and sharing their diverse perspectives and expertise with me. Particularly, many discussions with Alex Grubb and Stéphane Ross were critical in the development of the ideas and methods used in this (and future) work.

Over the years I've been fortunate to have successful collaborations with multiple people: Elliot Cuzzillo, Debadeepta Dey, Alex Grubb, Hanzhang Hu, Ondra Miksik, Varun Ramakrishna, Nick Rhinehart, Stéphane Ross, Nicolas Vandapel, and Xuehan Xiong. Thank you for calling me out on my half-baked ideas and teaching me new tricks. Additionally, a big thanks to Brian and Sean Bittner for helping me with many experimental setups.

I could not have made it all the way through grad school without critical help along the way. In roughly chronological order:

- I am forever in debt to Siva "MF" Srinivasan for trying to teach me remedial computer science & discrete math concepts throughout undergrad – who knows where I would've ended up without you.

- I am thankful for the enthusiastic support from my undergrad advisor, Yongwon Lee. He helped me put things into perspective and instilled belief in me that I

could survive in the big leagues of grad school.

- Nicolas Vandapel was my unofficial MS advisor and taught me many things that I still closely hold on to (including all I know about 3-D point clouds). Furthermore, my early work (and consequently my admission into the PhD program) wouldn't have been possible without his insights and help over countless late nights – thank you.

- The first year of grad school was overwhelming, but it was easy to get through it with the support of great people: Brian Becker, Alvaro Collet, Santosh Divvala, Michael Furlong, and Uma Nagarajan.

- I am extremely grateful for early discussions with Nathan Ratliff on his excellent research. Those conversations definitively shaped the way that I think about problems – thank you for your time and patience with me.

- A huge thanks to Suzanne Muth for ensuring that I didn't fall through the cracks along the way.

Many thanks to my officemates for tolerating my incessant mutterings, entertaining my aloud musings (*i.e.*, rants) on research & the purpose of life, and (especially) celebrating FTS o'clock: Carl Doersch, Ed Hsiao, Tomasz Malisiewicz, Stéphane Ross. Scott Satkin, and Yuandong Tian. Also, a big thanks to Michael Dille for being a great housemate and sharing his savory dishes.

Lastly, but most importantly, getting to now would not have been possible without the unwaning support of my family and friends, particularly, DHQ. I have the fortunate problem of having many people playing important parts of my life, so I apologize for copping out with an insufficient, impersonal acknowledgement: thank *you* for being there.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Perception remains one of the depressingly difficult bottlenecks to the deployment of reliable autonomous systems. It has been consistently demonstrated that improving the representation of the robot's environment often results in improving the success rate of subsequent robotic tasks. For example, robustly recognizing landmarks from seemingly disparate viewpoints can help the robot to simultaneously localize in, and construct maps of, novel environments that span large areas (Cummins and Newman, 2011). Reliably identifying obstacles from long-range can help path planners avoid exploring places in the world that might be difficult to navigate through (Sofman et al., 2006). And consistently identifying different kinds of objects and their pose can help robotic manipulators to succeed in interacting with objects of interest without collision (Collet et al., 2011a). These examples highlight the importance of extracting a better representation, even if the representation is only a specific portion of the environment.

1.1 Problem

This work focuses on improving the representation of entire environments. Specifically, we address the problem of *scene parsing*, *i.e.*, assigning a semantic categorical label to all *sites*[1] in an observed scene, as illustrated in Figure 1.1. In our extensive experimental analysis, these labels include objects (*e.g.*, people, cars, animals, signs, *etc.*), man-made structures (*e.g.*, buildings, power lines, sidewalks, *etc.*), and landforms (*e.g.*, grass, mountains, water, *etc.*).

Scene parsing is a challenging problem because it is unclear how to best represent a scene in order to label it. For example, labeling each pixel in an image is problematic because many objects appear very similar at a local scale in the image. In contrast,

[1] We refer to a "site" as the most basic element in the domain to be labeled. In images, sites are typically pixels or superpixels, and sites are points or voxels in 3-D point clouds.

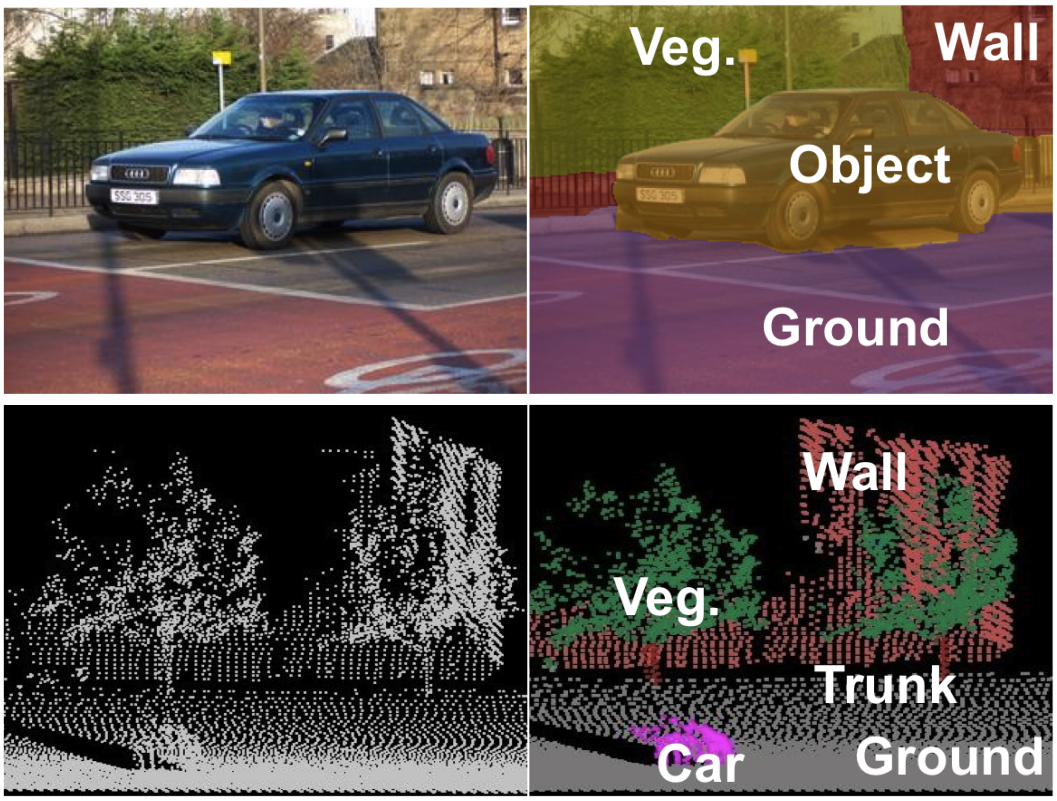


Figure 1.1: Two examples of scene parsing from images (top) and 3-D point clouds (bottom). In both examples, the input unlabeled data are on the left and parsings from this work are shown on the right.

in an ideal world, we would label the exact regions/segments of the things of interest. However, it is impossible to generate this perfect segmentation, and, in practice, we are instead left with imperfect regions that often contain multiple categories. In addition to representation, it is necessary for any approach to be holistically consistent, *e.g.*, cars should not be floating in the sky (yet). Learning models for scene parsing that effectively integrate these contextual relations remains difficult.

From a machine learning perspective, scene parsing can be considered as a structured prediction problem, in that we want to model the structure of the outputs in the prediction, *i.e.*, the interactions of the sites' label assignments. The two prevalent tools for structured prediction are graphical models, such as Conditional Random Fields (CRFs) (Lafferty et al., 2001, Kumar and Hebert, 2006), or unnormalized, energy-based models (Taskar et al., 2003, Tsochantaridis et al., 2005, LeCun et al., 2006). While indeed these models have been demonstrated to give large improvements over local classifiers for scene parsing (He et al., 2004, Kumar and Hebert, 2005, Anguelov et al., 2005), most are limited in the types of interactions they can model among the predictions.

In the recent computer vision literature, there is a trend of developing models with increasingly sophisticated interactions in order to achieve more accurate predictions, which is evident from recent award-winning papers (Ladicky et al., 2010, Gupta et al., 2010, Desai et al., 2011, Kuettel et al., 2012). However, it should be noted that even the simplest models over multiple labels are known to be NP-hard (Veksler, 1999), and many recent works are focused on developing *approximate* inference algorithms. Unfortunately, while graphical models provide a clean separation between modeling and inference, learning these models with approximate inference is not well understood (Wainwright, 2006, Kulesza and Pereira, 2007, Finley and Joachims, 2008). Furthermore, restricting our models to use approximate inference techniques, even with bounded guarantees on the solution, can lead to learning models that infer labelings with a better score than the *ground truth* labeling, which indicates a modeling mismatch of the problem (Szeliski et al., 2007).

1.2 Approach

We address the scene parsing problem from a machine learning approach, but we are also grounded with representations that are in spirit with the probabilistic models developed in the computer vision literature. However, we make the key distinction of avoiding learning a probabilistic model, which leads to my thesis statement.

> **Thesis statement**: As we ultimately care about predicting a correct labeling of the scene, we should directly train an inference procedure to do so. By exploiting useful intermediate representations of the scene, we can break down the inference procedure into a sequence of simple modules that we can effectively train to net a holistically consistent labeling.

At the heart of our approach, we consider approximate inference as a procedure: we can view an iterative inference algorithm, such as variational mean field on a random field (Wainwright and Jordan, 2008), as a network of computational modules taking in observations and other local computations on the graph (messages). We can then iteratively train each of these modules to output ideal intermediate messages, culminating in a holistically consistent parsing of the scene. This approach, which we refer to as an *inference machine*, eschews the theoretical and empirical difficulties of learning a global or probabilistic model of the data, and is heavily inspired from work in machine learning for sequence prediction (Cohen and Carvalho, 2005, Daume III et al., 2009). Furthermore, training the inference procedure builds off work in training deep, modular networks (Fahlman and Lebiere, 1990, LeCun et al., 1998, Bengio, 2009) with key distinctions. Firstly, we can effectively train each subproblem with what is the ideal

intermediate output. Secondly, this results in a more controlled overall learning problem because we can observe of how much progress is being made after each subproblem. Thirdly, the process is trained to follow the same procedure during test-time in order to avoid cascading of errors and overfitting between subproblems.

We demonstrate on a variety of datasets that this inference machine framework generates state-of-the-art scene parsings not only in terms of classification accuracy but also computationally efficiency. Additionally in this work, we address critical problems which are necessary for operating within real-world systems: integrating data from multiple sensor modalities, and processing both image and point cloud data that is streaming from the sensor.

1.3 Overview and Contributions

The next three chapters review background material which will be leveraged throughout the remainder of this thesis. In Chapter 2, we describe the various image and point cloud datasets we analyze in our experimental evaluations. In Chapter 3, we review image and point cloud preprocessing techniques, including feature computation and segmentation. In Chapter 4, we review statistical machine learning concepts, including classification, regression, and convex function(al) minimization. The remaining chapters detail the main contributions of this thesis:

- In Chapter 5, we revisit the canonical approach of parsing scenes using graphical models. This chapter presents an effective max-margin learning algorithm for these models with non-parametric clique potentials that improves upon previous methods. This work was published in (Munoz et al., 2009a) and also contributes a new annotated point cloud dataset to the community.

- In Chapter 6, we present the *Inference Machine* framework for parsing scenes from images and 3-D point clouds and demonstrate its state-of-the-art perfomance in both accuracy and efficiency. This work was published in (Munoz et al., 2010b, Xiong et al., 2011).

- In Chapter 7, we show how to adapt the inference machine framework to incorporate data from multiple modalities. This addresses the problem of parsing scenes from images and registered 3-D data in the general scenario when there is not a one-to-one correspondence between each pixel and point. This work was published in (Munoz et al., 2012) and also contributes a new annotated dataset of images with registered 3-D point clouds to the community.

- In Chapter 8, we show how to efficiently impose temporally consistent scene parses from streaming video by learning a distance metric that discriminatively propagates information between frames. This work was published in (Miksik et al., 2013)

- In Chapter 9, we show how to modify the representation in order to efficiently parse unorganized 3-D points that are streaming from the laser sensor. This work was published in (Hu et al., 2013)

Chapter 2

Datasets

This thesis analyzes classification and efficiency performances on a variety of different real-world (and publicly available) datasets. These datasets were collected using a diverse set of sensor modalities (*e.g.*, different optical cameras and laser scanners) in diverse environments (*e.g.*, natural, urban, and college campus) and are annotated with various classes. The datasets were chosen both to compare with prior work on scene parsing and because they were collected under real-world operating conditions for a mobile robot, *i.e.*, the data is not synthetic or collected under controlled conditions.

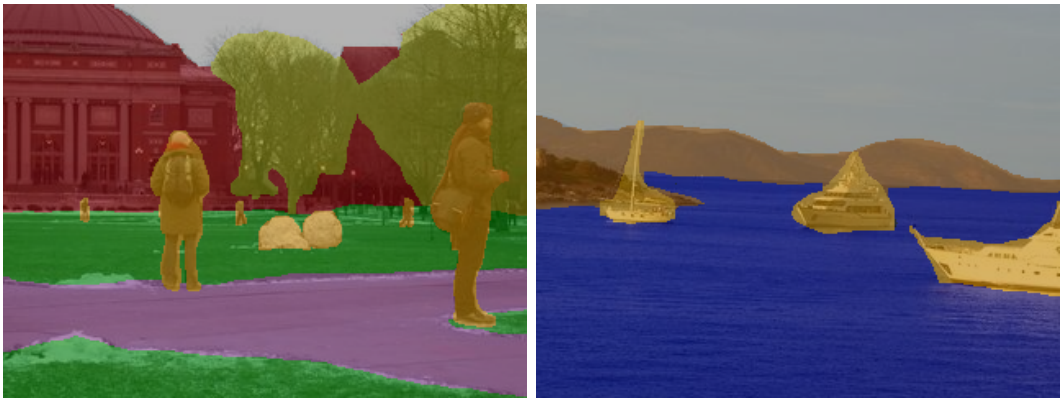
2.1 Image Datasets

2.1.1 Stanford Background

This dataset contains 572 images of a diverse set of outdoor environments (Gould et al., 2009). The vast majority of images are fully annotated into 8 classes, as illustrated in Figure 2.1, making it a moderately large and useful dataset to analyze scene parsing. Evaluation is averaged over 5 random folds of 572 training/143 testing images.

2.1.2 CamVid

This dataset consists of over 10 minutes of 30 Hz video captured around Cambridge, U.K. during both daylight and dusk (Brostow et al., 2008, 2009). A subset of the video is sparsely annotated in time at 1 Hz, resulting in an annotated dataset of 701 annotated images. Following the authors' original analysis, we evaluate predictions over 11 classes, as illustrated in Figure 2.2. The diverse set of annotated classes and real-world operating conditions make this a good dataset to evaluate scene parsing for mobile robotics. Static image evaluation is performed on one fold of 468 training/233 testing images.



Sky	Tree	Road	Grass	Water	Building	Mountain	Object
14.5	14.1	22.6	6.88	4.02	22.7	1.35	13.7

Figure 2.1: Examples from STANFORD BACKGROUND and its distribution of classes (%)

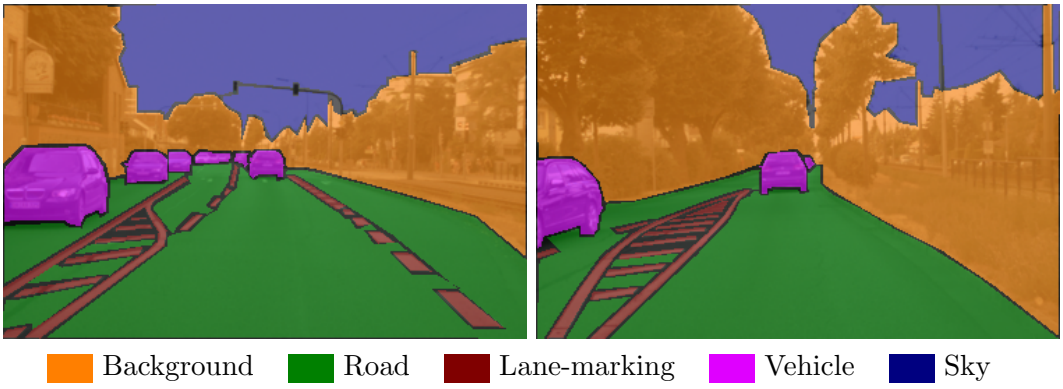
Build.	Tree	Sky	Car	Sign	Road	Ped.	Fence	Pole	Sidewalk	Bicyc.
29.9	9.0	12.8	2.88	1.07	31.5	0.78	1.59	1.03	8.67	0.70

Figure 2.2: Examples from CAMVID and its distribution of classes (%)

2.1.3 MPIVehicleScenes

This dataset consists of 156 annotated frames[1] captured from a dashboard-mounted camera while driving in a city (Wojek et al., 2010). The images are annotated into 5 classes, as illustrated in Figure 2.3.

[1]We use video sequence CONTINUOUS_2008.07.30_AT_13.10.53



Background	Road	Lane-marking	Vehicle	Sky
48.0	30.0	2.87	4.02	15.2

Figure 2.3: Examples from MPIVehicleScenes and its distribution of classes (%)

Building	Car	Door	Person	Plant	Pole	Road
14.1	21.6	1.26	1.14	0.39	0.47	27.0

Sidewalk	Sign	Sky	Staircase	Tree	Window
1.74	0.12	0.37	1.50	28.9	1.51

Figure 2.4: Examples from NYU Scenes and its distribution of classes (%)

2.1.4 NYU Scenes

This dataset consists of 74 annotated frames from a video of a person walking with a hand-held camera in an urban street in New York City; it was provided by Clement Farabet. In addition to the moving objects, the video has a lot of camera motion due to the motion of the person, making it a challenging dataset to perform well on. The

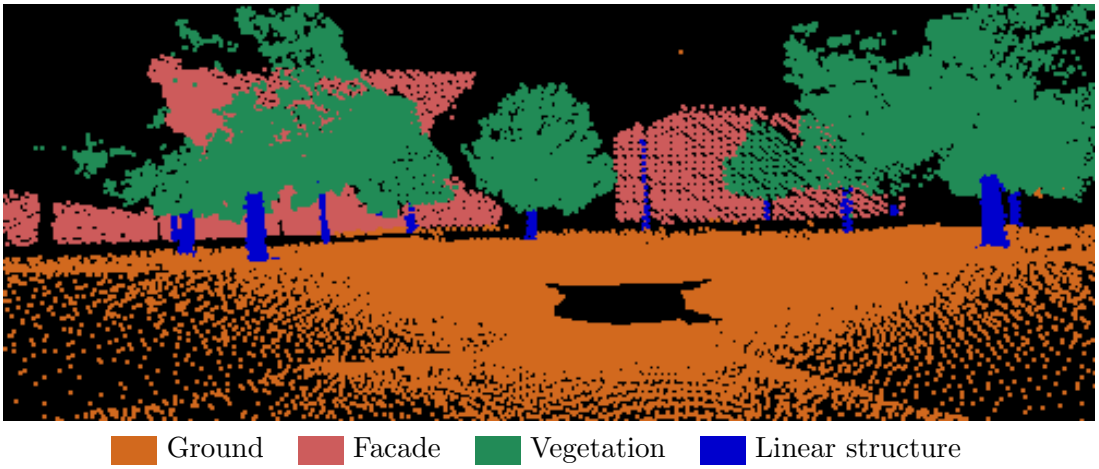


Figure 2.5: An annotated point cloud from the FREIBURG dataset

images are annotated into the same 33 classes from the SIFT FLOW dataset (Liu et al., 2011), as illustrated in Figure 2.4.

2.2 3-D Point Cloud Datasets

2.2.1 Freiburg

This point cloud dataset was captured using a pan-tilting SICK LMS laser scanner around the University of Freiburg (Steder et al., 2010). Following (Behley et al., 2012), we evaluate predictions over 4 classes, as illustrated in Figure 2.5. The large spatial extent of the data in a college campus environment makes it a unique dataset. Evaluation is averaged over 5 folds of 4 training/1 testing scans, where each scan contains $170,000$ 3-D points.

2.2.2 GML-PCV

This point cloud dataset was captured using an airborne laser scanner covering large outdoor areas (Shapovalov et al., 2010). The dataset is annotated into 5 classes, as illustrated in Figure 2.6. Evaluation is performed performed over two partitions of 1 training/1 testing scan, each of which contains 1 M points.

2.2.3 VMR Oakland

We collected and annotated this dataset using a push-broom SICK LMS laser scanner around Carnegie Mellon University (Munoz et al., 2009a). In VMR OAKLAND-V2, the points are annotated into 7 classes illustrated in Figure 2.7. In VMR OAKLAND-V1, the

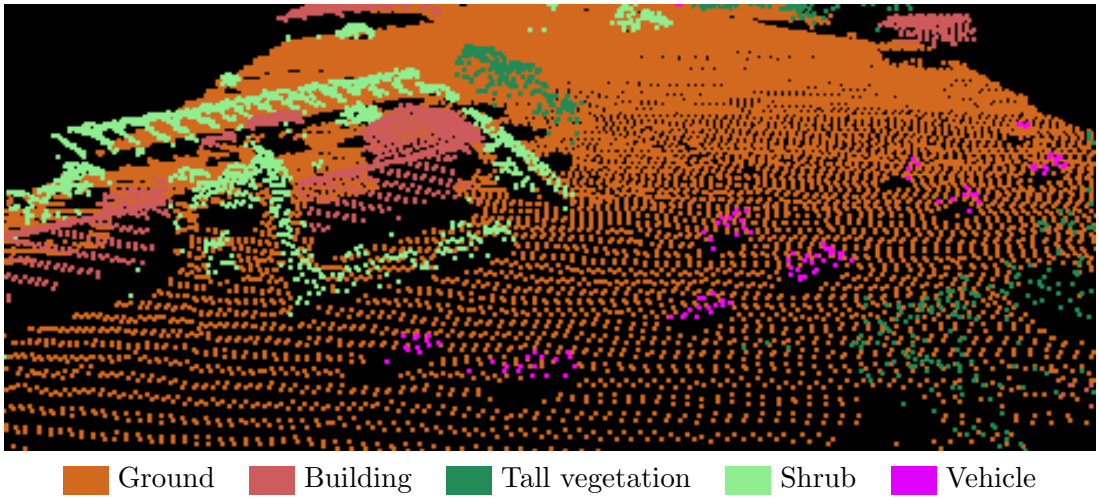


Figure 2.6: An annotated point cloud from the GML-PCV dataset

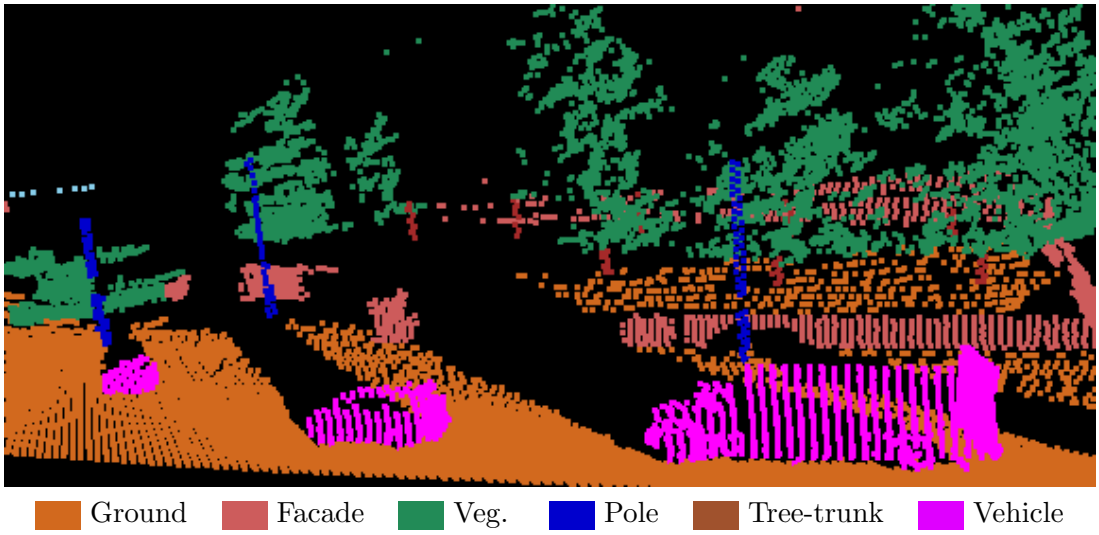


Figure 2.7: An annotated point cloud from the VMR Oakland-v2 dataset

Pole and Tree-trunk classes are collapsed into a Linear-structure class and the Vehicle class is removed, resulting in a 5-class dataset. Its diverse set of annotated classes and spatial extent make it one of the largest, laser-based 3-D point cloud datasets. For VMR Oakland-v1, the training set is a single scan of $37,000$ points and tested on 15 scans, totaling of 1.3 M points overall. For VMR Oakland-v2, the evaluation is averaged over 6 folds of 6 training/24 testing scans.

2.3 Registered Image and 3-D Point Cloud Datasets

2.3.1 CMU Image+Laser

We collected and annotated this dataset of 372 scenes (images with registered 3-D point clouds) obtained from a vehicle driving around Carnegie Mellon University (Munoz et al., 2012). The camera used a wide-angle lens and the laser scanner operated in push-broom mode. Hence, the displacement is often on the order of tens of meters between the location of the laser when it scanned a point vs. the location of the camera when it observes that point. The images were annotated using LabelMe (Russell et al., 2007) into 29 classes, as illustrated in Figure 2.8. The 3-D points are mapped into a global reference frame and then registered to corresponding images. On average, 31,000 3-D points project into an image. The 3-D annotations are obtained by back-projecting these 2-D annotations. Hence, the 3-D annotations are susceptible to subtle projection errors when objects are transparent/porous and/or have a high incident angle with the camera. Since the laser scans in push-broom mode, there exist many scenes containing 3-D scan lines that do not cover the image due to when the vehicle moves slowly/stops. Evaluation is performed on 5 different partitions (297 training/75 testing) of the data, grouped by time, which is needed to avoid testing on scenes that might overlap with the training data.

2.4 Evaluation

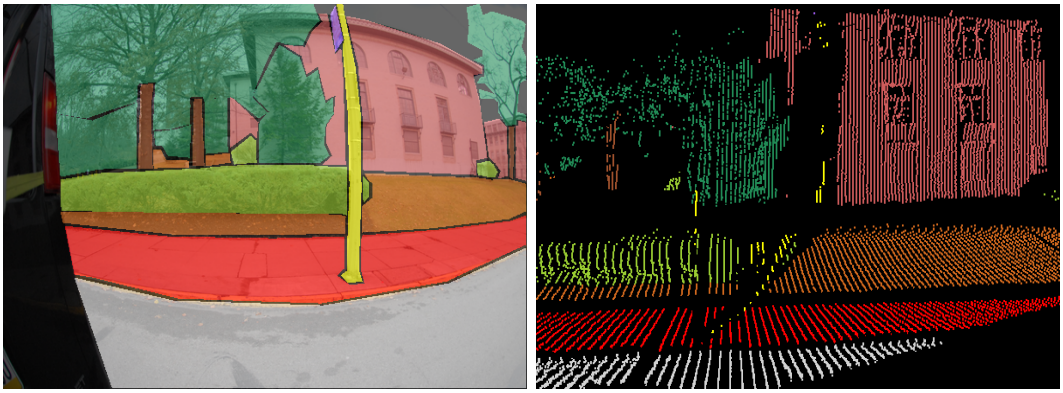
In both image and point cloud datasets, performance is quantitatively evaluated on the classification of pixels or 3-D points/voxels. We report the k'th class' precision p_k and recall r_k values,

$$p_k = \frac{t_k}{i_k}, \tag{2.1}$$

$$r_k = \frac{t_k}{n_k}, \tag{2.2}$$

where t_k is the number of pixels/points correctly classified for the k'th class, i_k is the number of pixels/points inferred/predicted as the k'th class, and n_k is the number annotated of pixels/points from the k'th class. Hence, r_k is equivalent to the per-class accuracy. To summarize the tradeoff between precision and recall values, we report each class' F_1 score, defined as

$$\frac{2p_k r_k}{p_k + r_k}. \tag{2.3}$$



	Road	Sidewalk	Ground	Building	Barrier	Bus-stop
2-D	27.65	12.66	5.99	17.48	3.25	0.11
3-D	10.79	8.01	8.88	27.06	2.54	0.21
	Shrub	Tree-trunk	Tree-top	Small-veh.	Big-veh.	Bike
2-D	2.46	0.79	17.89	7.22	1.78	0.03
3-D	4.42	1.37	26.51	5.41	1.55	0.04
	Flag-pole	Tall-light	Short-light	Post	Sign	Util.-pole
2-D	0.01	0.17	0.03	0.27	0.24	0.22
3-D	0.04	0.31	0.10	0.43	0.36	0.26
	Traffic-pole	Traffic-signal	Bag	Trash	Hydrant	Mailbox
2-D	0.09	0.04	0.03	0.04	0.01	0.01
3-D	0.14	0.06	0.03	0.06	0.01	0.02
	Bench	Stairs	Person	Wire	Obstacle	
2-D	0.02	0.19	0.62	0.57	0.13	
3-D	0.03	0.44	0.72	0.10	0.10	

Figure 2.8: Examples from CMU IMAGE+LASER and its distribution of classes (%)

Finally, we consider two metrics to summarize the performance for a particular dataset. First is the overall per point/pixel accuracy,

$$\frac{\sum_k t_k}{\sum_k n_k}. \tag{2.4}$$

Achieving a high overall accuracy can be misleading due to the often severe imbalance in the proportions of each class in the dataset, *e.g.*, buildings and the ground plane often dominant a large proportion of the scene. Hence, we also report the average/macro F_1 score, which is the unweighted averaged of the per-class F_1 scores, to summarize how well an algorithm performs across all classes.

Chapter 3

Computer Vision Tools

In order to process either image or point cloud data, it is necessary to compute a compressed representation of the raw input signal that is consistent across different observations, *e.g.*, images of different resolutions. This chapter describes these representations that are used in the experiments in the later chapters. In Section 3.1, we discuss low-level feature reprsentations computed locally in the input signal. In Section 3.2, we discuss how the signal can be partitioned/segmented into a compressed representation. In Section 3.3, we discuss higher-level descriptors that can be computed over these segments.

3.1 Low-level Features

3.1.1 Images

A basic pixel descriptor for image processing is to use each pixel's color channel values, *e.g.*, red, green, and blue. Additionally, it can be helpful to consider non-linear transformations of RGB into additional colorspaces, such as HSV and CIELAB.

In order to capture edges/texture in the image, we can convolve the image with various spatial filters, as illustrated in Figure 3.1a. In order to account for variations in appearance, we convolve multiple scaled versions of the filters. We refer to the pixel descriptor that is the formed by aggregating the responses from the filters at the pixel as TXT.

A related descriptor that encodes relative order of discontinuities is a local binary pattern (LBP), as illustrated in Figure 3.1b. Centering a $w \times w$ window at a pixel, a binary desciptor of length $w^2 - 1$ is formed by comparing the intensity of the center pixel to every other pixel in an ordered manner, *e.g.*, row-major order.

A more expressive descriptor that captures image gradient statistics over larger neighborhoods is the Scale Invariant Feature Transform (SIFT) descriptor (Lowe, 2004), as

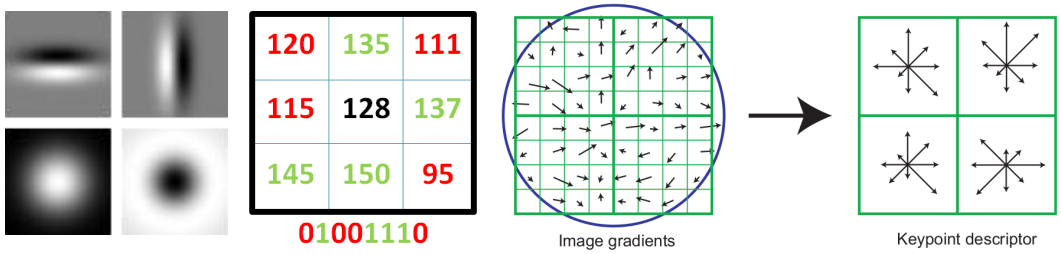


Figure 3.1: (a) Spatial filters: Derivative of Gaussians at horizontal and vertical orientations (top), Gaussian (bottom-left), Laplace of Gaussian (bottom-right). (b) A 3×3 Local Binary Pattern is formed by comparing the center pixel intensity (128) with all other intensities. (c) A SIFT descriptor is formed by quantizing the gradients into prespecified angular bins (8) over cells/windows (4 cells each containing 16 pixels); image reproduced from (Lowe, 2004).

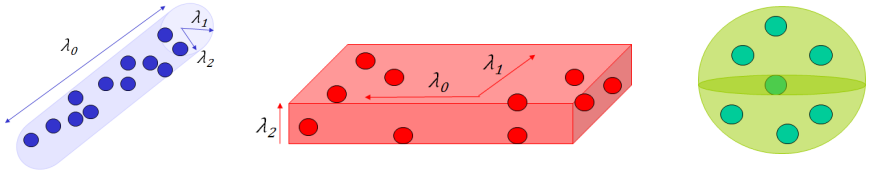
illustrated in Figure 3.1c. To construct this descriptor, pixel gradients (orientation and energy) are first computed. At each pixel, the gradient orientations are then quantized into angular bins. Finally, these angular bins are pooled/averaged over spatial cells/windows in the image. Hence, the pixel descriptor is formed by concatenating each cell's angular histogram. This descriptor can be computed over intensity images (I-SIFT) or over each channel in a colorspace (C-SIFT).

We use the publicly available Automatic Labeling Environment software library (Ladicky, 2011) to compute these low-level features in our experiments.

3.1.2 3-D Point Clouds

Similar to images, when processing 3-D points it is necessary to consider statistics computed over neighborhoods of the data. This thesis focuses on processing unorganized 3-D point clouds, *i.e.*, the points do not lie on a regular grid/lattice structure. Hence, these neighborhoods are defined in terms of absolute measurements (*e.g.*, meters) instead of an underlying representation (*e.g.*, an 8 pixel neighborhood in an image).

A powerful descriptor to quantify the local shape/geometry of a point is to use a local neighborhood of 3-D points and tensor voting (Medioni et al., 2000, Lalonde et al., 2007). Let $p \in \mathbb{R}^3$ be a 3-D point, $\mathcal{N}_p$ be a set of its neighboring 3-D points (inclusive),



(a) (b) (c)

Figure 3.2: Tensor Voting. The eigenvalues of the covariance matrix formed by the points in $\mathcal{N}_p$ describe three basic linear structures: (a) linear, (b) planar, (c) scattered.

Σ_p be the covariance matrix constructed from $\mathcal{N}_p$,

$$\Sigma_p = \frac{1}{|\mathcal{N}_p|} \sum_{q \in \mathcal{N}_p} (q - \mu_p)(q - \mu_p)^T, \tag{3.1}$$

$$\mu_p = \frac{1}{|\mathcal{N}_p|} \sum_{q \in \mathcal{N}_p} q, \tag{3.2}$$

$\lambda_0 > \lambda_1 > \lambda_2$ be the eigenvalues of Σ_p, and v_0, v_1, v_2 be its respective eigenvectors. When $\mathcal{N}_p$ represents a linear structure, *e.g.*, street poles, then $\lambda_0 >> \lambda_1, \lambda_2$ (Figure 3.2a). When $\mathcal{N}_p$ represents a planar structure, *e.g.*, building facades, then $\lambda_0, \lambda_1 >> \lambda_2$ (Figure 3.2b). When $\mathcal{N}_p$ represents a spherical/scattered structure, *e.g.*, vegetation, then $\lambda_0 \approx \lambda_1 \approx \lambda_2$ (Figure 3.2c). Hence, the three quantities $\lambda_0 - \lambda_1, \lambda_1 - \lambda_0, \lambda_2$ encode the saliency of these three geometric structures (GEOM).

Additionally, from the eigendecomposition of Σ_p, v_0 indicates the principal direction of the linear structure and v_2 is the normal of the plane; both directions are up to a sign ambiguity. Assuming the $+z$ axis extends towards the sky, then local orientation (ORIENT) can be encoded with the values $|v_0^T \mathtt{e}_z|$ and $|v_2^T \mathtt{e}_z|$, where $\mathtt{e}_z = [0, 0, 1]^T$ is a standard basis vector.

As with the SIFT descriptor in images, a more expressive 3-D descriptor can be constructed by analyzing the spatial distribution of points. The SPINIMAGE descriptor (Johnson and Hebert, 1999) compresses the relative 3-D locations of points into a 2-D representation/image. Conceptually, a cell image with cell sizes $r_\beta \times r_\alpha$ m^2 is centered at a point and then revolved/spun around an axis v_β, and the respective cell is incremented for every point that it collides with. Formally, letting $v_{pq} = q - p$ be the vector from the centered point p to neighbor point $q \in \mathcal{N}_p$, the cell with signed row coordinate $\left\lfloor \frac{v_{pq}^T v_\beta}{r_\beta} \right\rfloor$ and column coordinate $\left\lfloor \frac{\|v_{pq} \times v_\beta\|_2}{r_\alpha} \right\rfloor$ is incremented.

Finally, gravity is an informative cue which can be exploited for recognizing objects. For example, shrubs, by definition, are close to the ground, and power lines overhang from above. We can model relative elevation using the position of the sensor for each

point; however, this information may not be available for particular datasets. Instead, to be consistent across all evaluations, we approximate local elevation by constructing a 2.5-D elevation grid which records the lowest and highest point inside each cell. The point's descriptor is then the difference in min. and max. elevation for the respective cell it falls into (ELEV).

3.2 Segmentation

Segmenting an image or 3-D point cloud can be considered as a clustering problem for which a variety of techniques can be applied. K-means (LLoyd, 1982) is a canonical algorithm for partitioning the data into K clusters, each of which with members that have minimal distance to the cluster's mean. In images, a simple segmentation can be done by clustering over a descriptor that includes a pixel's color value and location in the image and using a metric that scales the relative importance of these values. In point clouds, we can simply cluster over the 3-D coordinates.

One drawback of K-means is the requirement to pre-specify the number of desired clusters, which we do not know in advance. Alternatively, we can use other techniques that *implicitly* parameterize the number of resulting clusters based on the distribution of the data. For its simplicity, efficiency, and applicability to both image and point cloud data, we use the graph-based segmentation algorithm from (Felzenszwalb and Huttenlocher, 2004) (F-H). Briefly, this graph-based approach operates similarly to Kruskal's algorithm for finding a minimum spanning tree in a graph. The input to the algorithm is an undirected graph $G = (V, E)$ with vertices V and edges $(ij) \in E$ that are associated with edge weights w_{ij} whose values indicate dissimilarity between vertices, *i.e.*, if vertices v_i and v_j are exactly the same then $w_{ij} = 0$. The algorithm is initialized with each vertex as its own component (cluster/segment). Iterating over the weighted edges $\{w_{ij}\}$ in ascending order, two components separately containing v_i and v_j are merged if

$$w_{ij} \leq \min\left(c(v_i) + \tau(v_i), c(v_j) + \tau(v_j)\right), \tag{3.3}$$

where $c(v)$ is the value of the largest edge weight in the minimum spanning tree of the component that contains vertex v, and τ is a threshold function that controls when differing components are merged. We use the commonly used threshold function of

$$\tau_t(v) = \frac{t}{s(v)}, \tag{3.4}$$

where $s(v)$ returns the size (*i.e.*, number of vertices) of the component that contains vertex v and t is a pre-specified parameter. Hence, Equation (3.3) becomes harder to satisfy as the components grow in size, causing Equation (3.4) to shrink, and the edge weights are iterated in ascending order.

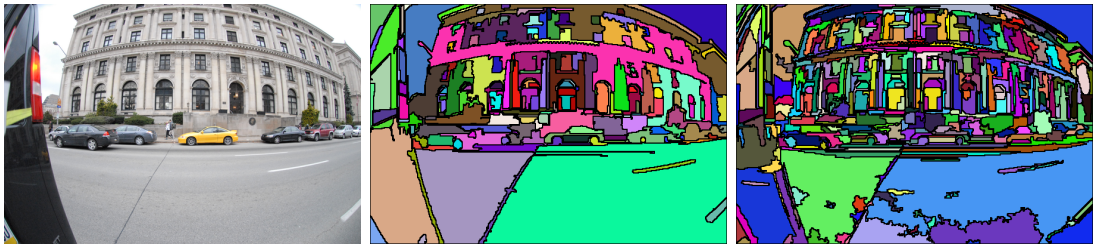


Figure 3.3: Example hierarchical segmentation of an image from the CMU IMAGE+LASER dataset which was constructed by using F-H and varying t in Equation (3.4).

In this thesis, we use the F-H algorithm both in images and point clouds. In images, the graph is constructed over the pixels as vertices with an 8-pixel neighborhood connectivity; an edge weight is the Euclidean distance between two pixels' RGB values. In point clouds, the graph is constructed over points/voxels as vertices and linking to neighboring vertices within a 1.0 m radius; an edge weight is the Euclidean distance between two points' concatenated GEOM and ORIENT features.

In practice, the segmentation parameters (*e.g.*, K in K-means, kernel bandwidth in mean shift, and t in F-H) that work well for one scene often do not generalize as expected in other scenes. To account for this, it is common to vary the segmentation parameters to generate multiple segmentation hypotheses, or a *hierarchy* of segmentations, to reason over, as illustrated in Figure 3.3. While varying the parameters using the same input will not necessarily generate a true hierarchy of a unique parent-to-child relationship, if necessary, a true hierarchy can be achieved by running the next segmentation using the previous segmentation as input. Chapter 5 and Chapter 6 study how to effectively integrate the multiple segmentations to improve classification.

3.3 Region Features

The regions resulting from a segmentation can provide better spatial support to compute richer feature representations. In both images and point clouds, the shape/geometry of the regions can be an informative cue.

In images, we compute the following region descriptor, as similarly used in (Gould et al., 2008b), which we refer to as RSHAPE2D:

- The region area and its ratio to the entire image area and the 25th, 50th, 75th percentile region areas.

- The ratio of the region's perimeter to both its and the entire image's areas.

- The regions variance along the rows, columns, and cross-covariance.
- The relative row and column offsets of the region's center with respect to the image center, normalized into $[-1, 1]$.

In 3-D point clouds, we compute similar a descriptor to encode the shape of the 3-D region (RSHAPE3D):

- The region's length along the z-axis ("height").
- The three lengths of the bounding box that encloses the region along its principal components and their ratios relative to each other and "height".
- The diagonal length of the 2-D bounding box that encloses the projection of the region's points into the x-y plane ("2-D span").
- The ratio of the "2-D span" to "height".
- The product of the "2-D span" and "height" ("volume").
- The number of points in the region and its ratio to the "volume".

Additionally, we adapt two of the low-level features (Section 3.1.2) to be used over regions: 1) in the GEOM descriptor, we use the region's points to define the neighborhood $\mathcal{N}$, and 2) in the ELEV descriptor, we compute min. and max. differences using the region's highest, lowest, and medoid points, resulting in a descriptor length of 6.

3.3.1 Pooled Statistics

In addition to computing explicit statistics over the shape of the region, a powerful feature representation is to look at the variation of the low-level features aggregated inside the region. For example, let $\mathcal{X}_r$ be the set of low-level feature descriptors $x_i \in \mathbb{R}^d$ for each site, *i.e.*, pixel or point, inside region r. Then, a region descriptor $x_r \in \mathbb{R}^d$ can be formed by computing empirical moments (*e.g.*, mean, variance, skewness, kurtosis) for each component/dimension (RMOMENTS).

Instead of using statistics computed over the original feature space, it can be better to use sparser representations of the low-level features that are pooled over region. A common technique to achieve this sparse representation is through vector quantization, where a sparse representation $z_i \in \mathbb{R}^m$, referred to as a *code*, with few non-zero values is computed from the original representation $x_i \in \mathbb{R}^d$. In computer vision, performing vector quantization using K-means has demonstrated to perform well for classification-based tasks (Leung and Malik, 2001). Briefly, K-means is run over a large and diverse dataset of features, *e.g.*, the low-level features in Section 3.1, to obtain a set of m exemplar cluster

centers $\{\mu_k\}_{k=1}^m$ which can be thought of as words/elements in a dictionary/codebook. Letting

$$d_k(x) = \|x - \mu_k\|_2^2, \tag{3.5}$$

denote the squared Euclidean distance to dictionary element μ_k, a sparse code $z \in \mathbb{R}^m$ of one non-zero value can be constructed using the coding rule

$$z[k] = \mathbb{1}(k = \arg\min_j d_j(x)), \quad \forall k = 1, \ldots, m, \tag{3.6}$$

where $\mathbb{1}(\cdot)$ is the indicator function. A region can be described as a sum of its parts by either averaging or max pooling over the sparse codes inside the region (RCodes).

Instead of constructing the sparse code z through a single assignment to the closest center, it has been shown that pooling over softer codes, generated from multiple assignments, can increase classification performance, even when compared to deep networks used for feature learning (Coates et al., 2011). These soft codes are constructed using the coding rule

$$z[k] = \max\left(0, \frac{1}{m}\sum_{j=1}^m d_j(x) - d_k(x)\right), \quad \forall k = 1, \ldots, m, \tag{3.7}$$

In words, components are non-zero for the respective dictionary elements μ_k that the original descriptor x is closer to vs. the average squared distance to all elements in the dictionary. Note that values of each component in the code can be computed independently by expanding distance and regrouping,

$$z[k] = \max\left(0, \frac{1}{m}\sum_{j=1}^m d_j(x) - d_k(x)\right) \tag{3.8}$$

$$= \max\left(0, \frac{1}{m}\sum_{j=1}^m (\|x\|_2^2 - 2x^T\mu_j + \|\mu_j\|_2^2) - (\|x\|_2^2 - 2x^T\mu_k + \|\mu_k\|_2^2)\right) \tag{3.9}$$

$$= \max\left(0, \frac{1}{m}\sum_{j=1}^m \|\mu_j\|_2^2 - \frac{2}{m}\sum_{j=1}^m x^T\mu_j + 2x^T\mu_k - \|\mu_k\|_2^2)\right) \tag{3.10}$$

$$= \max\left(0, x^T(2\mu_k - \frac{2}{m}\sum_{j=1}^m \mu_j) + \frac{1}{m}\sum_{j=1}^m \|\mu_j\|_2^2 - \|\mu_k\|_2^2\right) \tag{3.11}$$

Furthermore, note that many terms do not depend on the argument and can be precomputed as constants. Hence, the computations for each component are independent from each other and can be done with by thresholding a single inner product with zero.

Finally, we note that there exist other techniques to feature quantization including auto-encoders (Vincent et al., 2010), sparse coding (Mairal et al., 2012), K-SVD

(Aharon et al., 2006), and deep networks (Farabet et al., 2013). In our experiments, we use a variant of the K-means quantization where we define the dictionary elements simply using the K-means++ initialization procedure (Arthur and Vassilvitskii, 2007). Conceptually, this procedure can be thought of as defining the dictionary to be a diverse set of exemplar prototypes that maximally covers the set of training samples in feature space.

3.4 Normalization

Many clustering and machine learning algorithms are sensitive to the scale of the feature values and it is necessary to first standardize the features before processing. A common normalization technique is *whitening* that decorrelates each feature component. That is, letting X denote a zero-mean random variable for the feature vector, whitening finds a linear transformation W such that $\mathbb{E}[WX(WX)^T] = I$, where I is the identity matrix. Hence, the whitening transformation W is obtained by eigendecomposing the covariance matrix $\Sigma = V\Lambda V^T$ and defining $W = V\,\bar{\sqrt{\Lambda}}V^T$.

Note that a naive application of whitening the low-level features before vector quantization can be computationally expensive in practice when d is large due to the multiplication with a $d \times d$ matrix. However, this quadratic cost can be reduced to an inner product by noting that the squared norm of the transformed features $\|Wx\|_2^2$ is constant across distance computations to all dictionary elements. For example, in the single assignment case,

$$\arg\min_j d_j(Wx) = \arg\min_j \|Wx - \mu_j\|_2^2 \tag{3.12}$$

$$= \arg\min_j \|Wx\|_2^2 - 2(Wx)^T\mu_j + \|\mu_j\|_2^2 \tag{3.13}$$

$$= \arg\min_j x^T\tilde{\mu_j} + \|\mu_j\|_2^2, \tag{3.14}$$

where $\tilde{\mu_j} = -2W^T\mu_j$. This observation holds in the soft assignment cases and vector quantization with whitening still costs $O(dm)$ time to compute the entire sparse code.

Chapter 4

Machine Learning Tools

This work leverages several machine learning concepts which are used and built upon throughout the thesis. At a high level, we will predict classes using the feature representations described in the previous chapter. As there are a variety of prediction techniques to choose from, in this chapter we review these commonly used supervised learning techniques in isolation. In the remaining chapters we discuss how they are pieced together to form a coherent scene parser.

4.1 Regression

A common supervised learning task is to use an annotated training set to fit a function $h : \mathbb{R}^d \to \mathbb{R}^m$ that regresses numerical values $y \in \mathbb{R}^m$ from a feature representation of the data $x \in \mathcal{X}$. When classifying a 3-D point cloud over a set of 3 possible classes, $\mathcal{K} = \{1, 2, 3\}$, y could be a binary indicator/basis vector of length $m = 3$ with a value of 1 in the component of the sample's respective class index and 0 elsewhere, *e.g.*, $y_i = \mathsf{e}_k$ if the i'th sample belongs to the k'th class.

4.1.1 Linear Regression

If h is a linear model parameterized by $\theta \in R^d$, we can define the score for assigning sample x to the k'th class as $\theta^T \phi(x, k)$, where $\phi : \mathcal{X} \times \mathcal{K} \to \mathbb{R}^d$ is a feature function that computes a (potentially label-specific) feature representation for the sample, *e.g.*, the Geom descriptor from Section 3.1.2. Finding the parameters can be formulated as minimizing the regularized squared error

$$\min_{\theta} \lambda \Omega(\theta) + \sum_i w_i \left\| y_i - \theta^T \phi(x_i, k_i) \right\|_2^2, \tag{4.1}$$

where $\Omega : \mathbb{R}^d \rightarrow \mathbb{R}$ is a regularizer function that measures some notion of model complexity, $w_i \in \mathbb{R}^+$ weights the error of the i'th sample's fit, and λ trades off between the model's complexity and error. When the regularizer decouples across the model parameters, *e.g.*, the squared L^2 norm, $\Omega(\theta) = \|\theta\|_2^2$, the parameters for each class can be solved for independently (and in closed form). Alternatively, one can obtain a model that better generalizes to new data by coupling/sharing the parameters across classes, *e.g.*, a rank constraint or trace norm if the parameters θ were interpretted in matrix form (Srebro and Shraibman, 2005).

4.1.2 Tree Regression

In practice, more accurate predictions can often be achieved using a non-linear model h, *e.g.*, kernel regression (Nadaraya, 1964), neural networks (Werbos, 1974), and regression trees/forests (Breiman, 2001). Due to its demonstrated effectiveness with computer vision representations (Criminisi and Shotton, 2013), we will use the regression forest framework throughout this thesis. Briefly, given an annotated dataset[1], $\mathcal{D} = (X, Y) = \{(x_i, y_i)\}_{i=1}^n$, a regression tree is constructed by recursively partitioning the data into non-overlapping two left and right subsets (children nodes), $\mathcal{D}_L = (X_L, Y_L), \mathcal{D}_R = (X_R, Y_R)$, respectively, with targets that maximally reduce the variance from the input set (parent node). Formally, letting Σ_Z be the covariance matrix of the targets $Y \in \mathcal{D}_Z$, we split the current parent node $\mathcal{D}_P$ into two using the objective

$$\begin{aligned} \max_{\mathcal{D}_L, \mathcal{D}_R} \quad & |\mathcal{D}_P| J(\Sigma_P) - |\mathcal{D}_L| J(\Sigma_L) - |\mathcal{D}_R| J(\Sigma_R) \\ \text{s.t.} \quad & \mathcal{D}_L \cup \mathcal{D}_R = \mathcal{D}_P, \\ & \mathcal{D}_L \cap \mathcal{D}_R = \emptyset, \end{aligned} \tag{4.2}$$

where $J(\Sigma) = \text{Tr}(\Sigma)$ measures the sum of the output components' variance[2]. The data can be partitioned by 1) exhaustively iterating over each feature component in x, 2) partitioning samples with values less than or greater than a hypothesis threshold, 3) picking the threshold (from an exhaustive set) that maximizes Equation (4.2).

The tree is grown until some criterion is met, *e.g.*, a maximum tree depth, performance on a validation set, and/or a minimum number of samples a node needs in order to further partition the data. At prediction time, a descriptor follows the sequence of threshold comparisons down the tree and outputs the average targets from the training samples that fell into the respective leaf node. In practice, it has been observed that an exhaustive search for feature splits is prone to overfitting (Breiman, 2001). To better

[1] In this context, $x_i \in R^d$ is the feature representation for the sample that is shared for all the classes.

[2] Alternative measures for variance can be used such as $J(\Sigma) = \log(|\Sigma|)$. We consider the trace because it is efficient to evaluate.

generalize the predictions, a *forest* of trees can be grown simultaneously where at each node, only a small *random* subset of features are considered to split over. At prediction time, the average of each tree's output from the random forest is used.

4.2 Classification

Another canonical supervised learning task is to learn a discriminative function that explicitly classifies the data into a predefined set of labels/categories, often with a notion of confidence in the prediction.

4.2.1 SVM Predictor

A standard classification tool is the Support Vector Machine (SVM) (Cortes and Vapnik, 1995) which attempts to score the true label over other labels with maximum margin. This can be formulated as the program

$$\begin{aligned} \max_{\theta,\gamma,\xi} \quad & \frac{\lambda}{2} m - \sum_i w_i \xi_i \\ \text{s.t.} \quad & \theta^T \phi(x_i, k_i) \geq \max_{k \in \mathcal{K} \setminus k_i} \theta^T \phi(x_i, k) + \gamma - \xi_i, \quad \forall i \\ & \xi_i \geq 0, \quad \forall i \\ & \|\theta\| = 1, \end{aligned} \tag{4.3}$$

where ξ are slack variables that relax the margin constraints in the case the data is not linearly separable, w_i weights the cost of the i'th sample violating the margin (this is referred to as slack-scaling), and λ trades off between maximizing the margin γ and satisfying the constraints. Program (4.3) can be equivalently written as the convex program

$$\begin{aligned} \min_{\theta,\xi} \quad & \frac{\lambda}{2} \|\theta\|_2^2 + \sum_i w_i \xi_i \\ \text{s.t.} \quad & \theta^T \phi(x_i, k_i) \geq \max_{k \in \mathcal{K} \setminus k_i} \theta^T \phi(x_i, k) + 1 - \xi_i, \quad \forall i \\ & \xi_i \geq 0, \quad \forall i. \end{aligned} \tag{4.4}$$

And, Program (4.4) can be equivalently written as the unconstrained minimization

$$\min_{\theta} \frac{\lambda}{2} \|\theta\|_2^2 + \sum_i w_i \max(0, \max_{k \in \mathcal{K} \setminus k_i} \theta^T \phi(x_i, k) + 1 - \theta^T \phi(x_i, k_i)). \tag{4.5}$$

Each term in the summation of Equation (4.5) is often referred to as the (weighted) *hinge loss* because it is piecewise linear with a "hinge" at the point of non-differentiability.

Instead of weighting/scaling the slack of violating the i'th constraint by w_i, another way to soften/strengthen the constraint is to scale the margin value 1 to a smaller/bigger value; this is referred to as margin-scaling.

4.2.2 MaxEnt Predictor

While the max-margin SVM approach provides a notion of confidence via the margin/score of a class relative to others, it is often useful to have a probabilistic interpretation. The principle of maximum entropy (Jaynes, 1957) can be used to find a conditional probability distribution, $P(Y|X)$, that is least committed to predicting any particular class Y, *i.e.*, the distribution has maximal conditional entropy, $H(Y|X) = -\sum_{x,y} P(x,y) \log P(y|x))$, while matching the feature statistics of the data X. Finding this distribution can be formulated as the concave program

$$\max_{P(Y|X)} \quad H(Y|X) \tag{4.6}$$

$$\begin{aligned} \text{s.t.} \quad & \mathbb{E}_{P(X,Y)}[\phi_d(X,Y)] = \mathbb{E}_{\tilde{P}(X,Y)}[\phi_d(X,Y)], \quad \forall d, \\ & \sum_y P(y|x) = 1, \quad \forall x \\ & P(y|x) \geq 0, \quad \forall x, y, \end{aligned} \tag{4.7}$$

where $\tilde{P}$ is the empirical distribution of the data, and $\phi_d(x,y) \to \mathbb{R}$ computes a (label-specific) feature statistic, *e.g.*, the planar-ness of a 3-D point. Using the method of Lagrange multipliers, the satisfying distribution has the exponential family form

$$P(y|x;\theta) = \frac{1}{Z_\theta(x)} \exp(\theta^T \phi(x,y)), \tag{4.8}$$

$$Z_\theta(x) = \sum_y \exp(\theta^T \phi(x,y)), \tag{4.9}$$

where Equation (4.9) is referred to as the partition function that normalizes the distribution. We refer to this conditional distribution as a MaxEnt "predictor". The resulting dual problem is the canonical maximum (log-)likelihood estimation (MLE) method, negated,

$$\min_\theta \mathbb{E}_{\tilde{P}(X,Y)}[-\log(P(Y|X;\theta))] \equiv \min_\theta \mathbb{E}_{x\sim\tilde{P}(X)} \left[\mathbb{E}_{y\sim\tilde{P}(Y|x)}[-\log(P(y|x;\theta))]\right]. \tag{4.10}$$

In practice, we assume the marginal distribution of the data $\tilde{P}(X)$ follows a uniform distribution.

Note that if we replace the equality constraints in Equation (4.7) with the inequality constraints

$$(\mathbb{E}_{P(X,Y)}[\phi_d(X,Y)] - \mathbb{E}_{\tilde{P}(X,Y)}[\phi_d(X,Y)])^2 \leq 2\lambda, \quad \forall d, \tag{4.11}$$

then the resulting dual problem is the maximum *a posteriori* (MAP) problem with L^2 regularization (Chen and Rosenfeld, 2000)

$$\min_{\theta} \frac{\lambda}{2}\|\theta\|_2^2 + \mathbb{E}_{x\sim\tilde{P}(X)}\left[\mathbb{E}_{y\sim\tilde{P}(Y|x)}[-\log(P(y|x;\theta))]\right]. \tag{4.12}$$

See (Dudik et al., 2007) for generalizations of modifying the constraints to use other convex functions.

4.3 The Subgradient Method

Learning the parameters of these linear models can be done using standard function minimization techniques. In this thesis, we are only concerned with minimizing convex functions due to their well-understood properties; see (Boyd and Vandenberghe, 2004) for in-depth analysis. While there exist a variety of minimization techniques that can be used, we use a simple first-order technique, the projected subgradient method (Shor, 1985), for its generality and efficient convergence properties (Ratliff et al., 2007).

Briefly, for a convex function $l : \Theta \to \mathbb{R}$ defined over a convex set Θ (*e.g.*, $\Theta \subseteq \mathbb{R}^d$, or, as is used in Chapter 8, Θ is the set of positive semi-definite, symmetric matrices), a subgradient $g_{\theta_0} \in \mathbb{R}^d$ of the function at a point θ_0 forms a linear lower bound of the function. Formally, the subdifferential $\partial l(\theta_0)$ of the function at point θ_0 is the set of all subgradients,

$$\partial l(\theta_0) = \{g_{\theta_0} \mid l(\theta) \geq l(\theta_0) + \langle g_{\theta_0}, (\theta - \theta_0)\rangle,\ \forall\theta \in \Theta\}. \tag{4.13}$$

If l is differentiable, *e.g.*, Equation (4.12), then $g_{\theta_0} \equiv \nabla_\theta l(\theta_0)$ and $|\partial l(\theta_0)| = 1$. For example, the (sub)gradient for one sample x of the negative log likelihood (log loss) of Equation (4.12) is

$$g_\theta = \mathbb{E}_{P(Y|x;\theta)}[\phi(x, Y)] - \mathbb{E}_{\tilde{P}(Y|x)}[\phi(x, Y)]. \tag{4.14}$$

In words, the gradient is the difference between the expected feature counts induced by the current model vs. empirical feature counts.

If l is non-differentiable, *e.g.*, Equation (4.5), then $|\partial l(\theta_0)| = \infty$ at each non-differentiable point θ_0. For example, a subgradient for one hinge loss term in the summation of Equation (4.5) is

$$g_\theta = w_i\left(\phi(x_i, k_i^*) - \phi(x_i, k_i)\right) \cdot \mathbb{1}((\theta^T\phi(x_i, k_i^*) + 1 - \theta^T\phi(x_i, k_i)) > 0), \tag{4.15}$$

$$\tag{4.16}$$

where

$$k_i^* = \arg\max_{k\in\mathcal{K}\setminus k_i} \theta^T\phi(x_i, k). \tag{4.17}$$

Algorithm 1 Online Subgradient Method

Given: Additive function to minimize $l(\theta) = \sum_i l_i(\theta)$, step-sizes $\{\alpha_t\}_{t=1}^T$
Output: $\arg\min_{\theta \in \Theta} l(\theta)$
$\theta = 0$
for $t = 1, \ldots, T$ **do**
  Randomly sample l_i
  Compute direction $g_\theta \in \partial l_i(\theta)$
  $\theta \leftarrow \mathcal{P}_\Theta(\theta - \alpha_t g_\theta)$
end for

In words, the subgradient is non-zero when the constraint is violated and its value is the difference between the features of the label with the highest margin/score, k_i^*, vs. the features of the true label, k_i, scaled by w_i.

In a similar iterative procedure as gradient descent, the function l can be minimized by taking small steps, $\alpha_t = \frac{c}{\sqrt{t}}$, where c is the learning rate and t is the iteration number, along the negative subgradients. When optimizing a function that is additive over a large number of samples, $l(\theta) = \sum_i l_i(\theta)$, faster convergence can be achieved by performing online, stochastic updates over the dataset, instead of in batch (Bottou and LeCun, 2004). Finally, when optimizing over a *closed*, convex set Θ and an update steps off the set, it is necessary to project to the closest point in the set, which we denote with the function $\mathcal{P}_\Theta : \mathbb{R}^d \to \Theta$. The entire procedure is summarized in Algorithm 1.

4.4 Boosting

As presented so far, the SVM and MaxEnt models are (log) linear in the features, *i.e.*, the score for assigning a sample x to the k'th label is defined as $\theta^T \phi(x, k)$. Often we can obtain better performance by using a more expressive model where the score is defined from a non-parametric function of the features, $f(\phi(x, y)) \to \mathbb{R}$. Alternatively, we can use a single feature representation of the data that is shared for all classes, which we simply denote as x, and have $f(x) \to \mathbb{R}^m$ output the scores for all classes at once. Then, we select the score for k'th class using the respective standard basis vector e_k. For example, we can specify the MaxEnt predictor as

$$\log P_f(y|x) = f(x)^T \mathsf{e}_y - \log Z_f(x). \tag{4.18}$$

Boosting (Freund and Schapire, 1997) is a powerful technique to fit a strong/complex function f as a weighted combination of weak/simple functions $\{h_t\}$,

$$f(x) = \sum_t \alpha_t h_t(x), \tag{4.19}$$

Algorithm 2 Boosting via Functional Gradient Descent

Given: Additive functional to minimize $L[f] = \sum_i l_i(f(x_i))$, step-sizes $\{\alpha_t\}_{t=1}^T$, hypothesis set of functions $\mathcal{H}$
Output: $\arg\min_f L[f]$
$f \leftarrow 0$
for $t = 1, \ldots, T$ **do**
  Compute functional gradient $\overrightarrow{\nabla}$ (Equation (4.21))
  Find $h_t \in \mathcal{H}$ that is maximally correlated with $-\overrightarrow{\nabla}$ (Solve Equation (4.25))
  $f \leftarrow f + \alpha_t h_t$
end for

where $\alpha_t \in \mathbb{R}$ are the weights. One way to view boosting is that it is performing gradient descent in the space of functions that minimizes the empirical loss, *e.g.*, hinge-loss and log-loss, over the training data (Mason et al., 1999, Friedman, 2001, Ratliff, 2009, Grubb and Bagnell, 2011). Using functional analysis techniques (Rudin, 1991), we define $L[f]$ to be the empirical loss functional we want to minimize, which measures the cumulative loss of a function f evaluated over the training set,

$$L[f] = \sum_i l_i(f(x_i)), \tag{4.20}$$

and we define $\overrightarrow{\nabla} : \mathcal{X} \to \mathbb{R}^m$ to be the functional gradient of Equation (4.20) evaluated at $f(x)$,

$$\overrightarrow{\nabla}(x) = \sum_i \frac{\partial L}{\partial f(x_i)}(x) = \sum_i \frac{\partial}{\partial f(x_i)} l_i(f(x_i)) \cdot \mathbb{1}(x = x_i). \tag{4.21}$$

If we descend along the successive functional gradients, with step-sizes α_t, the learned function will have the form

$$f(x) = -\sum_t \alpha_t \overrightarrow{\nabla_t}(x), \tag{4.22}$$

which is non-zero only at feature locations in the training set and will not generalize to new data. Instead, we step in the direction of the negative functional gradient $-\overrightarrow{\nabla}$ projected onto a hypothesis set functions $\mathcal{H}$ in order to generalize to new data. This projection is done by finding a function $h \in \mathcal{H}$ with maximal correlation with the negative

functional gradient,

$$h^* = \arg\max_{h \in \mathcal{H}} \frac{\langle h, -\overrightarrow{\nabla} \rangle}{\|h\|} \tag{4.23}$$

$$= \arg\max_{h \in \mathcal{H}} \frac{\sum_i \langle h(x_i), -\overrightarrow{\nabla}(x_i) \rangle}{\sum_i \|h(x_i)\|_2} \tag{4.24}$$

$$\equiv \arg\min_{h \in \mathcal{H}} \sum_i \left\| h(x_i) + \overrightarrow{\nabla}(x_i) \right\|_2^2, \tag{4.25}$$

where the equivalence holds under technical assumptions[3]. Hence, this projection procedure can be implemented by training a regressor (Section 4.1) to match the negative functional gradient evaluated at each sample in the training set. In Chapter 5 and Chapter 6 we detail the specifics of the functional gradients for specific loss functions that are used as the targets. As summarized in Algorithm 2, after T projected functional gradient updates, the function is then defined as the weighted sum of the fit regressors h (which are referred to as weak learners/predictors),

$$f(x) = \sum_t \alpha_t h_t(x). \tag{4.26}$$

When the functional L is non-differentiable, an analogous functional version of the parametric subgradient can be defined; however, the projection of the functional subgradient onto $\mathcal{H}$ may not necessarily improve the loss. In order to guarantee convergence, it has been shown that it is necessary to incorporate the cumulative errors from the previous projections at each step (Grubb and Bagnell, 2011).

[3]Notably, iff $\mathcal{H}$ is closed under multiplication (Friedman, 2001).

Chapter 5

Parsing Scenes with Graphical Models

5.1 Introduction

The typical approaches to the scene parsing problem are with graphical models, such as Conditional Random Fields (CRFs) (Lafferty et al., 2001), or unnormalized energy-based models, such as Max-Margin Markov Networks (M3Ns) (Taskar et al., 2003). The models tie together the labels of each site such that the labeling of the scene requires a joint optimization over all label variables, which typically is typically intractable to solve exactly. In this chapter, we review the common approaches under this framework and discuss their intrinsic limitations. Additionally, we extend upon previous work on higher-order models, *i.e.*, models that consider the label assignment to *regions* of sites at a time. Specifically, we demonstrate how to effectively learn these higher-order interactions using functional gradient techniques Section 4.4. Learning these higher-order models is important in practice as it enables one to easily reason over multiple segmentation hypotheses using feature statistics at multiple scales. We evaluate these methods for 3-D point cloud classification and estimating the 3-D geometric surfaces from images (Hoiem et al., 2007).

5.2 Background

In this section we describe the graphical framework in a general form. In the following section we instantiate the specific, computationally tractable forms of these models that are used in our experiments.

5.2.1 Conditional Random Fields

A Conditional Random Field (Lafferty et al., 2001) is an extension of the MaxEnt model presented in Section 4.2.2 to the joint distribution

$$P(Y|X) = P(Y_1, \ldots, Y_n|X), \tag{5.1}$$

where there n sites in the scene (*e.g.*, the number of points in a 3-D point cloud), $Y_i \in \mathcal{K}$ is the label random variable of the i'th site (*e.g.*, the label of a 3-D point), and X is the data variable (*e.g.*, the entire 3-D point cloud). Instead of matching feature statistics over individual sites (Equation (4.7)), the feature statistics are now coupled over groups of label random variables. For a given scene, we define C to be the set of all such groupings and $c \in C$ to be a particular grouping of indices, *i.e.*, $Y_c = \{Y_i \mid i \in c\}$. Then, the feature matching constraints are changed to

$$\mathbb{E}_{P(X,Y)}[\phi_c(X, Y_c)] = \mathbb{E}_{\tilde{P}(X,Y)}[\phi_c(X, Y_c)], \quad \forall c, \tag{5.2}$$

where $\phi_c(x, y_c) \to \mathbb{R}^d$ is a feature function that computes features statistics from the data x for the assignment y_c to grouping c, *e.g.*, the variance of the planar-ness of the 3-D points in the group c. To simplify the notation, we drop the data argument x and it is implicit that the feature function ϕ_c has access to the entire data. The satisfying dual distribution has the exponential family form

$$P(y|x;\theta) = \frac{1}{Z_\theta(x)} \exp(\sum_{c \in \mathcal{C}} \theta^T \phi_c(y_c)), \tag{5.3}$$

$$Z_\theta(x) = \sum_{y \in \mathcal{K}^n} \exp(\sum_{c \in \mathcal{C}} \theta^T \phi_c(y_c)). \tag{5.4}$$

This distribution can be interpreted as a Gibbs random field over a graph with n nodes and cliques defined by the groupings C. That is,

$$P(y|x;\theta) \propto \exp(\sum_{c \in \mathcal{C}} \psi_c(y_c;\theta)), \tag{5.5}$$

where $\psi_c(y_c;\theta) = \theta^T \phi_c(y_c)$ are the clique potentials. Furthermore, we define

$$\Psi(y;\theta) = \sum_{c \in \mathcal{C}} \psi_c(y_c;\theta), \tag{5.6}$$

to be the total (negative[1]) energy of the system.

[1]From a physics viewpoint, a highly probable assignment y is described to have low energy, whereas in this work we associte a large $\Psi(y)$ to indicate a good score in the assignment.

The gradient of the negative log-likelihood of Equation (5.3) has the familiar feature difference form,

$$-\frac{\partial \log P(y|x;\theta)}{\partial \theta} = \sum_{c \in C} (\mathbb{E}_P[\phi_c(Y_c)] - \phi_c(y_c)) \tag{5.7}$$

$$= \sum_{c \in C} (\sum_{\hat{y}_c} P(\hat{y}_c|x;\theta)\phi_c(\hat{y}_c) - \phi_c(y_c)). \tag{5.8}$$

Unfortunately, computing the exact marginal probabilities is NP-hard for general graphs. A tractable case is when the groupings C form a tree; however, this model greatly restricts the interactions among the sites. In practice, an approximate inference algorithm, *e.g.*, loopy belief propagation (Wainwright and Jordan, 2008), can be used instead to approximate the marginals. However, there are often limitations dependent on the form of the clique potentials, and the approximation can still be computationally expensive to perform in practice, especially for graphs with dense edges. Hence, due to the requirement of estimating the marginals at each gradient step, MLE is often limited to graphs that have low treewidth.

5.2.2 Max-Margin Markov Networks

A Max-Margin Markov Network (M3N) (Taskar et al., 2003) is the analog of an SVM to the structured prediction setting and can also be thought of as an unnormalized version of a CRF. In the spirit of the SVM formulation (Section 4.2.1), we can frame the problem to finding parameters that maximize the score of the scene's ground truth labeling, y, over any other hypothesis labeling, $\hat{y}$, by a margin. This can be formulated (Tsochantaridis et al., 2005) as the convex program

$$\min_{\theta,\xi} \quad \frac{\lambda}{2}\|\theta\|_2^2 + \sum_{\hat{y} \in \mathcal{K}^n} \xi_{\hat{y}} \tag{5.9}$$

$$\text{s.t.} \quad \Psi(y;\theta) \geq \Psi(\hat{y};\theta) + \mathbb{1}(\hat{y} \neq y) - \xi_{\hat{y}}, \quad \forall \hat{y} \tag{5.10}$$

$$\xi_{\hat{y}} \geq 0, \quad \forall \hat{y}. \tag{5.11}$$

Due to the number of constraints exponential in the number of sites n, this slack-scaling formulation is computationally intractable to solve, though there do exist tractable approximations for certain forms (Sarawagi and Gupta, 2008),

Alternatively, we can use a margin-scaling approach and change the margin depending on the discrepancy between different hypothesis labelings. This can be formulated

as the convex program,

$$\min_{\theta,\xi} \quad \frac{\lambda}{2}\|\theta\|_2^2 + \xi \tag{5.12}$$

$$\text{s.t.} \quad \Psi(y;\theta) \geq \Psi(\hat{y};\theta) + \gamma(y,\hat{y}) - \xi, \quad \forall \hat{y} \in \mathcal{K}^n \tag{5.13}$$

$$\xi \geq 0, \tag{5.14}$$

where $\gamma(y,\hat{y}) \to \mathbb{R}^+$ measures the discrepancy between the two labelings. Analogously with SVMs, Program (5.12) can be rewritten as the unconstrained convex minimization (Ratliff et al., 2007),

$$\min_{\theta} \frac{\lambda}{2}\|\theta\|_2^2 + \max_{\hat{y}\in\mathcal{K}^n}(\Psi(\hat{y};\theta) + \gamma(y,\hat{y})) - \Psi(y;\theta), \tag{5.15}$$

where the second and third terms are collectively referred to as the structured hinge-loss. In practice, we use the Hamming distance, $\gamma(y,\hat{y}) = \sum_i \mathbb{1}(y_i \neq \hat{y}_i)$, to measure the disparity between labelings[2].

A subgradient g_θ of the structured hinge-loss is

$$g_\theta = \sum_{c\in C}(\phi_c(y_c^*) - \phi_c(y_c)), \tag{5.16}$$

where

$$y^* = \arg\max_{\hat{y}\in\mathcal{K}^n}(\Psi(\hat{y};\theta) + \gamma(y,\hat{y})). \tag{5.17}$$

Hence, unlike MLE, the max-margin approach requires computing the MAP labeling, augmented with the margin term (Equation (5.17)), instead of computing the marginal probabilities. Unfortunately, again, computing the MAP labeling is NP-hard in general; however, there exists a much larger set of approximate inference techniques that can be used with the additional benefits of being convergent and often more computationally efficient in practice than message-passing algorithms. These techniques include graphcuts (Kolmogorov and Zabih, 2004), linear programs (Wainwright et al., 2005), and primal-dual formulations (Jojic et al., 2010). Additionally, using MAP inference in the inner loop of the subgradient method follows the beneficial observation to use the same approximate inference technique during both training and testing (Kumar et al., 2005, Wainwright, 2006).

5.3 Smoothing-based Markov Networks

In this section, we instantiate the specific models that we use in our experiments.

[2] In general, any function that quantifies margin between labelings can be used here; however, more complicated functions can make resulting optimization difficult to optimize.

5.3.1 Pairwise Model

We begin with the simplest structured model which only encodes interactions between pairs of variables,

$$\Psi(y;\theta) = \sum_{i \in C_1} \psi_i(y_i;\theta) + \sum_{ij \in C_2} \psi_{ij}(y_i, y_j;\theta), \tag{5.18}$$

where C_1 and C_2 are the sets of nodes and edges in the graph, respectively,

$$\psi_i(y_i;\theta) = \theta_1^T \phi_i(y_i), \tag{5.19}$$

$$\psi_{ij}(y_i, y_j;\theta) = \theta_2^T \phi_{ij}(y_i, y_j), \tag{5.20}$$

and $\theta = [\theta_1^T, \theta_2^T]^T$. Hence, in this formulation we have all node potentials ψ_i sharing the same parameters θ_1 and all edge potentials ψ_{ij} sharing the same parameters θ_2.

5.3.2 Associative Markov Networks

For cyclic graphs, finding the MAP labeling y that minimizes the energy, $-\Psi(y;\theta)$, in Equation (5.18) is NP-hard in general. However, if the label set is binary ($|\mathcal{K}| = 2$), and all potentials $ij \in C_2$ follow the form

$$-\psi_{ij}(0,0) - \psi_{ij}(1,1) \leq -\psi_{ij}(0,1) - \psi_{ij}(1,0), \tag{5.21}$$

then it can be shown that *exactly* minimizing this energy can be done by minimizing an equivalent submodular function over the nodes (Hammer, 1965, Kolmogorov and Zabih, 2004), which is implemented by solving a mincut problem over a graph. Note there are no constraints on the node potentials.

Unfortunately, yet again, finding the MAP labeling is NP-hard when there more than 2 labels ($|\mathcal{K}| > 2$). However, a factor of 2 approximation can be achieved by using the move-making α-expansion algorithm (Boykov et al., 2001). This algorithm requires the similar constraint,

$$-\psi_{ij}(\alpha,\alpha) - \psi_{ij}(y_i, y_j) \leq -\psi_{ij}(\alpha, y_j) - \psi_{ij}(y_i, \alpha), \quad \forall \alpha, y_i, y_j \in \mathcal{K}. \tag{5.22}$$

Hence, this constraint can be viewed as transforming the multi-class labeling into a binary-class labeling where value 1 in Equation (5.21) indicates the current assignment y_i and value 0 indicates the move to label α.

One multi-label model that satisfies both of the constraints in Equation (5.21) and Equation (5.22) is the Associative/Pott's Markov Network (AMN) (Taskar et al., 2004) with edge potentials with the form and property, respectively,

$$\psi_{ij}(y_i, y_j) = \mathbb{1}(y_i = y_j) \cdot \theta_2^T \phi_{ij}(y_i, y_j), \tag{5.23}$$

$$\psi_{ij}(y_i, y_j) \geq 0, \quad \forall y_i, y_j. \tag{5.24}$$

These potentials have a smoothing effect because better energy is achieved if the neighboring label assignments around a node are in agreement. In practice, ensuring the non-negativity constraint can be implemented by forcing the parameters θ_2 and features ϕ_{ij} to always be non-negative; Section 5.3.4 further discusses this issue. Hence, these associative potentials can not model repulsive properties such as certain labels should not be adjacent to each other.

5.3.3 Robust Associative Markov Networks

The above pairwise models are local in the sense that they reason among the labels of the sites in the scene. Especially in images to due scale ambiguity, it can be hard to define the right spatial support for feature representation: if it is too small then computed information is too local, and if it is too big then the are errors for defining a site over multiple classes.

Alternatively, the graphical model approach can be extended to consider higher-order interactions that reason over groups/regions of sites at a time,

$$\Psi(y;\theta) = \sum_{i \in C_1} \psi_i(y_i;\theta) + \sum_{ij \in C_2} \psi_{ij}(y_i, y_j;\theta), + \sum_{r \in C_3} \psi_r(y_r;\theta), \tag{5.25}$$

where C_3 are the set of regions/grouped sites, $Y_r = \{Y_i \mid i \in r\}$ and $\psi_r(y_r;\theta) = \theta_3^T \phi_r(y_r)$. While these expressive models can potentially encode more complex iterations, performing tractable (approximate) inference remains the bottleneck.

One class of higher-order models for which inference is tractable is the extension of the associative model to higher-order cliques,

$$\psi_r(y_r) = \sum_{k \in \mathcal{K}} \Pi_{i \in r} \mathbb{1}(y_i = k) \cdot \theta_3^T \phi_r(k). \tag{5.26}$$

Hence, this model prefers for entire regions to take on the same label. As similar to the pairwise models, if ψ_r is always non-negative, then MAP inference can be performed exactly using a linear program (Taskar et al., 2004) or a mincut (Kohli et al., 2009) in the two label case, and approximately in the multi-label case.

While this model enables one to extract statistics over regions in the scene, it is quite restrictive as the model encodes the same score if 99% of the $y_i \in y_r$ are the same value as if only 1% were the same. A softer/robust version of this model is the *Robust P^n Pott's* model (Kohli et al., 2009) which enables some a portion of the region to be in disagreement. Letting $\rho_k(y_r) = \frac{1}{|y_r|} \sum_{y_i \in y_c} \mathbb{1}(y_i = k)$ be the fraction of $y_i \in y_r$ have value k, the potential is defined

$$\psi_r(y_r) = \sum_{k \in \mathcal{K}} \left(\mathbb{1}(1 - \rho_k(y_r) \leq q_r) \cdot (1 - \frac{1 - \rho_k(y_r)}{q_r}) \cdot \theta_3^T \phi_r(k) \right), \tag{5.27}$$

where $q_r = q|y_r|$ and $0 < q < 0.5$ is a parameter that controls the robustness of the potentials. In words, q can be thought as the fraction of the region that can disagree with the most common (mode) label in the region. The indicator function specifies that the potential is non-zero for the k'th label that is the mode label. The middle term then scales down the score for assigning the k'th label to region label by the number of nodes in the region that disagree with the mode. The constraint on q ensures at most only one indicator term in the summation is satisfied. This model, along with several variants (*i.e.*, linear envelope potentials (Kohli and Kumar, 2010)) can be optimized using mincuts.

5.3.4 Learning

As previously mentioned, a benefit of the max-margin approach over MLE is that there exists a larger set of approximate inference techniques which we can use in the inner loop inside the subgradient method. For the smoothing models above, we require the potentials to be non-negative, which can be done by forcing the features to be positive and projecting the respective parameters (θ_2, θ_3) onto the positive orthant after each subgradient update. However, models learned using this approach often converge slowly in practice and are typically dominated by the cliques parameters that occur most frequently. For example, there are typically many more edges than nodes in a graph, so for the subgradient of Equation (5.18), the components corresponding to θ_2 are typically larger in scale than the components corresponding to θ_1. Furthermore, the SVM and M3N models are not invariant to scale of the features (Herbrich and Graepel, 2000). For example, were we to scale a random feature dimension by $1,000$, that scaled feature would essentially determine the margin and the parametric optima would converge to a different solution, which is not the equivalent rescaled solution (and most likely worse). Due to these issues, it can be hard to effectively regularize these parametric models in practice.

Instead, we demonstrate that learning these models using functional gradient boosting obtains models with better predictive performance. In contrast to related work (Dietterich et al., 2004, Torralba et al., 2004, Liao et al., 2007), this max-margin approach provides an efficient and effective way to integrate feature statistics computed over large regions in the scene, *i.e.*, from multiple segmentations, and to learn non-parametric potentials without requiring the marginal probabilities.

As a concrete example, we consider a pairwise, Pott's model with energy,

$$\Psi(y; f) = \sum_{i \in C_1} \psi_i(y_i; f) + \sum_{ij \in C_2} \psi_{ij}(y_i, y_j; f), \tag{5.28}$$

where

$$\psi_i(y_i; f) = f_1(\phi_i(x))^T \mathsf{e}_{y_i}, \tag{5.29}$$

$$\psi_{ij}(y_i, y_j; f) = \mathbb{1}(y_i = y_j) \cdot f_2(\phi_{ij}(x))^T \mathsf{e}_{y_i}, \tag{5.30}$$

and $\phi_i : \mathcal{X} \to \mathbb{R}^{d_1}$ and $\phi_{ij} : \mathcal{X} \to \mathbb{R}^{d_2}$ extract feature representations for the respective node and edge cliques. Then, defining the loss function

$$l(f(x, y)) = \max_{\hat{k}} (\Psi(\hat{y}; f) + \gamma(y, \hat{y})) - \Psi(y; f), \tag{5.31}$$

we train regressors h_1 and h_2 to match the respective targets (negated)

$$\frac{\partial}{\partial f_1(x, y)} l(f(x, y)) = \sum_{i \in C_1} \left(\mathsf{e}_{y_i^*} - \mathsf{e}_{y_i} \right), \tag{5.32}$$

$$\frac{\partial}{\partial f_2(x, y)} l(f(x, y)) = \sum_{ij \in C_2} \left(\mathbb{1}(y_i^* = y_j^*) \mathsf{e}_{y_i^*} - \mathbb{1}(y_i = y_j) \mathsf{e}_{y_i} \right), \tag{5.33}$$

at each node $\phi_i(x)$ and edge $\phi_{ij}(x)$ feature location.

In order to ensure non-negative clique potentials, we use the analog of exponentiated gradient descent (Kivinen and Warmuth, 1997) to function space (Ratliff, 2009). As in the parametric version, this algorithm places a different prior over the space of functions and encourages functions emphasizing few regions in feature space and little everywhere else. The update rule is changed from

$$f \leftarrow f + \alpha_t h_t \tag{5.34}$$

to

$$f \leftarrow f \cdot \exp(\alpha_t h_t). \tag{5.35}$$

5.4 Experimental Analysis

We compare the performances of M3N models trained with the subgradient method and functional gradient boosting on two vision problems: 1) 3-D point cloud classification from laser range finders and 2) 3-D geometric surface estimation from images (Hoiem et al., 2007). These experiments demonstrate the difference in predictive performance of the models when using different optimization procedures.

Table 5.1: Per-class F_1 scores and overall point accuracy comparisons.

	HO-FGrad	HO-PGrad	Pair-PGrad
Vegetation	**0.96**	0.93	0.92
Wire	**0.64**	0.40	0.40
Pole/tree-trunk	**0.39**	0.35	0.30
Ground	0.99	0.99	0.99
Facade	0.89	0.89	0.89
Overall Accuracy	**97.2**%	96.1%	95.7%

5.4.1 3-D Point Cloud Classification

Random field Structure

We create models using pairwise and higher-order models. The regions for the higher-order models result from performing two segmentations of K-means over on the 3-D coordinates using values $K_1 = 0.026n$ and $K_2 = 0.042n$, where n is the number of 3-D points in the scene. These values were chosen to generate compact groups of 3-D points that typically consisted of a single label. The node features are the Geom, Orient, Elev features from Section 3.1.2, the edge features are the difference Geom and Orient features, and the region features are the same as the nodes' except except now the neighborhood volume is defined within a 1.0 m radius from the clique medoid.

Models

We compare the performances of three AMN models. The fist AMN model is a pairwise model (Equation (5.18)) trained using the parametric subgradient method (Section 4.3), which we refer to as Pair-PGrad. The next two AMN models are both higher-order models (Equation (5.25)) trained using the parametric and functional (Section 4.4) subgradient methods, which we refer to as HO-PGrad and HO-FGrad, respectively. The HO-FGrad model was trained using linear regression to project the functional gradient.

Results

We evaluate performance on the VMR Oakland-v1 (Section 2.2.3) dataset. The performance breakdowns of the models are given in Table 5.1. Though the change in overall accuracy is marginal, this can be attributed to the severe imbalance in label proportions. However, the table shows favorable per-class improvements with HO-FGrad over Pair-PGrad for the smaller classes: wire and pole/tree-trunk. In addition, there is noticeable improvement in the vegetation class as well. Figure 5.1 qualitatively demonstrates this behavior of HO-FGrad producing improved classifications in challenging areas.

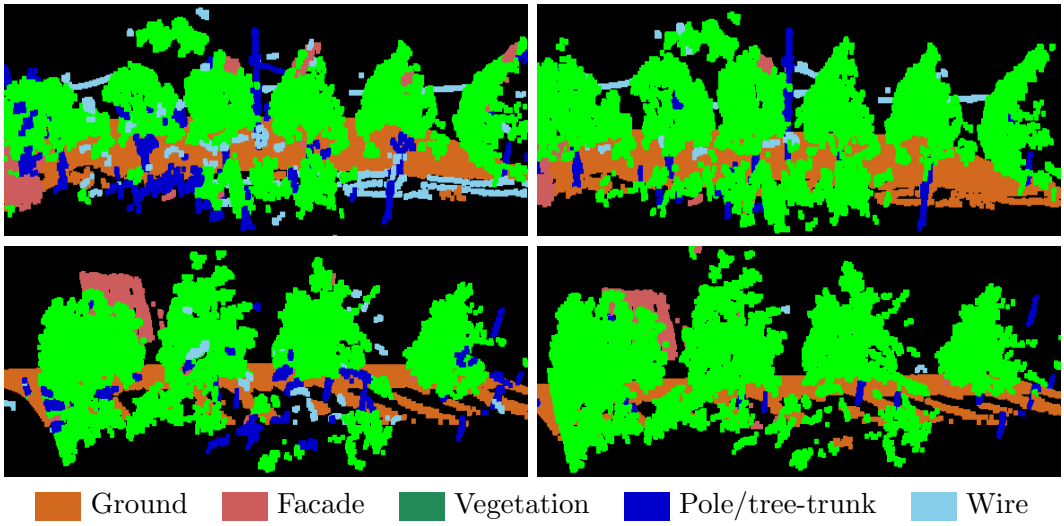
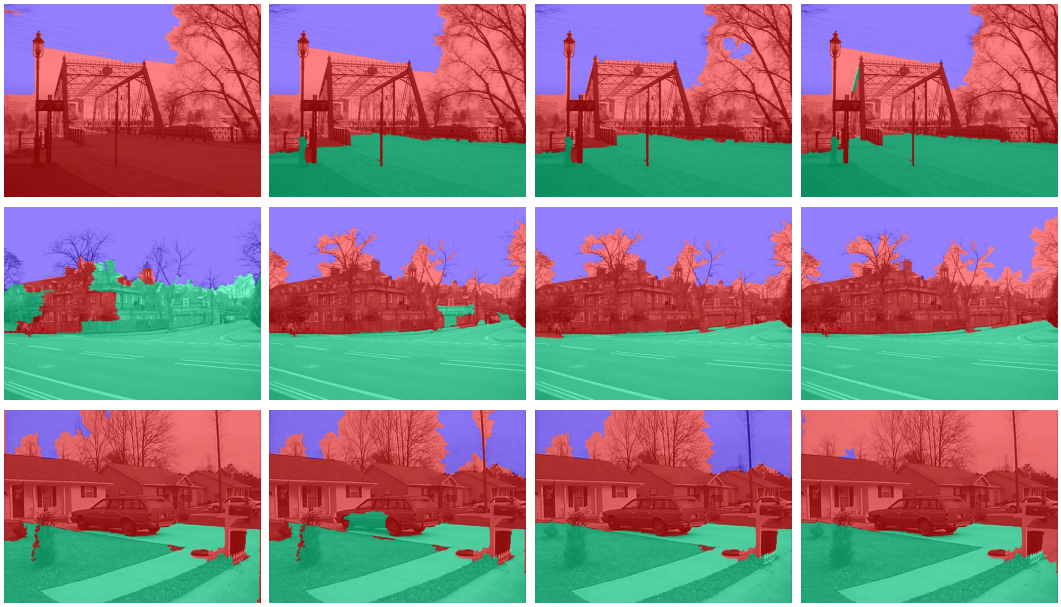


Figure 5.1: Qualitative comparisons of 3-D point cloud classification on two scenes: HO-PGrad (left side) and HO-FGrad (right side).

In our experiments, we found a trade-off between improving precision and better preserving object boundaries. In principle, one would expect the robust potentials to improve performance as there will always be segments spanning multiple objects. However, in this application we found that the number of such regions is small compared to the number of regions with noisy data, as evident in Figure 5.1. The robust potentials caused a decrease in performance because they allowed the noisy points to maintain the wrong labels. The next experiment will demonstrate how robust potentials can be used to improve classification when there are many more segments containing inhomogeneous labels.

5.4.2 3-D Surface Estimation

We also evaluated these models on the problem of recovering 3-D geometric surfaces from from images, *i.e.*, classifying pixels into one of the three classes: {Ground, Vertical (object standing on the ground), and Sky}. We use the Geometric Context Dataset provided by (Hoiem et al., 2007), where the overall distribution of pixels per class is: $\mathit{Ground} = 31.4\%$, $\mathit{Vertical} = 48.0\%$, $\mathit{Sky} = 20.6\%$. In (Hoiem et al., 2007), the authors extract features from overlapping segments in the image and average the independent classifications from boosted decision tree classifiers. In this experiment we compare the method used in (Hoiem et al., 2007) with different M3Ns models. We note that recent work from (Saxena et al., 2008), have also used random fields to extract 3-D information from single images. However, our model is not best suited in that scenario since we perform discrete label classification and they are estimating continuous distances.



(a) (b) (c) (d)

Figure 5.2: Qualitative comparison of geometric surface estimation on three scenes: (a) POTTS-HO-PGRAD, (b) POTTS-HO-FGRAD, (c) ROBUST-HO-FGRAD, (d) (Hoiem et al., 2007).

Features

In (Hoiem et al., 2007), the authors first group pixels into superpixels to use as the sites in their classification. They then perform 15 different segmentations: by increments of 5 segments between 5 and 50, and then by increments of 10 segments until 100. Each segment is composed of a group of superpixels. For each superpixel and segment, they extract 50 and 94 features, respectively, which capture location, shape, color, texture, and perspective. We used the features provided by the authors.

Random Field Structure

In the M3N model, the nodes represent superpixels and the segments are the cliques. As the sizes of the segments from segmentation-5 will differ with those from segmentation-100, the types of computed features will also vary with respect to the segment size. We model how these features vary per clique by creating different potentials based on segment size. In addition to the nodes (C_1), we defined three clique-sets: C_2, C_3, C_4, each of which contain the regions resulting from the five smallest, middle, and largest segmentations, respectively.

	Potts-HO-PGrad			Potts-HO-FGrad		
	Ground	Vertical	Sky	Ground	Vertical	Sky
Ground	0.74	0.24	0.02	0.84	0.15	0.01
Vertical	0.24	0.70	0.07	0.13	0.83	0.04
Sky	0.03	0.20	0.78	0.02	0.07	0.91
	Accuracy: 72.8%			Accuracy: 84.9%		

	Robust-HO-FGrad			(Hoiem et al., 2007)		
	Ground	Vertical	Sky	Ground	Vertical	Sky
Ground	**0.85**	0.14	0.01	0.83	0.16	0.00
Vertical	0.13	0.84	0.04	0.09	**0.89**	0.02
Sky	0.01	0.05	**0.94**	0.00	0.10	0.89
	Accuracy: 86.0%			Accuracy: 87.1%		

Figure 5.3: Quantitative comparisons on the Geometric Context dataset. Confusion matrices are row-normalized.

Models

We evaluate three different M3N models and compare with the original algorithm of (Hoiem et al., 2007). The first model is learned using the subgradient method with Potts model interactions (Potts-HO-PGrad). The second model uses the functional gradient learning with Potts model interactions (Potts-HO-FGrad). Because many cliques in this framework will contain multiple labels, we train the final model using functional gradient learning with Robust Potts interactions (Robust-HO-FGrad). Again, we use linear regressors to project the functional gradient.

We define the robustness of the clique-sets corresponding to the five largest (C_4) and five second largest (C_3) segments to allow 20% ($q_r = 0.2$) and 10% ($q_r = 0.1$), respectively, of the nodes within a clique to disagree with the mode label. We use a Potts model ($q_r = \epsilon$) for the clique-set with the smallest segments (C_2). In this application, the feature dimensions are ten times the size as the features used in the point cloud experiments. We found that the exponentiated functional gradient obtained better predicted performance. Finally, we make a small modification to the functional gradient algorithm. Since the size of the segment can be an indicator of the quality of features extracted, we weight the clique features during regression proportional to the number of pixels contained in the clique.

Results

In (Hoiem et al., 2007), the authors evaluate their algorithm by performing five-fold classification on a dataset of 250 images where 200 random images are chosen for training

and the remaining 50 are used for evaluation. Visual comparisons of the four models are presented in Figure 5.2. Figure 5.3 gives the quantitative classification comparisons of the above three models. We immediately see a clear improvement in performance across all classes using POTTS-HO-FGRAD model over the parametric version. Furthermore, using the robust potentials (ROBUST-HO-FGRAD) improves per-class performance, especially the preservation of the smallest class, Sky.

5.5 Summary

This chapter analyzed the canonical approach of using graphical models for scene parsing. Specifically, we started out with the probabilistic Conditional Random Field (Lafferty et al., 2001) framework, derived its unnormalized counterpart, the Max-Margin Markov Network (Taskar et al., 2003) framework, and then discussed how we can leverage recent advances in inference (Kohli et al., 2009) and learning (Ratliff, 2009) techniques, to learn models that account for feature statistics and label compatibility defined over *regions* of pixels/points in an image/3-D point cloud (higher-order cliques).

Our experiments demonstrated that learned parametric models were inferior to those obtained with non-parametric/functional formulations of the learning objective. Practically, the inferiority of the parametric model can be attributed to the imbalance of features for the different parameters. This imbalance comes from both the imbalance in number of samples per class, but more importantly, due to the imbalance in number of clique types (*e.g.*, the number of nodes vs. edges) in the graph. Hence, because the parametric formulation is not invariant to this imbalance (Herbrich and Graepel, 2000), we observed convergence to a solution whose performance was often much worse to the non-parametric formulation which is agnostic to this discrepancy. Additionally, with the non-parametric approach, we can derive models with non-linear predictions such as decision trees or neural networks (though we did not pursue this in our experiments).

Finally, we highlighted a major bottleneck for training these global models: the need for practically-efficient, global inference techniques. In the following chapter, we discuss how we can eschew this problem as well as the theoretical difficulties that arise when only an approximate solution can be reached.

Chapter 6

Hierarchical Inference Machines

6.1 Motivation

Before proceeding, it is important to highlight fundamental drawbacks of using (unnormalized) graphical models for scene parsing that were briefly mentioned in the previous chapter.

6.1.1 Restrictive Interactions

One fundamental limitation is that performing *exact* inference even over the simplest models, *e.g.*, a multi-class Pott's model, is often NP-hard. Additionally, in order to perform efficient, approximate inference with bounded guarantees on the solution, only a restrictive class of models can be used, *e.g.*, smoothing-like potentials (Veksler, 1999, Kohli et al., 2009) and linear envelope potentials (Kohli and Kumar, 2010). Typically these potentials greatly restrict the types of interactions among objects and/or are limited to pairwise interactions. While other approximate inference techniques can be used to handle more general potentials, *e.g.*, Sequential Tree-ReWeighted message passing (TRW-S) (Kolmogorov, 2006), these techniques are typically much more computationally expensive.

6.1.2 Modeling Mismatch

As we are forced to use an approximate inference technique during learning, it has been demonstrated that the learned model (*e.g.*, parameters) is tightly tied to the chosen inference procedure in both theory (Wainwright, 2006) and in practice (Kumar et al., 2005). Additionally, it is unclear how to learn the best model for the chosen inference procedure. Further complicating matters, there are cases when the energy of the *true* maximum *a posteriori* (MAP) labeling is *worse* than the energies returned from vari-

ous approximate inference techniques (Szeliski et al., 2007). This undesirable behavior suggests an incorrectly modeling of the problem.

6.1.3 Provably Difficult to Learn

Finally, from a theoretical viewpoint, correctly learning these models requires exact inference, and it is not well understood when approximate MAP or marginal inference is used in its place (Wainwright, 2006, Kulesza and Pereira, 2007, Finley and Joachims, 2008). For example, in (Kulesza and Pereira, 2007), the authors demonstrate a pathological case over a four-node graph with two parameters where using an approximate MAP inference technique during the learning procedure prevents convergence onto the true solution. Similarly, in another example they show that learning with loopy belief propagation can also diverge in practice.

Due to the above three fundamental limitations when training models relying on approximate inference techniques, this work proposes to bypass the use of global, probabilistic models for scene parsing. Instead, in a manner that is inspired by the inference mechanics over these models, we propose to break down the complex inference process into a series of simple machine learning subproblems that iteratively decodes the scene.

6.2 Approach

6.2.1 Intuition

In this work, we propose a hierarchical approach for scene parsing. Our approach is reminiscent of early vision literature in that we use a hierarchical decomposition of the image in order to encode relational and spatial information (Bouman and Shapiro, 1994, Feng et al., 2002). As we saw with the higher-order model in the previous chapter, these probabilistic models typically have a node to represent the label of a site and variously sized groupings of the nodes to form hierarchical regions. The nodes at the bottom of the hierarchy provide low-level discriminative information, while regions towards the top of the hierarchy resolve ambiguities using global information.

While the structure of the representation is similar to a graphical model, the key difference is that we no longer attempt to model the joint distribution/energy. Instead, we consider the mechanics of an approximate inference algorithm, specifically variational mean field (Wainwright and Jordan, 2008), over this hierarchical representation as a procedure. That is, we unroll the procedure as a sequence of computational modules taking in observations and other local computations on the graph (messages). Then, we train each of these modules, taking in as input the observations and messages, to predict the ideal intermediate messages. After traversing over the hierarchical representation as

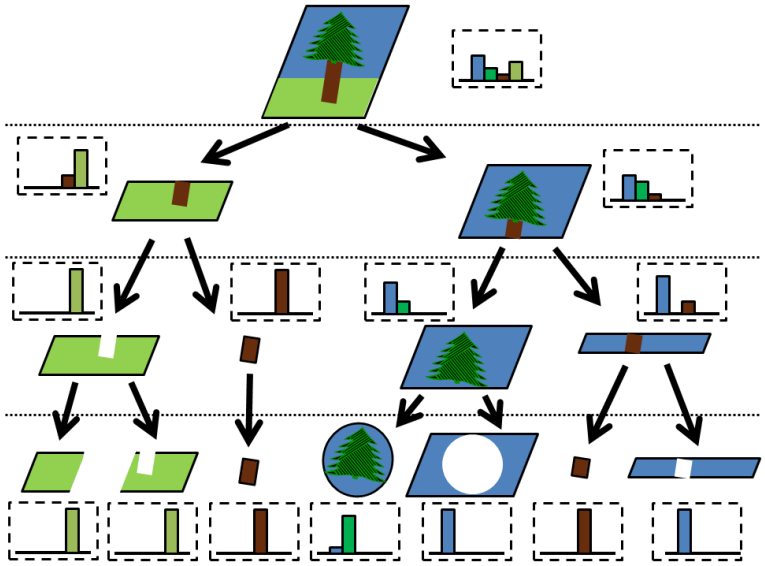


Figure 6.1: Simplified mechanics of a top-down hierarchical inference machine on a synthetic image. Given an image and its hierarchical decomposition of regions, we sequentially predict the proportion of labels present (drawn in the dashed boxes) using image features and previous predictions.

desired, the final parsing of the scene is extracted from the nodes of the site. Note that each prediction along the sequence will propagate relevant contextual information which can help the intermediate modules make correct predictions. We refer to this process of iterated predictions over a hierarchical representation of the scene as a *hierarchical inference machine* (HIM).

An idealized example of a top-down version of our approach on an image is depicted in Figure 6.1. Given an image, we first create a hierarchy of regions that range from very large regions in the image (potentially including the image itself as one region at the top) down to small regions (*e.g.*, superpixels) at the bottom. See for Section 3.2 for techniques to create this segmentation. We represent the inference procedure as a series of predictions along the hierarchy, in this case, from coarse to fine. Ideally, high levels in the hierarchy can represent the type of environment which "primes" the lower levels with a smaller set of labels to consider. We do not assume that this segmentation will contain regions with homogeneous labels. Instead, we explicitly model the *distribution* of labels inside a region to represent the fact that the segmentation is often imperfect in practice. Starting with the region that is the entire image at the top of the hierarchy, we train a predictor to predict the proportions of labels inside of it. As will be discussed in detail, these predictions are passed to the child level and are used to train another predictor for the children subregions' proportions of labels. Incorporating predictions from the previous predictor can be thought as message passing in a graphical model.

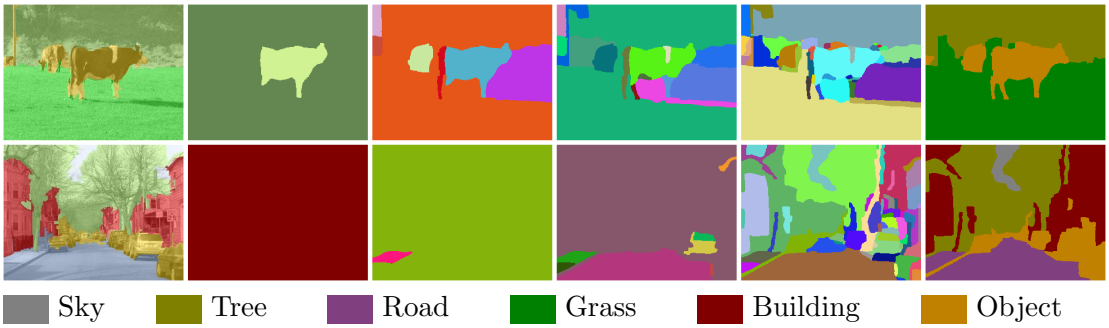


Figure 6.2: Example hierarchical segmentation and classification from STANFORD BACKGROUNDdataset. From left to right: the input image with colorized ground truth overlaid; four levels in the segmentation hierarchy; the inferred parsed scene from taking the max label for each region in the leaf level.

However, note that messages are not combined by a sum of products, but rather are arbitrarily combined by the predictor. The procedure ends when the leaves are reached and the labeled scene is derived from the max label inside each region in the leaves. An example hierarchical segmentation and labeling are shown in Figure 6.2. Note that since we model label proportions over regions:

- We do not make a single hypothesis segmentation.
- We are robust to imperfect segmentations.
- We can use features defined over large regions.
- We do not make hard commitments during inference.

6.2.2 Related work

Our hierarchical formulation resembles early directed graphical models from (Bouman and Shapiro, 1994) and (Feng et al., 2002) for scene analysis. Whereas these approaches rely on tree-based interactions to enable tractable learning, we no longer train a graphical model and are not restricted in the types of contextual cues that we can use. Instead we focus on maximizing what we ultimately care about: predicting correct labelings. This idea is analogous to the difficult and non-convex problem of maximizing the marginals (Kakade et al., 2002). The notion of training the inference algorithm to make correct predictions is also similar to (Barbu, 2009) for image denoising, in which a model is trained knowing that an inaccurate, but fast, inference algorithm will be used. In our approach we break up the complex structured prediction problem into a series of simpler classification problems, inspired by recent works in machine learning focused on

sequence prediction (Cohen and Carvalho, 2005, Daume III et al., 2009). In the vision setting, this notion of a series of classification problems is similar to Auto-context (Tu and Bai, 2010), in which pixel classifiers are trained in series using the previous classifier's predictions with pairwise information to model contextual cues. In our work, we go beyond typical site-wise representations that require entities to contain one label. Because we model label proportions, we can use features defined over large regions to better represent the context, rather than an aggregation of site-wise labels. Furthermore, the hierarchy provides spatial support context between levels and naturally propagates long-range interactions that may be hard to capture with pairwise interactions. We build on the forward sequential learning approach used and analyzed in (Viola and Jones, 2004, Heitz et al., 2008, Ross and Bagnell, 2010) to prevent cascading errors and leverage the sequential stacking idea to minimize cascaded overfitting (Wolpert, 1992, Cohen and Carvalho, 2005, Kou and Cohen, 2007).

6.3 Parsing Images with Inference Machines

For clarity in the explanation, we describe the HIM framework applied to parsing images. We will describe the application to 3-D point clouds later in this chapter at Section 6.5.

6.3.1 Overview

Given an image and its hierarchical region representation, we train a series of predictors, where each predictor operates on regions from a level in the segmentation hierarchy. Each predictor is explicitly trained to predict the proportion of labels that are contained within each region of its respective level. Then, the predictions are passed to, and incorporated by, the subsequent predictor, *etc.*. Figure 6.3 illustrates (on test data) how the predicted probabilities are refined while traversing coarse-to-fine down the hierarchy.

The main components of the training procedure are detailed over the following subsections:

- Training the predictor to predict label proportions (Section 6.3.2).
- Incorporating predictions from the previous predictor (Section 6.3.3).
- How to effectively train the entire procedure (Section 6.3.5).

6.3.2 Modeling Heterogeneous Regions

We define $\mathcal{K}$ to be the set of possible labels, $x \in \mathcal{X}$ to be a specific image, $\mathcal{R}(x)$ to be the set of regions of the hierarchical segmentation of image x, $r \in \mathcal{R}_\ell(x)$ to be a specific region in the ℓ'th level of the hierarchy, and L to be the number of levels in the hierarchy.

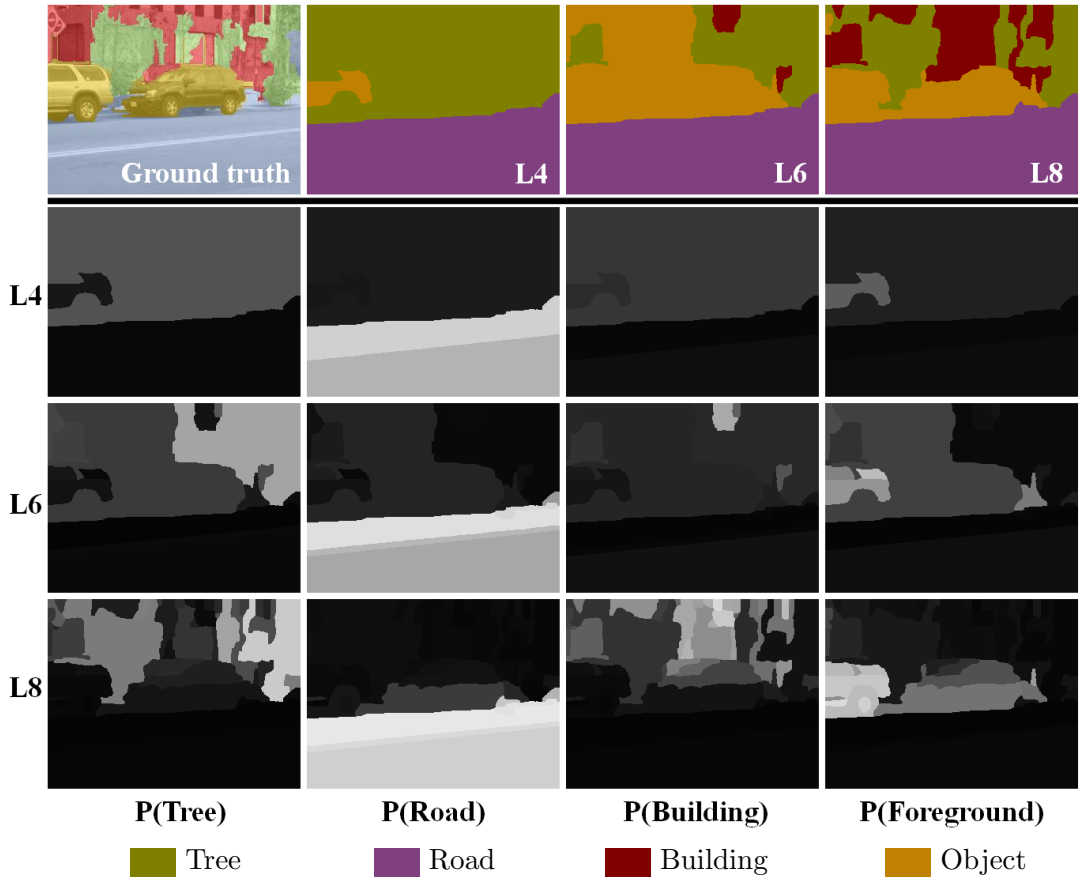


Figure 6.3: Refinement of predicted label proportions while traversing coarse-to-fine down the hierarchy. Row 1: Test image (with ground truth overlaid) and the max labels for each levels' regions. Rows 2-4: The respective level's label probability maps, where white indicates high probability.

Given an image x with each pixel labeled, for any region $r \in \mathcal{R}(x)$, we can compute an empirical distribution $\tilde{P}$ of labels for the region. Specifically, we denote

$$\tilde{P}(Y_r = k|x) = \text{The \% of pixels assigned to the } k\text{'th class in region } r \text{ of image } x. \quad (6.1)$$

Given features for region r, we want to train a predictor to match this distribution of labels for the region.

We use the MaxEnt framework for its properties discussed in Section 4.2.2, where we use the feature function $\phi_r(x, k)$ to compute (label-specific) features for region r (*e.g.*, Section 3.3). At level ℓ in the hierarchy, we train a MaxEnt predictor by minimizing the empirical log-loss,

$$\min_{\theta} \frac{\lambda}{2}||\theta||_2^2 - \sum_{x \in \mathcal{X}} \sum_{r \in \mathcal{R}_\ell(x)} \sum_{k \in K} \tilde{P}(Y_r = k|x) \log P(Y_r = k|x; \theta), \quad (6.2)$$

Figure 6.4: Illustration of the pixels being used (grayed) to compute the context features. In c_{local}, the pixels above and below the dashed line are used to compute c^{+}_{local} and c^{-}_{local}, respectively.

where

$$P(Y_r = k|x;\theta) = \frac{\exp(\theta^T \phi_r(x,k))}{\sum_{k' \in \mathcal{K}} \exp(\theta^T \phi_r(x,k'))}. \tag{6.3}$$

If we want a non-linear model, we can similarly use boosting (Section 4.4). We denote $\tilde{b}_r, b_r \in \mathbb{R}^{|\mathcal{K}|}$ to be the vectorized forms of the discrete distributions $\tilde{P}(Y_r|x), P(Y_r|x)$, respectively, and $\phi_r(x) \to \mathbb{R}^d$ to compute region features from the image that are shared for all classes. Then, defining the loss function

$$l_r(f(x)) = -\sum_{k \in K} \tilde{P}(Y_r = k|x) \log P(Y_r = k|x; f), \tag{6.4}$$

we train regressor h to match the targets (negated)

$$\frac{\partial}{\partial f(x)} l_r(f(x)) = b_r - \tilde{b}_r, \tag{6.5}$$

at feature location $\phi_r(x)$.

Partial Annotations

Training the predictors assumes each region has a valid empirical label distribution, *i.e.*, it sums to one. However, obtaining images that are fully annotated can be difficult in practice, and this is the case for many datasets. In these scenarios, we simply discard regions that are not sufficiently labeled – we use a threshold of 75% – and for the remaining distributions that do not sum to one, we assume the unlabeled pixels belong to a "none-of-the-above" class.

6.3.3 Context Features

In graphical models, the beliefs of a node's current marginal distribution are typically propagated between adjacent factors via message-passing algorithms (Wainwright and

Jordan, 2008). Informally, during an update a node receives messages from its neighbors, combines them (typically, by multiplying them), and the node sends its updated beliefs to its neighbors. One can view the step of combining the messages as the result of a simple function of the messages that takes the product of its inputs. From this perspective, a natural idea is to use a more sophisticated function that combines the messages (and the available image features) in an arbitrary manner. Furthermore, for each message-update we can train the function to directly predict what the node's state/marginal distribution *should* be, which is known at training time. Finally, as the initial predictions will be noisy, we can iterate this procedure over multiple passes of the graph.

Using this strategy, we can traverse over the hierarchy, level-by-level, and train a predictor for each level to use predictions from the previous predictor. For example, in a coarse-to-fine traversal, having access to the parent region's predicted label distribution should make it easier for its smaller child region to make a refined prediction that spatially localizes the labels. However, using the parent region's prediction is only one "message" for the child. We need not limit ourselves to simple (pairwise) interactions and can instead form arbitrary interactions from the previous predictions in the sequence. However, since the number of regions is of variable length per image, we need a way to summarize the previous predictions into a fixed-length *context feature* descriptor. It is this context information that ties together the per-region predictions to yield a globally consistent parsing.

For each image x, we denote the set of predicted probability vectors for all the regions in the image that the predictor operated on as

$$\mathcal{B}_t = \{b_r\}_{r \in \{\mathcal{R}_{\ell(t)}(x)\}}, \tag{6.6}$$

where t indexes the predictor P_t in the sequence and $\ell(t)$ returns the respective level of the hierarchy that P_t operated on. When training the t'th predictor, we will use $\mathcal{B}_{t-1}$ to compute different types of context feature descriptors. The following describes three types of context features that are useful for a top-down inference machine.

The first context feature encodes what was predicted for a particular region's parent,

$$c_{\text{parent}}(r, \mathcal{B}) = b_{\pi(r)}, \tag{6.7}$$

where $\pi(r)$ returns the parent region for region r. This is useful for priming the child region to make a more refined, spatially localized prediction. The description of the next two are illustrated in Figure 6.4. The second context feature models the global context by encoding the weighted average of the of all predictions in the previous level, weighted by the region sizes,

$$c_{\text{global}}(\mathcal{B}) \propto \sum_{b_r \in \mathcal{B}} |r| b_r. \tag{6.8}$$

This global distribution models the co-occurence of labels in the entire scene. The third context feature, models the local context around a region. We denote $\mathcal{I}_r$ to be a local neighborhood of pixel indices around region r. (In practice, this local neighborhood can come from dilating the region or placing an expanded bounding box around the region. In our experiments use the expanded bounding box and observed a slight decrease in performance with much improved computational efficiency over region dilation.) The local context is encoded by averaging the per-pixel probabilities, which are obtained by mapping the per-region probabilities from the previous predictor, intersected with this local neighborhood,

$$c_{\text{local}}(r, \mathcal{B}) \propto \sum_{b_{r'} \in \mathcal{B}} (r' \cap \mathcal{I}_r) b_{r'} \tag{6.9}$$

To encode spatial layout, we compute two separate averages for pixels above (c^{+}_{local}) and below (c^{-}_{local}) the region centroid, respectively. Now, when training the predictor P_t, we use $\mathcal{B}_{t-1}$ to compute an augmented feature descriptor for each region,

$$\breve{\phi}_r(x, \mathcal{B}_{t-1}) = [\phi_r(x) \ ; \ c_{\text{parent}}(r, \mathcal{B}_{t-1}) \ ; \ c_{\text{global}}(\mathcal{B}_{t-1}) \ ; \ c^{+}_{\text{local}}(r, \mathcal{B}_{t-1}) \ ; \ c^{-}_{\text{local}}(r, \mathcal{B}_{t-1})]. \tag{6.10}$$

6.3.4 Parsing Strategies

As described so far, given a segmentation hierarchy of L levels, parsing the scene is done by performing a sequence of $T = L$ predictions that traverses the hierarchy from coarse-to-fine. However, similar to how message-passing algorithms perform multiple passes over a graphical model, we can also arbitrarily traverse the segmentation hierarchy with multiple predictors. Specifically, instead of training a single predictor for the regions in a level and then passing these predictions $\mathcal{B}_t$ to the next level, it can be helpful to train additional predictors, each of which updates the distributions for all regions in the same level, before proceeding to the next level in hierarchy. These multiple stages of predictions can help hone in on predicting the correct distribution of labels before being refined at the next level. Under this multiple prediction approach, $\mathcal{B}$ can be grown to consider the predictions on *all* regions from *all* previous predictors and Equation (6.10) can be augmented with an additional $c_{\text{prev}}(r, \mathcal{B})$ context feature that simply returns the label distribution for region r from the last predictor that operated on it. Finally, instead of traversing the hierarchy coarse-to-fine, it can be effective to first traverse fine-to-coarse to make initial predictions, and then refine these estimates coarse-to-fine. Similarly, Equation (6.10) can be augmented with a c_{children} descriptor that is analogous to the c_{parent} descriptor and is defined as the weighted average of the region's children predictions (weighted in proportion to the children's region sizes). Note that the sequence should always end on the leaf level to extract the sites' label distributions.

Algorithm 3 `train_inference_machine`

Inputs: Set of labeled images and respective region hierarchies $\mathcal{X}$, Traversal sequence $[t_1, \ldots, t_T]$.
$\mathcal{B} = \emptyset$ // Predictions from previous predictors that are used to derive context features
for $t = t_1 \ldots t_T$ **do**
    // Train test-time predictor using stacked predictions from previous stage
    Train P_t using $\mathcal{B}$ and regions $\mathcal{R}_{\ell(t)}(x), \forall x \in \mathcal{X}$

    // Create hold-out folds for stacking
    $[\mathcal{U}, \mathcal{V}] =$ `split_data`$(\mathcal{X})$ // $\mathcal{U} \cup \mathcal{V} = \mathcal{X}, \mathcal{U} \cap \mathcal{V} = \emptyset$
    Train P_U using $\mathcal{B}$ and regions $\mathcal{R}_{\ell(t)}(x), \forall x \in \mathcal{U}$
    Train P_V using $\mathcal{B}$ and regions $\mathcal{R}_{\ell(t)}(x), \forall x \in \mathcal{V}$

    // Generate stacked predictions for the next stage
    for $r \in \mathcal{R}_{\ell(t)}(x), x \in \mathcal{U}$ **do**
        $\mathcal{B} \leftarrow \mathcal{B} \oplus \{P_V(Y_r|x)\}$
    end for
    for $r \in \mathcal{R}_{\ell(t)}(x), x \in \mathcal{V}$ **do**
        $\mathcal{B} \leftarrow \mathcal{B} \oplus \{P_U(Y_r|x)\}$
    end for
end for
Return: Trained test-time predictors $\{P_t\}_{t=1}^{T}$

6.3.5 Training the Procedure

Training the predictors is prone to two problems with cascading errors. *First*, if we independently train each predictor using ground truth context features, we will have cascading errors at test-time due each predictor being trained with with perfect signal. Therefore, we have to train the hierarchical procedure in the same way it is executed at test-time: in sequence, using the previous predictor's output. *Second*, now using predictions from the same data used for training is prone to a cascade of errors due to overfitting as subsequent predictors will rely heavily on still optimistically correct context. While previous predictions are important, we also want to learn how to recover from mistakes that will be made at test time by trading off between the context from the previous predictions and image features. To achieve this robust training, we use the idea of stacking (Wolpert, 1992, Cohen and Carvalho, 2005) when training the classifier.

At its core, stacking implements the natural observation that when training sequential models, one should avoid using the predictions on the *same* data that was used to train the previous model. Instead, the predictions should be generated on data that was not used to train the model, *i.e.*, held-out predictions. In our context, we want to avoid training P_t using $\mathcal{B}_t$ as features if the same regions in $\mathcal{B}_t$ were used to train P_{t-1}. Obtaining held-out predictions is achieved in a manner similar to cross-validation

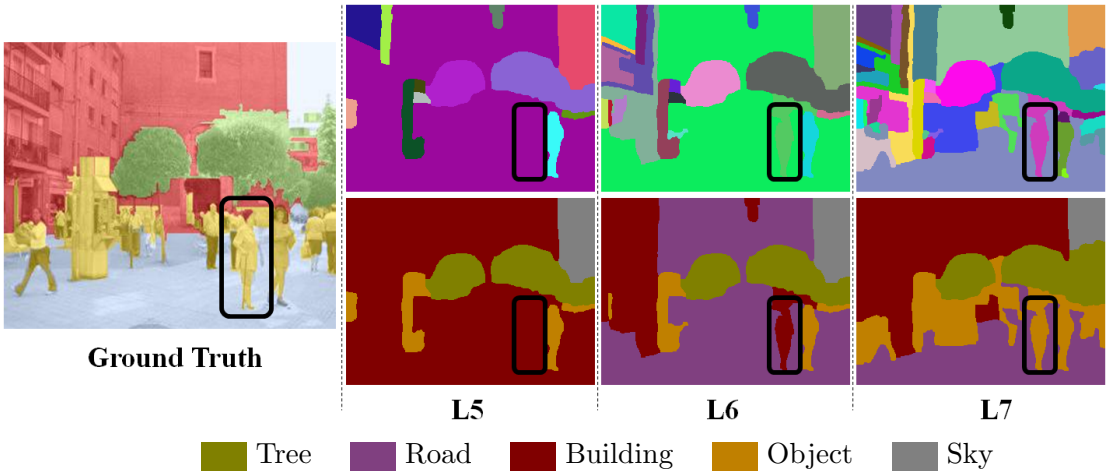


Figure 6.5: Learning to correct mistakes. Left: test image with ground truth overlaid. Top: levels 5, 6, 7 in the segmentation hierarchy. Bottom: the max label per region.

where the training data is split into *multiple* subsets that *multiple* models/predictors are respectively trained on. However, we do not use the subsets to do model selection, as would be done in cross-validation. Instead, a model that was trained on one subset of the data is evaluated on the held-out subset to generate predictions $\{b_r\}$ that are less overfit for the next predictor to train on. This stacking procedure is done solely at training-time to generate the predictions $\mathcal{B}_{t-1}$ for the next predictor to use. The model P_t that is used at test-time is trained over *all* the regions in B_{t-1} (and not used during training-time). An example training procedure using 2-fold stacking is summarized in Algorithm 3; however, in our experiments we use 10-folds.

Because the predictions are made on unseen data, the stacking procedure simulates the test-time behavior and ideally learns how to correct earlier mistakes. An example of this correcting behavior during test-time is illustrated in Figure 6.5. In the 5^{th} level of the hierarchy (L5), the person is part of a very large region for which building class has max probability. In L6, the person is segmented out from its large parent region; however, the most likely label for this region incorrectly follows from the parent's label (building). In L7, the region recovers from this error and correctly labels the object.

6.3.6 Inference

At test-time, the scene is parsed by traversing over the hierarchy in the same way that it was trained: at the t'th step, we use the previous set of predictions $\mathcal{B}_{t-1}$ to generate the augmented contextual feature reperesentation (Equation (6.10)) for each region, and then evaluate each region with model P_t (Section 6.3.2) to generate $\mathcal{B}_t$ for the next predictor. Hence parsing the scene is a simple sequence of T (MaxEnt) predictions.

6.4 Experimental Analysis on Images

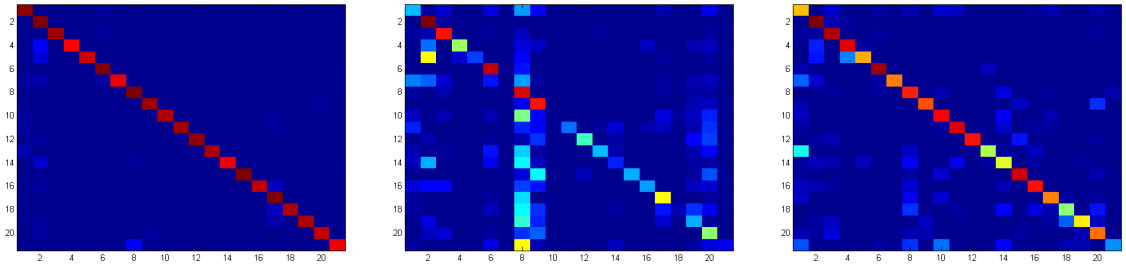
6.4.1 Initial Experiments

In this section, we present early analysis of the hierarchical inference machine (HIM) framework for scene parsing in images. In these experiments, we used 8-level and 9-level segmentation hierarchies that were constructed using the sophisticated segmentation technique of (Arbelaez et al., 2009). The inference procedure made one sequence of predictions from coarse-to-fine down the hierarchy, and we use boosting with per-class regression trees to project the functional gradient of the log-loss. The region features follow from the flat CRF model of (Gould et al., 2008b): we use RMoments over TXT, CIELAB, and RGB pixel values in a region (Section 3.3). We evaluate on the MSRC-21 (Shotton et al., 2009) and Stanford Background datasets and demonstrate favorable performance compared to a variety of other models at the time, most of which explicitly modeled the joint configuration of labels.

In Figure 6.6, we demonstrate the effect of stacking during the learning procedure. When training without stacking, we achieve very good performance on the training set, as indicated by a strong diagonal in the confusion matrix (Figure 6.6-a). However, when the resulting poor performance on the test-set confusion matrix demonstrates the learned predictors are overfit (Figure 6.6-b). Using stacking during training results in more general predictors (Figure 6.6-c).

In Table 6.1 and Table 6.2, we present quantitative comparisons of HIM with other techniques that use (unnormalized) graphical models (Zhang and Ji, 2010, Ladicky et al., 2009, Gould et al., 2009), hierarchical regions (Lim et al., 2009), and sequential prediction (over sites) (Tu and Bai, 2010). The baseline model Leaf is a predictor that is trained only over the leaf regions in the hierarchy without any context. In Figure 6.7, we quantify the pixel accuracy, by assigning each region's pixels its most probable label, at each stage of the inference procedure as the hierarchy is traversed from coarse (L1) to fine. In addition to performance that is competitive with and exceeds related work at the time, a major advantage of our approach is that inference is a simple sequence of MaxEnt predictions and is extremely efficient in practice. In these early experiments, holding segmentation and all feature computations constant, inference time took less than a second.

Another benefit over techniques that rely on MAP inference is is that we never make hard decisions and always predict a distribution of labels. Therefore, when eventually assigning a label to a region, we can extract a notion of confidence in the labeling. We define a labeling as *confident* when the difference between the first and second most probable labels is greater than 0.2, and otherwise *uncertain*. For example, in Figure 6.8,



(a) (b) (c)

Figure 6.6: The effects of stacking during the learning procedure. Confusion matrices on the MSRC-21 dataset: (a) performance on the training set without stacking, performance on the testing set (b) without and (c) with stacking.

Table 6.1: Performances on the MSRC-21 dataset. *Pixel* is the total number of pixels correct and *Class* is the mean across the columns. *Averaged over 5 different splits.

	Pixel	*Class*	Building	Grass	Tree	Cow	Sheep	Sky	Airplane	Water	Face	Car	Bicycle	Flower	Sign	Bird	Book	Chair	Road	Cat	Dog	Body	Boat
(Gould et al., 2008b)*	77	64	72	95	81	66	71	93	74	70	70	69	72	68	55	23	83	40	77	60	50	50	14
(Zhang and Ji, 2010)	75	65	77	93	70	58	64	92	57	70	61	69	67	74	70	47	80	53	73	53	56	47	40
(Lim et al., 2009)	–	67	30	71	69	68	64	84	88	58	77	82	91	90	82	34	93	74	31	56	54	54	49
(Tu and Bai, 2010)	75	69	69	96	87	78	80	95	83	67	84	70	79	47	61	30	80	45	78	68	52	67	27
(Ladicky et al., 2009)	86	75	80	96	86	74	87	99	74	87	86	87	82	97	95	30	86	31	95	51	69	66	9
LEAF	74	60	72	96	85	74	70	91	63	58	65	59	69	58	32	22	84	25	83	55	33	54	4
HIM	78	71	63	93	88	84	65	89	69	78	74	81	84	80	51	55	84	80	69	47	59	71	24

Table 6.2: Performances on STANFORD BACKGROUND dataset.

	Pixel	*Class*	Sky	Tree	Road	Grass	Water	Bldg.	Mtn.	Object
(Gould et al., 2009) Pixel CRF	74.3	66.6	93.9	67.1	90.3	83.3	55.4	71.4	9.3	62.2
(Gould et al., 2009) Region CRF	76.4	65.5	92.6	61.4	89.6	82.4	47.9	82.4	13.8	53.7
LEAF	72.8	58.0	89.7	58.3	85.8	69.8	15.8	78.1	1.5	64.9
HIM	76.9	66.2	91.6	66.3	86.7	83.0	59.8	78.4	5.0	63.5

the cars are *confident* in the labeling, but the trees in front of the building are *uncertain*. In MSRC-21, our *confident* predictions constitute 79% of the data and achieve an overall accuracy of 89%, while the accuracy of the *uncertain* samples is 37%. In STANFORD BACKGROUND, our *confident* predictions constitute 87% of the data and achieve an overall accuracy of 82%, while the accuracy of the *uncertain* samples is 40%. These numbers indicate that we make most errors when the labeling is *uncertain*.

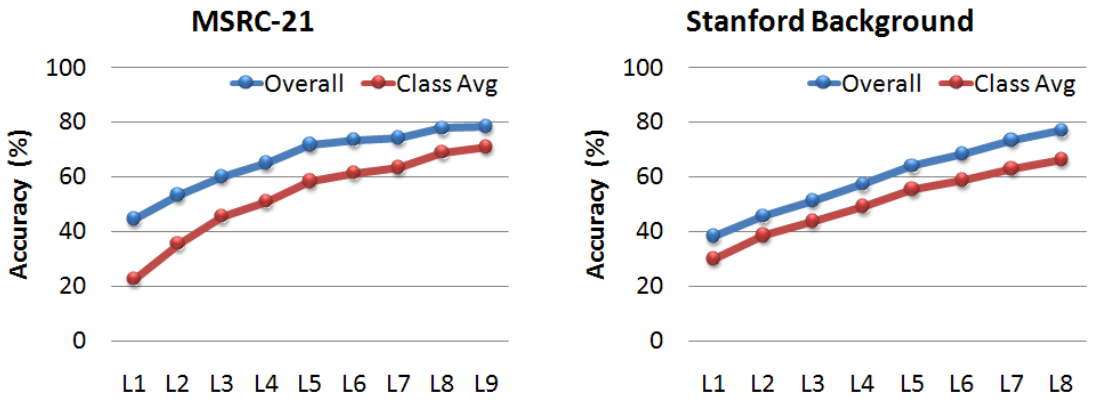


Figure 6.7: Per-pixel accuracies at each stage of the inference procedure.

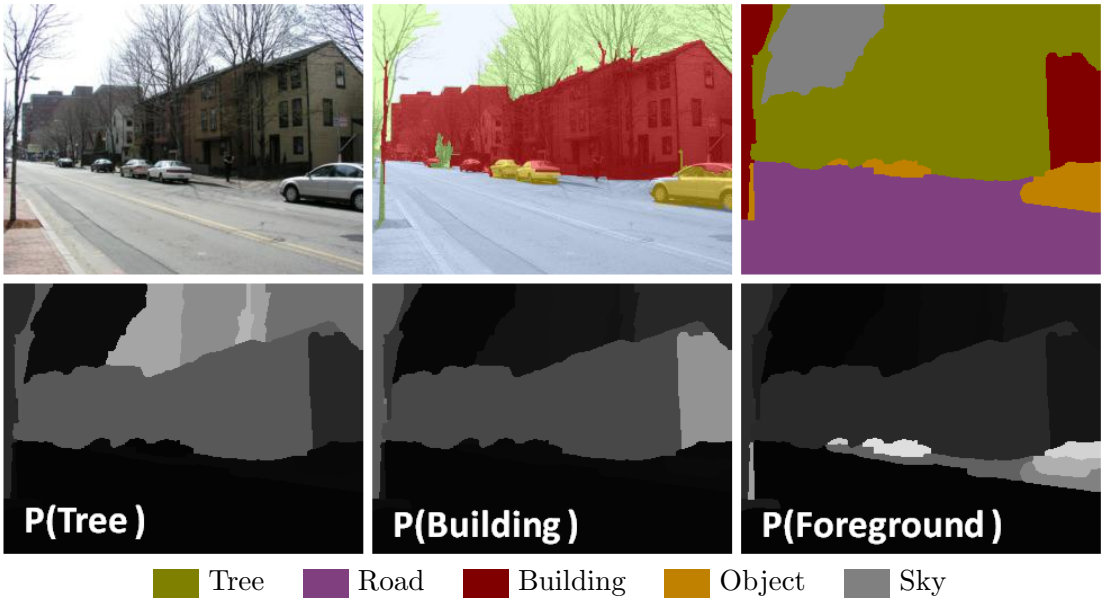


Figure 6.8: Predicting with uncertainty. The ambiguity in ground truth label (top-middle), which contains a tree in front of a building, is correctly modeled in our predicted probabilities (bottom row). The classification (*i.e.*, labels with highest predicted probablities) is given in the top-right.

6.4.2 Recent Experiments

Since the original publication of this work (Munoz et al., 2010b), we have made a number of practical improvements to achieve more efficient and accurate predictions. We found that different choices in feature descriptors can result in statistiscally different predictions. In particular, we found that average pooling over quantized features provides discriminative representation and is computationally efficient to compute over regions.

Table 6.3: Breakdown of HIM computations on the STANFORD BACKGROUND dataset.

	Segmentation	Features	Inference
Time (s)	0.095	0.462	0.037
Proportion	16%	78%	6%

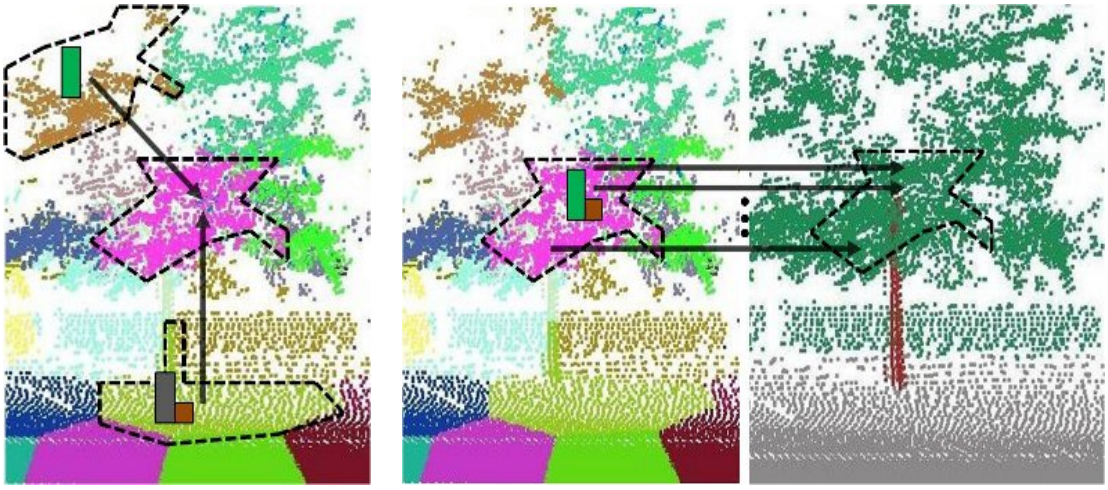
Table 6.4: Classification comparisons on the STANFORD BACKGROUND (top) and CAMVID (bottom) datasets. *Class* is the average class accuracy and *Pixel* is the per-pixel accuracy. †Uses temporal cues from the video stream. ‡Uses additional training data not leveraged by other techniques.

	Sky	Tree	Road	Grass	Water	Bldg.	Mtn.	Object	*Class*	*Pixel*
(Lempitsky et al., 2011)	-	-	-	-	-	-	-	-	72.4	81.9
(Socher et al., 2011)	-	-	-	-	-	-	-	-	-	78.1
(Ren et al., 2012)	95	76	92	87	68	85	29	63	74.5	82.9
(Farabet et al., 2013)	95.7	78.7	88.1	89.7	68.7	79.9	44.6	62.3	76.0	81.4
HIM-v2	92.5	76.9	90.2	81.5	70.4	82.1	16.1	68.9	71.8	81.6

	Bldg.	Tree	Sky	Car	Sign	Road	Ped.	Fence	Pole	Sdwlk.	Bike	*Class*	*Pixel*
(Brostow et al., 2008)†	46.2	61.9	89.7	68.6	42.9	89.5	53.6	46.6	0.7	60.5	22.5	53.0	69.1
(Sturgess et al., 2009)†	84.5	72.6	97.5	72.7	34.1	95.3	34.2	45.7	8.1	77.6	28.5	59.2	83.8
(Ladicky et al., 2010)‡	81.5	76.6	96.2	78.7	40.2	93.9	43.0	47.6	14.3	81.5	33.9	62.5	83.8
(de Nijs et al., 2012)†	59	75	93	84	45	90	53	27	0	55	21	54.7	75.0
HIM-v2	83.5	85.1	94.5	78.3	41.7	95.5	38.7	18.0	15.2	78.3	36.2	60.5	85.7

We now do separate average poolings (RCODES) over quantized CIELab, TXT and SIFT pixel descriptors (Section 3.1), in addition to RSHAPE2D. Additionally, we found the our procedure is robust to the choice of segmentation algorithm and noticed small variation in performance when using the sophisticated, but computationally expensive, segmentation algorithm from (Arbelaez et al., 2009) compared to using the seminal, and extremely efficient, graph-based F-H technique. On the learning side, we found that traversing fine-to-coarse-to-fine improves performance, and that using vector regression when projecting the functional gradient provides both feature sharing and computational efficiency.

In Table 6.3, we breakdown the computational costs for each step of the inference procedure when processing an image from the STANFORD BACKGROUND dataset. These timings were computed on a laptop with a Intel i7-2960XM, 4-core processor, and averaged over all images in a test fold. "Segmentation" is the time to construct all levels in the segmentation hierarchy. "Features" is the time to compute all region feature descrip-



(a) (b)

Figure 6.9: Adapting inference machines to 3-D point clouds. (a) A segmentation of the 3-D point cloud (from K-means). The pink segment receives predictions (contextual information) from its neighbors. (b) Contextual information is sent from the regions to its children "regions" that constitute it – in this example, the children regions are the 3-D points themselves and are colored by their class for visualization purposes.

tors, including neighborhood information for the contextual features. "Inference" is the time to make the sequence of T MaxEnt predictions along the hierarchy. From the table, we see that feature computation time vastly dominants our current inference procedure and making the actual predictions is only a small fraction of the total cost. In Table 6.4, we present updated classification performances on the Stanford Background and CamVid datasets. Many of the report techniques compute sophisticated feature descriptors and/or segmentation(s) whose computation time exceeds our total approach often by orders of magnititudes. In addition to achieve state-of-the-art performance, we stress the extreme simplicity and efficency of our approach.

6.5 Parsing 3-D Point Clouds with Inference Machines

At its core, the inference procedure requires two sources of information: 1) a representation of the scene to operate over, 2) a representation of the contextual information. Hence, only a few modifications need to be made to adapt the inference procedure to parse 3-D point clouds. First, we can use a variety of techniques to construct a hierarchical segmentation of the 3-D points, *e.g.*, K-means and F-H (Section 3.2), and compute 3-D descriptors (Section 3.3) for each region. Second, we can derive new context feature encodings of the predictions from neighboring regions in space. Figure 6.9 illustrates

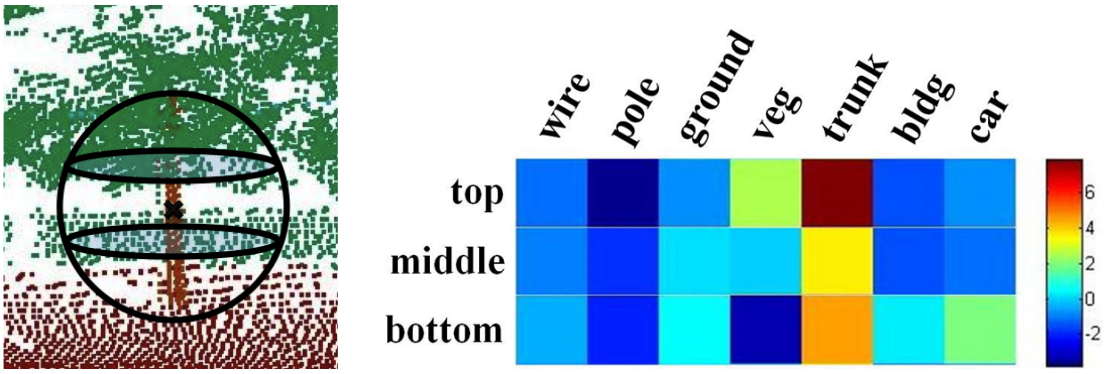
these adaptations to 3-D point clouds. Note that this representation operates over region entities and does not require an organized/lattice structure of the 3-D information, *e.g.*, a range/depth image. Hence, our approach is very useful in mobile robotics where the laser sensor returns an unorganized point cloud.

When parsing scenes, the most difficult classes to recognize are those that look the same as other classes at a local scale. In 3-D point clouds, a canonical example is the similarity among classes that are linear structures, *e.g.*, poles and tree-trunks. In the absence of any color/intensity readings from the laser (or other sensors), these two classes look geometrically the same for every 3-D point on the object. Hence, the only way to disambiguate these classes is with context, *e.g.*, vegetation typically appears "above" a tree-trunk. Effectively encoding this *a priori* knowledge into the feature descriptors on the 3-D points can be difficult due to the variation of object configurations in the world. For example, explicitly looking for vegetation at a fixed offset from a 3-D point can be problematic due to the various sizes of trees. Instead, it is better for the model to learn and propagate these interactions in a data-driven way. Note that these repulsive/attractive interactions between classes are much more expressive than the smoothing interactions we saw in Section 5.3.

In our experiments, we consider the relative spatial locations of the predictions using the contextual features. Analogously to the c_{local} context feature described in Section 6.3.3, we look at the neighboring predictions within a 3-D window/sphere around the region of interest. Because gravity is a discriminative cue, we spatially discretize the sphere into 3 cells by elevation, as illustrated in Figure 6.10-a. For each of the 3 cells, we average the neighboring predictions and concatenate them to form the contextual descriptor. After learning a linear model, we can inspect the parameters to see what was learned. For the weights corresponding to the tree-trunk class, $\theta_{\text{tree-trunk}}$, we observed that the model automatically learned sensible interactions, as illustrated in Figure 6.10-b. Here, we plot the weights corresponding to the 3 cells of the context feature. We see that the tree-trunk has high affinity for more tree-trunk points being above, below, and at the same level of a region, which corresponds to a smoothing behavior. However, we also see that affinity for vegetation being in the top slice (attraction) and negative weights for poles and vegetation being below the region (repulsion).

6.6 Experimental Analysis on 3-D Point Clouds

We analyze the HIM approach for scene parsing in 3-D point clouds. In these experiments, a simple 2-level hierarchy was constructed using K-means. The inference procedure makes multiple passes up and down the hierarchy, and we train each predictor using a parametric MaxEnt model. The region features compute GEOM, ORIENT,



(a) (b)

Figure 6.10: Learning context in 3-D point clouds. (a) Spatial discretization to encode neighboring context and (b) learned interactions between neighboring classes.

SpinImage, Elev descriptors (Section 3.1.2) centered at the centroid. We evaluate on the VMR Oakland-v2 and GML-PCV datasets (Section 2.2). On VMR Oakland-v2, we compare to the Associative Max-Margin Markov Networks (AMN) from Section 5.3, using the same feature representation. On GML-PCV, we compare to the non-associative version (NAMN), as described in (Shapovalov et al., 2010). For both datasets, we also report the MaxEnt predictions just using the leaves of the hierarchy without context (Leaf).

On the VMR Oakland-v2 dataset, HIM achieved a macro F_1 score of 0.76 while AMN and Leaf averaged 0.71 and 0.61, respectively. Qualitative results of AMN and HIM are shown in Figure 6.11. We can see a significant improvement on the points that are at the boundary of object regions using HIM. This can be explained by the way AMN represents interactions between points. The higher order potentials over regions prefer for the region to have a homogeneous label. In contrast, HIM supports heterogeneous label distributions and it can preserve the correct labeling at the boundary of regions. For example, in Figure 6.11-f the top border of a building is misclassified as vegetation using AMN model. This is because the points possess similar local features (scatter and high elevation) that resemble vegetation. Furthermore region-wise features do not disambiguate the problem and the high-order potential function will prefer to group these points as the same incorrect label. HIM solved this problem by learning the context of the regions in the top level. Such context becomes essential when local features and point context both failed to capture the correct label. Table 6.5 shows individual class performance of the three algorithms. As we expected, Leaf performs the worst of the three since it does not include any contextual information. The poor performance on tree-trunk, vehicle, and wire classes demonstrates the difficulty of this dataset.

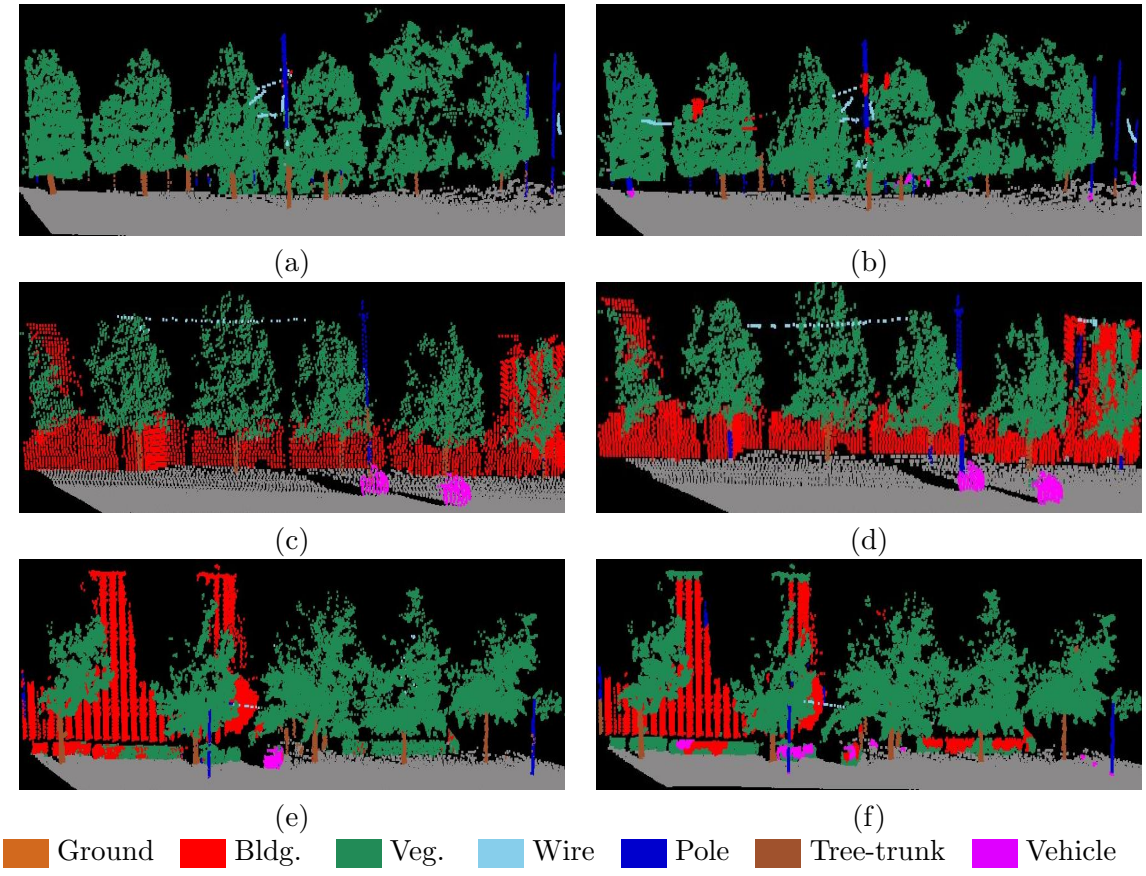


Figure 6.11: Point cloud classifications on the VMR Oakland-v2 dataset. Left column: HIM classification. Right column: AMN classification.

Table 6.5: Precisions (P) and recalls (R) on the VMR Oakland-v2 dataset.

		Wire	Pole	Ground	Veg.	Trunk	Bldg.	Veh.
P	AMN	0.66	**0.55**	**0.99**	0.94	0.55	0.80	0.70
	HIM	**0.73**	0.51	**0.99**	**0.96**	**0.65**	**0.83**	**0.79**
	Leaf	0.49	0.42	**0.99**	0.90	0.46	0.74	0.63
R	AMN	0.72	0.63	**0.99**	**0.94**	0.30	0.92	0.43
	HIM	**0.75**	**0.67**	0.98	0.93	**0.41**	**0.93**	**0.74**
	Leaf	0.52	0.53	**0.99**	0.90	0.13	0.87	0.38

On the GML-PCV dataset, HIM outscored NAMN 0.66 to 0.59 in Dataset A and 0.85 to 0.77 in Dataset B; Leaf averages 0.49 in A and 0.69 in B. Table 6.6 shows the per-class performance of the three algorithms. In Dataset A, all three algorithms perform poorly on vehicles and shrub classes. In contrast with the VMR Oakland-v2 dataset, few points constitute vehicles in this dataset. The shape information of small objects is almost lost in these long range aerial scans. Because of this,the vehicle and

Table 6.6: Precisions (P) and recalls (R) on the GML-PCV dataset.

	Dataset A	Ground	Bldg.	Tall-veg.	Shrub	Veh.
P	NAMN	0.90	0.87	0.92	**0.72**	0.37
	HIM	**0.95**	**0.91**	**0.99**	0.31	**0.54**
	LEAF	0.92	0.74	0.96	0.06	0.03
R	NAMN	0.96	0.58	**0.99**	0.09	**0.16**
	HIM	**0.98**	**0.77**	0.98	**0.36**	0.10
	LEAF	0.96	0.37	0.93	0.13	0.01
	Dataset B					
P	NAMN	**0.99**	**0.88**	0.95	0.25	
	HIM	**0.99**	0.83	**0.97**	**0.53**	
	LEAF	0.98	0.79	0.88	0.38	
R	NAMN	0.98	0.81	0.89	**0.57**	
	HIM	**0.99**	**0.92**	**0.97**	0.52	
	LEAF	**0.99**	0.63	0.96	0.10	

shrub classes share similar local features and no contextual information can be used to differentiate between the two. Without the ambiguity between vehicles and shrubs, we can see an improvement on shrub in Dataset B.

One key characteristic of this dataset is that the ground points are not on the same elevation, so the elevation of a point provides little information about its class. Due to such difficulty, LEAF has extremely poor performance on the shrub and vehicle classes. For example, LEAF cannot distinguish shrub from tall-vegetation because of their similar local features (Figure 6.12-f). HIM learns that shrubs have a high distribution ground in its neighborhood, while tall-vegetation does not and can fix this mistake (Figure 6.12-e).

Another challenging example is shown in (Figure 6.12-c). In this dataset, the roofs of buildings can extend for very large areas. Hence, the ELEV descriptor can no longer discrminate between buildings and the ground as the relative height is the same. HIM learns that building regions are above the neighboring ground regions and propagates this information. It corrects misclassified building points using this contextual information (Figure 6.12-b).

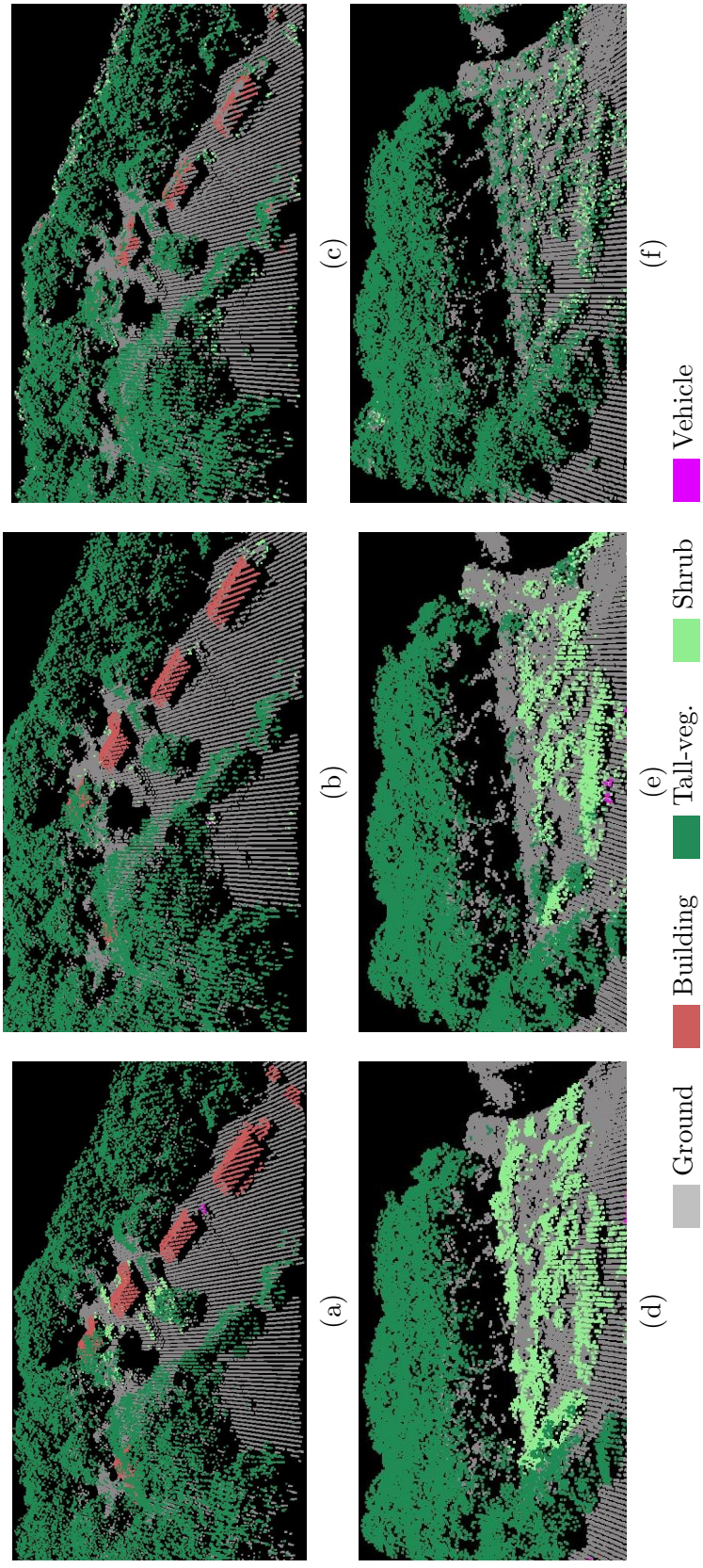


Figure 6.12: Point cloud classifications on the GML-PCV dataset. Left column: ground truth labeling. Middle column: HIM classification. Right column: LEAF classification.

Chapter 7

Co-inference Machines

7.1 Introduction

With the advent of an increasingly wide selection of sensing modalities (*e.g.*, optical cameras, stereo/depth cameras, laser scanners, flash ladar, sonar), it is now common to obtain multiple observations of a given scene. In general, however, the sensor observations from different modalities often do not uniquely correspond to each other. For example, 1) A laser scanner will never return any depth readings past a maximum range limit, while a camera can measure pixels infinitely far. 2) Range sensors, such as the X-Box Kinect, will often have missing depth information due to imperfect correspondences. 3) Scanning range sensors now commonly used on ground vehicles generate point clouds with highly variable point density in 3-D because of variations in depth and incidence angle coupled with complex scanning patterns. Further complicating matters is the fact that it is physically impossible for the two sensors to have the exact same viewpoint, and in practice the sensors are often physically far apart. As a consequence, objects are often visible in one sensor but occluded in the other(s).

In this chapter, we address these fundamental challenges that arise in scene parsing from multiple modalities. While our approach could be applied to multiple sensors, for clarity, we henceforth focus on parsing scenes from images and 3-D point clouds; however, our approach is not specific to this application and relies on general definitions and operators. In our application, we are given an image, a 3-D point cloud, and the camera parameters to project the 3-D points into the image plane. Our approach will simultaneously assign a semantic category (*e.g.*, building, car, *etc.*) to all elements in *both* domains, as illustrated in Figure 7.1. The main contribution of this work is a technique for performing simultaneous/co-inference across domains when there is *not* a unique correspondence between modalities. We evaluate the efficacy of our approach on our CMU Image+Laser dataset (Section 2.3.1).

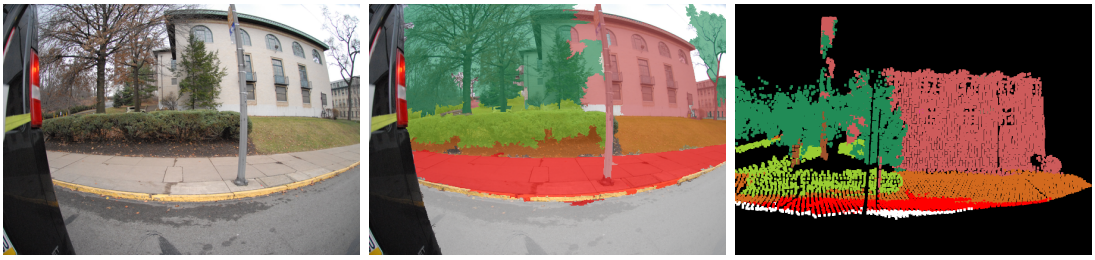


Figure 7.1: Multimodal scene parsing. The reference scene (left) is observed with a camera and laser scanner and simultaneously classified in the image (middle) and 3-D point cloud (right).

7.2 Background

7.2.1 Motivation

Two spatially adjacent scenes from the CMU Image+Laser dataset are shown in Figure 7.2 to highlight the challenges of this problem. Our CMU Image+Laser dataset was collected with a laser scanner and camera mounted on a vehicle driving in an urban environment. As the vehicle moves, the laser scanner continuously collects and maps the 3-D points to a global reference frame. Because the laser scanner operates in a push-broom mode, the displacement is often on the order of tens of meters between the location of the scanner when it observes a 3-D point versus the location of the corresponding camera(s) into which the 3-D point is projected. Hence, there are often multiple 3-D points of different objects along the ray of the camera's (occluded) viewpoint, *e.g.*, the building behind the trees. In addition, the laser scanner samples the scene at a much sparser rate, *i.e.*, we have many more pixels than number of points. Currently, many datasets with combined image and depth data are post-processed in order to obtain a full-resolution depth image (Liu et al., 2010, Silberman and Fergus, 2011, Janoch et al., 2011). While interpolation might work well under appropriate conditions, an accurate and complete interpolation is impossible in general, especially in outdoor environments (*e.g.*, there is no depth for pixels past the maximum range of the sensor, and the density of measured points in 3-D varies substantially).

7.2.2 Related Work

The problem of analyzing scenes in combined 2-D and 3-D data has been investigated early in the literature (Nitzan et al., 1977, Besl and Jain, 1986, Kweon et al., 1988);

Figure 7.2: Example images and point cloud from our CMU IMAGE+LASER dataset. The point cloud is colored by elevation. Colored circles are drawn to help the reader make correspondences between the domains.

however, the problem has received a considerable increase in attention due to the broad availibility of inexpensive sensors (Matthies et al., 2003, Dima et al., 2004, Douillard et al., 2011a). The conventional way to approach this problem is to constrain the representation into only one of the modalities while integrating information from the other discarded domain as features. That is, the approach can be *2-D driven* (Kweon et al., 1988, Dima et al., 2004, Baseski et al., 2007, Brostow et al., 2008, Gould et al., 2008a, Xiao and Quan, 2009, Zhang et al., 2010, Collet et al., 2011b, Silberman and Fergus, 2011), in that reasoning is done in the image while integrating 3-D features, or the approach can be *3-D driven* (Koppula et al., 2011, Tombari and Stefano, 2011, Douillard et al., 2011a, Lai et al., 2012), in that the predictions are made on the 3-D data while integrating 2-D features. These approaches are typically only applicable when the two modalities are in correspondence. In the commonly occurring case when there is a disparity between domains, constraining the modalities into a single representation can have negative consequences, as illustrated in Figure 7.3.

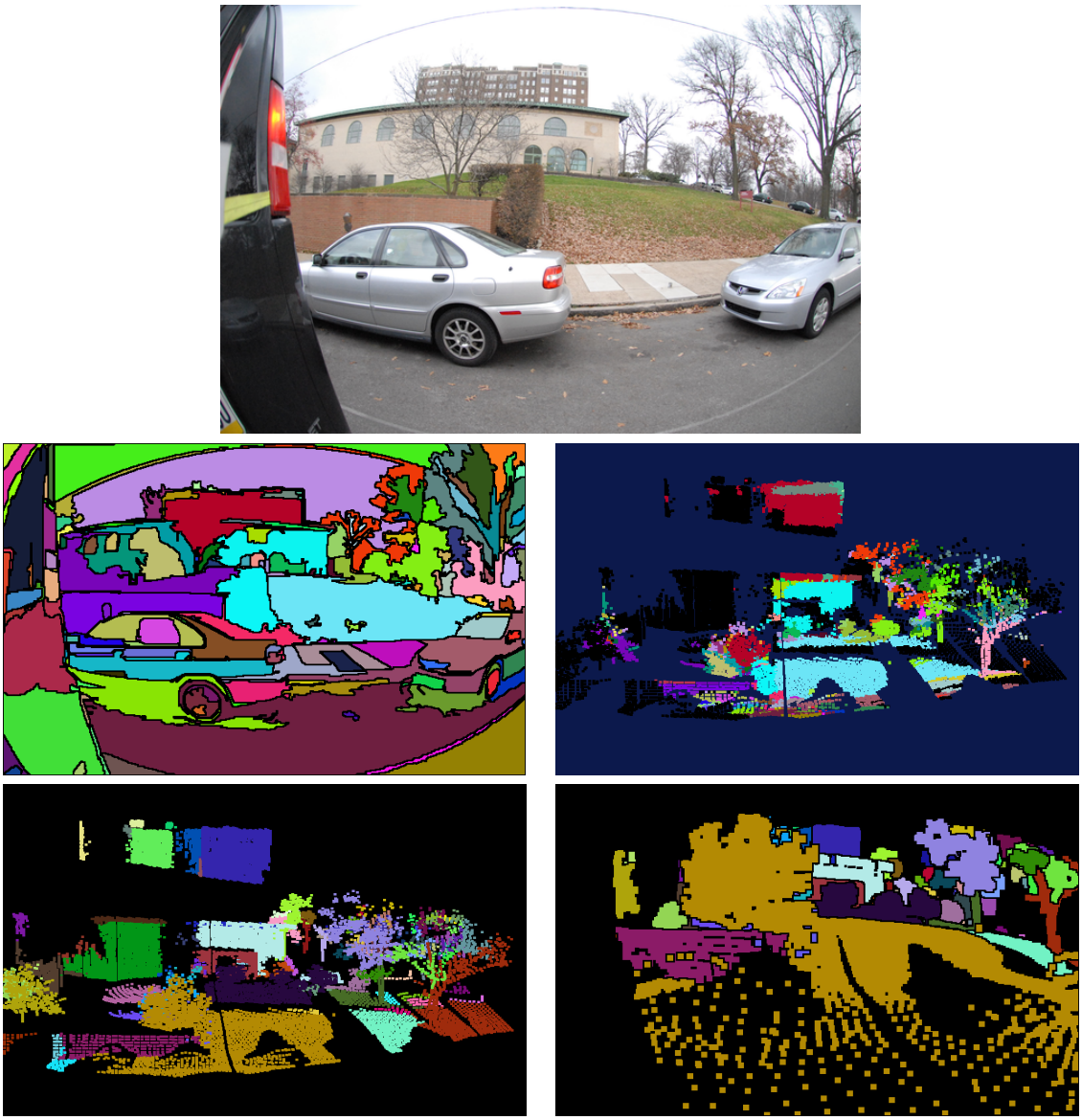


Figure 7.3: The effects of constraining the representation into a single domain. **Top** (a): Reference scene. **Middle** (b): 2-D driven approach. The image is segmented (left) and then back-projected into the 3-D point cloud (right) using occlusion reasoning. The 3-D region colors correspond to the 2-D segmentation, except the 3-D points colored black which are occluded with respect to the camera's viewpoint and are not associated with any 2-D region. **Bottom** (c): 3-D driven approach. The original 3-D point cloud is segmented (left) and then projected into the image plane (right) using occlusion reasoning. Note that not every pixel is associated with a 3-D region and that the the resulting 2-D regions are not connected due to occlusions, and sampling rates.

In the presence of this data mismatch, we instead propose to treat both modalities as first class objects, that is, we never discard data from either domain and we perform joint inference over all modalities. By coupling the inference over all modalities, we can propagate contextual information to and from data without correspondences, which would be discarded with the canonical approach, in order to aid predictions.

7.3 Reasoning with Multiple Modalities

7.3.1 Inference in One Modality

We wish to infer semantic labelings in both modalities simultaneously. In principle, we might define a single graphical model with edges linking nodes between modalities as well as high-order cliques over regions. Optimizing and learning parameters in such a graphical model is difficult because of the exponential number of label configurations and intractable structure. Instead, we build upon the hierarchical inference machine framework for scene parsing from the previous chapter (Chapter 6). To simplify the presentation of our multi-modal scene parsing approach, we first revisit and modify four notations used in Section 6.3.

First, we denote $\mathcal{X}_t$ to be the set of *regions* at level t (which was previously denoted as $\mathcal{R}_t(\cdot)$).

Second, we denote $q : \mathcal{X} \rightarrow \mathbb{R}^{|\mathcal{K}|}$ to be the MaxEnt probability distribution that predicts each class' probability for the region, *i.e.*,

$$b_{i,t}[k] = q_t(x_i)[k] \tag{7.1}$$

$$= P_t(Y_i = k|x_i), \tag{7.2}$$

where P_t is defined in Equation (6.3). Hence, the learning problem can be re-written as

$$\arg\min_{q_t} \sum_{x_i \in \mathcal{X}_t} D_{KL}(\tilde{b}_i || q_t(x_i)), \tag{7.3}$$

where D_{KL} is the KL divergence and $\tilde{b}_i$ is the ground truth, empirical label distribution for region x_i.

Third, we denote

$$B_t = \oplus_{\tau=1}^{t-1} \oplus_{x_i \in \mathcal{X}_\tau} \{b_{i,\tau}\}, \tag{7.4}$$

where $\oplus$ denotes the list concatenation operator, to be all previous predictions made over *all* regions in the hierarchy up to level t.

Fourth, we denote $\phi(x_i) \rightarrow \mathbb{R}^{d_1}$ to be the domain-specific features computed for region x_i, *e.g.*, the RSHAPE2D descriptor for images described in Section 3.3. Furthermore, we denote $\breve{\phi}_t(x_i, \mathcal{B}_{t-1}) \rightarrow \mathbb{R}^{d_1+d_2}$ to be the region features augmented with the contextual features, as described in Equation (6.10).

7.3.2 Co-inference in Multiple Modalities

We denote by $\mathcal{X}^{(1)}$ and $\mathcal{X}^{(2)}$ the set of regions in the hierarchical segmentations generated from two modalities, images and 3-D point clouds, respectively. A straightforward approach to analyze the modalities would be to construct two independent region hierarchies and to perform independent inference. However, instead of predicting over each domain separately, we want to couple the predictions so that information from one modality is propagated to the other. This is important because some domains are more apt at predicting certain categories than others. For example, as our experiments show, images are better for discriminating between physically similar things but with different texture (*e.g.*, road vs. sidewalk), and 3-D point clouds are better for semantically similar objects but at different scales (*e.g.*, buses vs cars). In order to use this inter-domain context, the predictors must incorporate this information at training-time. We now discuss how to modify the unimodal inference procedure to use information from multiple modalities.

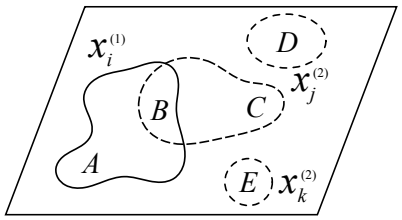
Inter-domain Co-neighborhoods

First, we need a notion of correspondence between regions in different domains. We define an inter-domain co-neighborhood function $\eta_j : \mathcal{X}^{(i)} \rightarrow \wp(\mathcal{X}^{(j)})$, where $\wp$ is the power set operator. Given a region in one domain, this function simply returns a (potentially empty) set of neighboring regions in the other domain; we refer to this set of corresponding neighbors in the other domain as *co-neighbors*.

As previously discussed for our application, it would be unwise to directly use pixel and 3-D point correspondences; instead, we use the following approach. For each 3-D region in the 3-D segmentation $\mathcal{X}^{(2)}$, we project its points into the image plane, using z-buffering to maintain closest-to-camera ordering, resulting in a (partial) projected 2-D segmentation. Now, for any 3-D region $x^{(2)} \in \mathcal{X}^{(2)}$, $\eta_1(x^{(2)})$ returns all the 2-D regions that the projected segmentation of $x^{(2)}$ touches in the 2-D segmentation, and for any 2-D region $x^{(1)} \in \mathcal{X}^{(1)}$, $\eta_2(x^{(1)})$ returns all 3-D regions that $x^{(1)}$ touches in the projected segmentation. Figure 7.4 illustrates our co-neighborhoods.

Inter-domain Overlap

Next, we need a notion of how much a region in one modality should influence a region in the other. We define an inter-domain overlap function $\nu : \mathcal{X}^{(i)} \times \mathcal{X}^{(j)} \rightarrow \mathbb{R}^{+}$, which assigns a non-negative value indicating a degree of correspondence between two regions in different modalities. We use the intersections of regions in the projected 3-D segmentation and the 2-D segmentation to define this overlap. Figure 7.4 illustrates inter-domain overlap.



$$\begin{aligned} \eta_2(x_i^{(1)}) &= x_j^{(2)} \\ \eta_1(x_k^{(2)}) &= \emptyset \\ \nu(x_i^{(1)}, x_j^{(2)}) &= B \\ |x_i^{(1)}| &= A + B \\ |x_j^{(2)}| &= B + C + D \end{aligned}$$

Figure 7.4: Synthetic example of inter-domain co-neighborhoods and overlaps. The solid outline is the only 2-D region $x_i^{(1)}$, and the dashed outlines are 2-D projections of the 3-D regions $x_j^{(2)}$ and $x_k^{(2)}$; note that the projection of $x_j^{(2)}$ is not simply connected.

Inter-domain Context Features

Using the above definitions, we define the fixed-length, inter-domain context feature function $h_t^{(i,j)} : \mathcal{X}^{(i)} \times \mathcal{B}^{(j)} \rightarrow \mathbb{R}^{K+1}$, which, for a given region in one domain, computes a contextual feature vector using its co-neighboring K-class predictions in the other domain. Formally,

$$h_t^{(i,j)}(x_k^{(i)}, B_t^{(j)}) = \sum_{x_l^{(j)} \in \eta_j(x_k^{(i)})} \frac{\nu(x_k^{(i)}, x_l^{(j)})}{|x_k^{(i)}|} \left[b_{l,t-1}^{(j)}, 1 \right]^T, \tag{7.5}$$

where $|x|$ is the area of the (projected) region as used in ν. In words, the first K values of this vector are the weighted average of the predictions of the co-neighboring regions in the other domain, where the weight is based on inter-domain overlap; and the last value is in $[0, 1]$ and is the fraction of overlap with the co-neighboring region(s). It is 0 when the first K values are 0, which happens when a region is observed in only one modality, and it is 1 when the first K values sum to 1, which happens when a region fully overlaps with co-neighbor region(s). This value is needed to disambiguate how much a region should trust its co-neighbors' predictions. For example, a co-context feature value of 0.2 could be due to high predicted probability and low overlap, or *vice versa*.

Putting It Together

Given two hierarchical segmentations and a procedure for propagating information between regions in the different modalities, we can now jointly train the entire procedure. For simplicity in the explanation, we assume that the two hierarchies have the same number of levels. We train two sets of predictors $\{q_t^{(1)}\}, \{q_t^{(2)}\}$, one set for each hierarchy. Instead of training all the predictors for one domain first before starting to train the other, we instead train pairs of predictors at a time as we iterate over the levels.

Algorithm 4 `train_co_inference`

1: **Inputs:** Labeled region hierarchies over N different modalities $\{\mathcal{X}^{(i)}\}_{i=1}^{N}$, Traversal sequence $[t_1, \ldots, t_T]$.
2: $Q^{(i)} = \emptyset, \forall i$ // predictors for each modality
3: $B = \emptyset, \forall i$ // predictions over all regions, in all domains encountered so far
4: **for** $t = t_1 \ldots t_T$ **do**
5: $\quad B_t = \emptyset$
6: $\quad$ **for** $i = 1 \ldots N$ **do**
7: $\quad\quad q_t^{(i)} =$ `train_predictor`$(\mathcal{X}_t^{(i)}, B)$ // Solve Equation (7.3) using Equation (7.6)
8: $\quad\quad Q^{(i)} \leftarrow Q^{(i)} \oplus \{q_t^{(i)}\}$ // Save for test-time
9: $\quad\quad [\mathcal{U}, \mathcal{V}] =$ `split_data`$(\mathcal{X}_t^{(i)})$ // $\mathcal{U} \cup \mathcal{V} = \mathcal{X}_t^{(i)}, \mathcal{U} \cap \mathcal{V} = \emptyset$
10: $\quad\quad q_U =$ `train_predictor`$(\mathcal{U}, B)$
11: $\quad\quad q_V =$ `train_predictor`$(\mathcal{V}, B)$
12: $\quad\quad$ **for** $x \in \mathcal{U}$ **do**
13: $\quad\quad\quad B_t \leftarrow B_t \oplus \{q_V(x)\}$
14: $\quad\quad$ **end for**
15: $\quad\quad$ **for** $x \in \mathcal{V}$ **do**
16: $\quad\quad\quad B_t \leftarrow B_t \oplus \{q_U(x)\}$
17: $\quad\quad$ **end for**
18: $\quad$ **end for**
19: $\quad B \leftarrow B \oplus B_t$ // Use the stacked predictions to couple the modalities
20: **end for**
21: **Return:** Trained test-time predictors for each modality $\{Q^{(i)}\}_{i=1}^{N}$

That is, we first train $q_{t-1}^{(1)}$ and $q_{t-1}^{(2)}$ before training $q_t^{(1)}$ and $q_t^{(2)}$. In order to couple the predictions and propagate context across domains, we augment our feature representation with the respective co-neighbors' predictions. That is, when training $q_t^{(i)}$ over the regions $x^{(i)} \in \mathcal{X}_t^{(i)}$, we use the fixed-length region feature representation

$$\breve{\phi}_t^{(i)}(x^{(i)}, B_t^{(j)}) = [\breve{\phi}_t^{(i)}(x^{(i)}) \; ; \; h_t^{(i,j)}(x^{(i)}, B_t^{(j)})] \in \mathbb{R}^{d_1+d_2+K+1}, \tag{7.6}$$

where $\breve{\phi}_t^{(i)}(x^{(i)}) \in \mathbb{R}^{d_1+d_2}$ is the feature representation augmented with context features derived only from the previous predictions in modality i. Using the features computed from $\breve{\phi}_t^{(i)}$ couples the contextual information from the other modality j's previous predictions when training $q_t^{(i)}$. Algorithm 4 summarizes the training procedure in the simplest case of one example/region hierarchy (observed with N modalities) and using 2-fold stacking (Wolpert, 1992); it is implied that each region x_i is associated with its ground truth empirical distribution $\tilde{b}_i$. The test-time inference follows similarly, except we replace lines 7-17 with $B_t \leftarrow B_t \oplus \{q_t^{(i)}(x)\}, \forall x \in \mathcal{X}_t^{(i)}$, where $q_t^{(i)} = Q^{(i)}[t]$.

Although the presentation has focused on the image and point cloud setting, the general definitions of η and ν can be applied to any multi-modality scenario for which there is an operational definition of the projection from one modality to another, which,

in order to leverage information, must exist. For example, co-neighborhoods can be defined between samples that correspond to the same physical space (*e.g.*, in images and infrared) and/or time (*e.g.*, in audio and video). The key benefit of our approach is that we eliminate the constraint of requiring a unique correspondence between domains and that we can pass information in a softer manner through contextual features.

7.4 Experimental Analysis

7.4.1 Models

Given an image and a point cloud, our approach returns a complete labeling of both modalities simultaneously. We compare this approach with the natural baselines of using one modality in isolation and with augmented features computed in the other modality. That is, we compare with the single-domain representations of the state-of-the-art hierarchical inference framework, analyzed in Chapter 6. Controlling for the same hierarchical representation, features, and predictors facilitates a fair comparison among six possible models:

1. **2D**: Hierarchical segmentation and features are computed only in the image (Section 6.3). No 3-D data can be classified.

2. **2D+A**: Hierarchical segmentation and features are computed in the image. In addition, the 2-D regions are back-projected into the point cloud (Figure 7.3(b)) and 3-D features are computed over these 3-D regions and appended to the feature descriptor. No 3-D data is classified with this model.

3. **3D**: Hierarchical segmentation and features are computed only in the point cloud (Section 6.5). No 2-D data can be classified.

4. **3D+A**: Hierarchical segmentation and features are computed in the point cloud. In addition, the 3-D regions are projected into the image (Figure 7.3(c)) and 2-D features are computed over these 2-D regions and appended to the feature descriptor. No 2-D data is classified with this model.

5. **Co**: Our proposed approach. Two hierarchical segmentations are separately constructed in the image and point cloud, with the same features computed over the regions as in **2D** and **3D**, respectively.

6. **Co+A**: Same as **Co**, but with each region's features augmented across domains as done in **2D+A** and **3D+A**

7.4.2 Inference Machine Setup

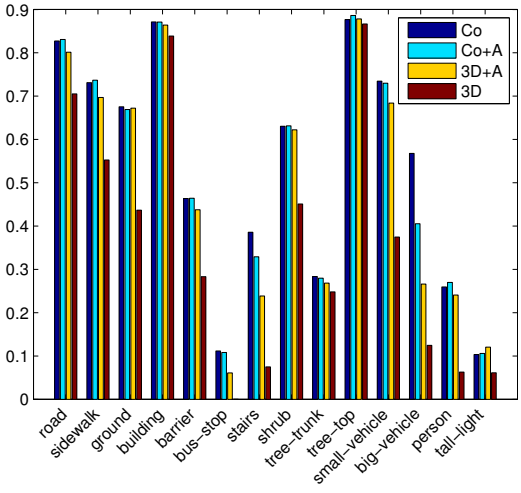
All of the models require 1) a hierarchical segmentation, 2) region features, and 3) MaxEnt predictors. We instantiate the (co-)inference machines using the specifications similar to those used in the experiments of Chapter 6. We use the F-H (Felzenszwalb and Huttenlocher, 2004) algorithm to create 4-level segmentation hierarchies in images and 3-D point clouds, as described in Section 3.2. For the 2-D region features, we compute the RSHAPE2D descriptor and averaged RCODES (Section 3.3) over quantized per-pixel TXT, LBP, I-SIFT, and C-SIFT descriptors (Section 3.1.1). For the 3-D region features, we compute the RSHAPE3D region-ELEV, region-GEOM descriptors and averaged RCODES (Section 3.3) over quantized per-point SPINIMAGE and combined [GEOM , ORIENT] descriptors (Section 3.1.2). Furthermore, we purposely do *not* use the point's distance from the laser sensor to help reduce dataset bias of observing scenes from a vehicle on the road. Finally, we use boosting (Section 4.4) to optimize Equation (7.3) where the weak learners are vector regression trees, and are sequentially trained using 10-fold stacking (Wolpert, 1992). We iterate over the hierarchy from bottom $\rightarrow$ top $\rightarrow$ bottom with the sequence: $[\ell_1, \ell_1, \ell_2, \ell_2, \ell_3, \ell_3, \ell_4, \ell_4, \ell_3, \ell_3, \ell_2, \ell_2, \ell_1, \ell_1]$, where ℓ_1, ℓ_4 are the leaf and root levels, respectively.

7.4.3 Analysis

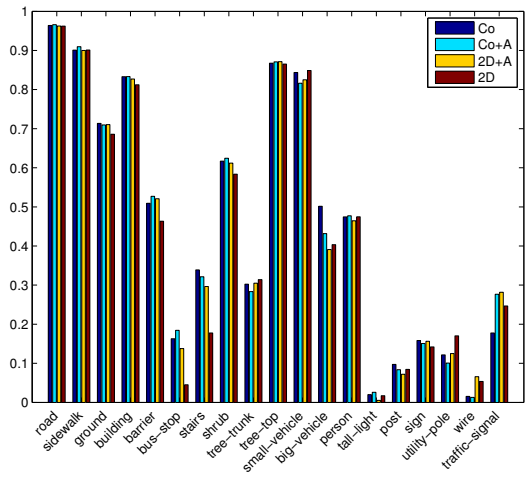
We evaluate on the CMU IMAGE+LASER dataset (Section 2.3.1). As the models defined only over a single modality cannot make predictions on the other, we evaluate the performance on the points and pixels that correspond so that the comparisons between **Co** vs. **2D**/**3D** are consistent. However, note that **Co** will make predictions over the entire image and point cloud. As there is a severe (and unavoidable) imbalance in the number of samples per class, we evaluate the the per-class F_1 score, computed separately over pixels/voxels in each domain.

In Figure 7.5, we present performance for each of the 6 models on the 3-D point clouds and images. We immediately see that feature augmentation in both domains is beneficial, especially in the 3-D point cloud. This result is expected as texture can help disambiguate among road, sidewalk, and ground in 3-D. Next, we see that in both domains **Co** $\geq$ **Co+A**, indicating that the information from the other domains can be encoded as our contextual features without a loss of representation power and avoids overfitting due to a larger, augmented feature representation. This is important as it simplifies the representation and computation time, *i.e.*, we do not need to duplicate the feature computation.

Figure 7.5-c shows an improvement in F_1 on all except one rare class in the 3-D point clouds. This improvement is due to the robustness of the representation: 1) There



(a)



(b)

Label	**Co**	**3D+A**	Diff.
Road	.827	.802	**.**026
Sidewalk	.731	.697	**.**034
Barrier	.464	.438	**.**026
Bus-stop	.112	.061	**.**051
Stairs	.386	.239	**.**147
Tree-trunk	.284	.268	.015
Small-vehicle	.735	.684	**.**051
Big-vehicle	.568	.266	**.**302
Person	.260	.241	.019
Tall-light	.103	.120	-.017

(c)

Label	**Co**	**2D+A**	Diff.
Barrier	.509	.521	-.012
Bus-stop	.163	.138	**.**025
Stairs	.339	.297	**.**042
Small-vehicle	.844	.825	.019
Big-vehicle	.502	.391	**.**111
Person	.474	.465	.010
Tall-light	.020	.005	.015
Post	.097	.072	**.**025
Wire	.015	.066	**-**.050
Traffic-signal	.178	.282	**-**.104

(d)

Figure 7.5: Per-class F_1 scores on our Image+Laser dataset. (a) Comparisons on the 3-D point clouds. (b) Comparisons on the images. Categories from (a) and (b) with at least a difference of 0.01 in F_1 are show in (c) and (d), respectively; differences of at least 0.02 are bolded. Categories not shown in (a) and (b) achieved 0.0 F_1 for all methods.

is bound to be back-projection errors when converting the 3-D point cloud into a 2-D segmentation from which 2-D features are computed. With the co-inference approach, we are more robust to these errors due to passing information as a distribution of labels, rather that encoding information in a large feature descriptor for which the spatial support could be poor. 2) As there is more image data than point cloud data, co-inference is indirectly passing larger amounts of global information to the 3-D point cloud, which is unavailable to **3D+A**. For example, the image component of **Co** examines the global context of all regions in the image, some of which might not have 3-D data. 3) As **Co**

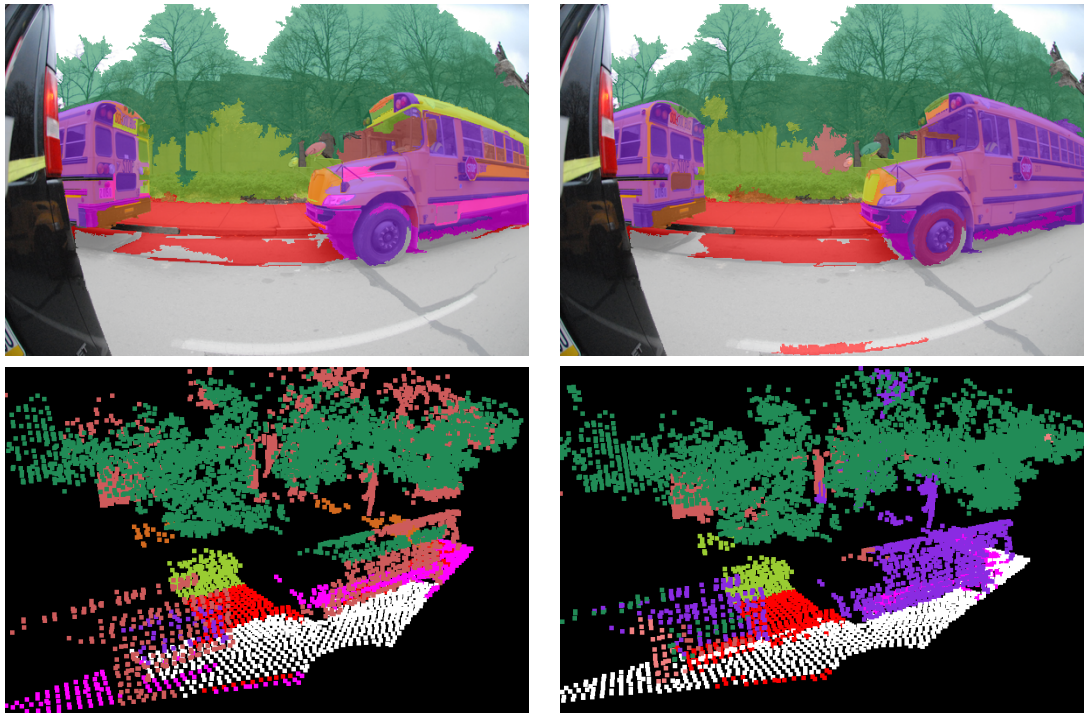


Figure 7.6: Qualtitative comparisons of multi-model parsings **2D+A** (a-top), **3D+A** (a-bottom) vs. **Co** (b). The proposed co-inference approach does a much better job of identifying the big-vehicles (buses).

does not augment the features across domains, its feature dimension is smaller and less susceptible to overfitting (for **Co/3D**, $\phi^{(2)}(x) \in \mathbb{R}^{98}$ and for **3D+A**, $\phi^{(2)}(x) \in \mathbb{R}^{693}$).

For images, Figure 7.5-d shows a big gain in the big-vehicle class, modest improvements in 3 other classes and slightly better overall. The large improvement in the big-vehicle class can be explained through Figure 7.6. For 2-D regions on the bus, corresponding 3-D regions have a large planar structure, similar to buildings for which they are often confused. Hence, simply augmenting the 3-D geometric features is not enough to disambiguate. By simultaneously reasoning in 3-D space, correct context can be propagated back into the image. Furthermore, we improve upon the vegetation that occlude each other. Figure 7.5 also shows a decrease in performance in the wire and traffic-signal classes because they are particularly hard classes to discriminate in 3-D point cloud data. As these two classes are physically very small and constitute a small fraction of the dataset, *none* of the 3-D models are currently able to detect them. Hence, since they cannot be discerned in the 3-D point cloud, co-inference cannot provide correct context

Table 7.1: Comparison of average (co-)inferences times per scene. Segmentation and feature computation time is held constant.

	Co	**2D+A**	**3D+A**	**2D**	**3D**
Time (s)	0.463	0.452	0.385	0.421	0.123

for these classes. Note that the remaining classes which the 3-D models can predict are improved upon, on average.

In addition to achieving improved performance and complete understanding in both domains simultaneously, co-inference is more efficient in practice. Table 7.1 summarizes the inference times for processing the entire scene, *i.e.*, the entire image and point cloud, with the different models. On average, **Co** takes 0.46 s to classify the entire scene whereas using **2D+A** and **3D+A** takes 0.45 s + 0.39 s = 0.84 s. Furthermore, from a practical viewpoint, co-inference is simpler to implement than feature augmentation due to the special cases which must be accounted for, *e.g.*, the special case when a region is observed in only one modality and the features cannot be computed for it in the other modality.

7.5 Summary

This chapter addresses the problem of parsing scenes from multiple modalities when there is not a unique correspondence between data points across modalities. Instead of restricting our representation to a single modality and integrating information from the unselected ones, we treat both modalities as first class objects and propose a joint inference procedure that couples the predictions among all of the modalities. Our experiments demonstrate that our co-inference approach obtains improved predictions in all modalities compared to multiple, decoupled representations with the added benefit of efficiency and simplicity.

Chapter 8

Temporal Consistency in Streaming Video

8.1 Introduction

In this chapter, we address the problem of generating spatially and temporally consistent predictions from streaming video, as illustrated in Figure 8.1, that would be seen from a moving platform. Simply applying a scene parsing algorithm to each image independently is not sufficient because it does not properly enforce consistent predictions over time. In practice, the temporally inconsistent predictions result in "flickering" classifications. This effect is not solely due to the motion of the camera through the 3-D scene: we often observe this behavior even on a sequence of images from a *static* scene due to subtle illumination changes. These inconsistencies in predictions can have a major impact on robotic tasks in practice, *e.g.*, predicted obstacles may suddenly appear in front of the robot in one frame and then vanish in the next. The situation is further complicated by the need for online/causal algorithms in robotics applications, in which the system does not have access to future frames, unlike video interpretation systems which can proceed in batch mode by using all the available frames either through a spatio-temporal graphical model or segmentation of the video.

Inspired by early work in robotics using linear filters (Giachetti et al., 1998), we consider a simple, causal filtering technique for maintaining temporally consistent predictions. Our approach is a meta-algorithm in the sense that it is agnostic to the specific way in which predictions are generated, so that it can be used with any per-frame scene analysis technique. Our only requirement is that the per-frame scene analysis technique predicts a per-pixel probability distribution over semantic labels, instead of a single label.

Our algorithm is illustrated in Figure 8.2. At the current frame $I^{(t)}$, each pixel i is associated with a label probability distribution $y_i^{(t)}$, which is produced by a scene analysis

Figure 8.1: Parsing scenes from video. Classified videos from this work are available at http://www.cs.cmu.edu/~dmunoz/.

algorithm. Our goal is to ensure that the final label distribution that we return for pixel i is consistent with the temporal prediction $\hat{y}_j^{(t-1)}$ from its corresponding pixel j in the previous frame $I^{(t-1)}$, which we do not know. Hence, we use optical flow (Werlberger et al., 2009) to estimate a neighborhood of candidate correspondences in the previous frame. Giving all neighbors equal weight and defining the smoothed prediction based on the average of the neighborhood's predictions is unwise because the neighborhood could include pixels of completely different objects. Therefore, between pixels $i \in I^{(t)}$ and $j \in I^{(t-1)}$, we propose to *learn* a data-driven, visual similarity function to assign a high weight w_{ij} between pixels that are likely to correspond to each other (and low weight for those that are not) in order to select correct correspondences and accurately propagate predictions over time.

Before discussing how predictions are combined between two frames, in Section 8.3, we first demonstrate the importance of using a data-driven function for measuring visual similarity between candidate pixels and present an efficient algorithm to learn this similarity function. In Section 8.4, we discuss how candidate pixels between frames are generated and how to combine the previous frame's predictions using the learned similarity function. In Section 8.5, we validate our proposed method over three distinct semantic labeling algorithms on three different datasets. Our experiments confirm that this natural approach yields temporally consistent predictions, with the additional important benefits of being very efficient and simple to implement.

Figure 8.2: Temporal consistency overview. (a) Frame $I^{(t-1)}$. (b) Frame $I^{(t)}$. The distribution of labels at a pixel in frame $I^{(t)}$ is combined with a weighted average of the distributions of labels in a neighborhood of pixels (the 3×3 orange grid) in the previous frame $I^{(t-1)}$. This neighborhood is initialized using optical flow techniques (c). We propagate predictions across time by *learning* a similarity function between pixels $i \in I^{(t)}$ (e) and $j \in I^{(t-1)}$ (d). This similarity assigns high values w_{ij} between visually similar pixels (green cells) and low values over visually different pixels (red cells).

8.2 Background

One popular way to incorporate temporal information is to compute Structure from Motion (SfM) between consecutive frames in order to compute geometric/motion-based features (Brostow et al., 2008, Sturgess et al., 2009, Micusik et al., 2012). One drawback of this approach is that accurate SfM computation may be slow or may require a large buffer of previous frames to process. Alternatively, or in addition, a large graphical model can be defined among multiple frames, where edges between frames propagate predictions over time (Wojek et al., 2010, Ess et al., 2009, Xiao and Quan, 2009, de Nijs et al., 2012, Badrinarayanan et al., 2010, Chen and Corso, 2011). Performing bounded, approximate inference over such large models remains a challenging problem. Furthermore, in order to efficiently compute approximate solutions, only an estimate of the MAP distribution is returned, *i.e.*, there is no uncertainty in the labeling or marginal distributions. To further improve efficiency in practice, techniques make further approximations at the cost of loss of guarantees on the solution quality. By returning label probabilities, our approach may be more useful as input for subsequent robotic algorithms, such as reasoning about multiple interpretation hypotheses. Another technique for maintaining temporal consistency, which is similar to defining a spatio-temporal graphical model, is to analyze volumes from a spatio-temporal segmentation (Grundmann et al., 2010, Brendel and Todorovic, 2011, Xu et al., 2012). This batch approach is omniscient in the sense that it requires processing the entire video sequence, which is typically not suitable for most robotic applications.

8.3 Learning Similarity

8.3.1 Metric Learning

In order to selectively propagate predictions from the previous frame, we assign high weight between pixels that are visually similar. One standard way to measure similarity w_{ij} between two pixels is through a radial basis kernel

$$w_{ij} = \exp\left(-\frac{d(f_i, f_j)}{\sigma^2}\right), \tag{8.1}$$

where $f_i \in \mathbb{R}^d$ is the vector of visual features of pixel i, $\sigma = 0.4$ controls the bandwidth of the kernel, and $d : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$ is a distance function. Only using the pixels' RGB values for the feature representation is not discriminative enough to correctly match correspondences. We instead augment RGB values with the responses from 17 gradient filters and a 11×11 local binary pattern window, resulting in a feature descriptor $f_i \in$

$\mathbb{R}^{140}$. Because of the increase in dimensionality, the standard squared Euclidean distance

$$d(f_i, f_j) = (f_i - f_j)^T (f_i - f_j) = \mathrm{Tr}(\Delta_{ij}), \tag{8.2}$$

where $\Delta_{ij} = (f_i - f_j)(f_i - f_j)^T$, is typically large between most samples in the feature space and is not discriminative between different types of objects, as illustrated in Figure 8.3-a,b. Alternatively, we can learn a distance that has the desirable properties for our application. That is, for a pair of pixels (i, j) from the set of pairs of pixels that truly correspond to each other, $\mathcal{E}_p$, we want $d(f_i, f_j)$ to be small, and for a pair of pixels (i, k) from the set that do not correspond, $\mathcal{E}_n$, we want $d(f_i, f_k)$ to be large. Learning a distance, or metric, also remains an active area of research (Xing et al., 2003, Davis et al., 2007, Goldberger et al., 2004, Hadsell et al., 2006, Weinberger and Saul, 2009, Chechik et al., 2010), and a variety of techniques could be used to learn d. As we are concerned with efficiency in the predictions, we use a simple Mahalanobis distance

$$d_M(f_i, f_j) = (f_i - f_j)^T M (f_i - f_j) = \mathrm{Tr}(M^T \Delta_{ij}), \tag{8.3}$$

and propose an efficient method to learn the parameters M offline from training data.

We follow the max-margin learning strategy and learn a metric such that the distances between pixels that do *not* correspond $(i, k) \in \mathcal{E}_n$ are larger by a margin than the distances between pixels that do correspond $(i, j) \in \mathcal{E}_p$. This can be formulated as the convex, semidefinite program

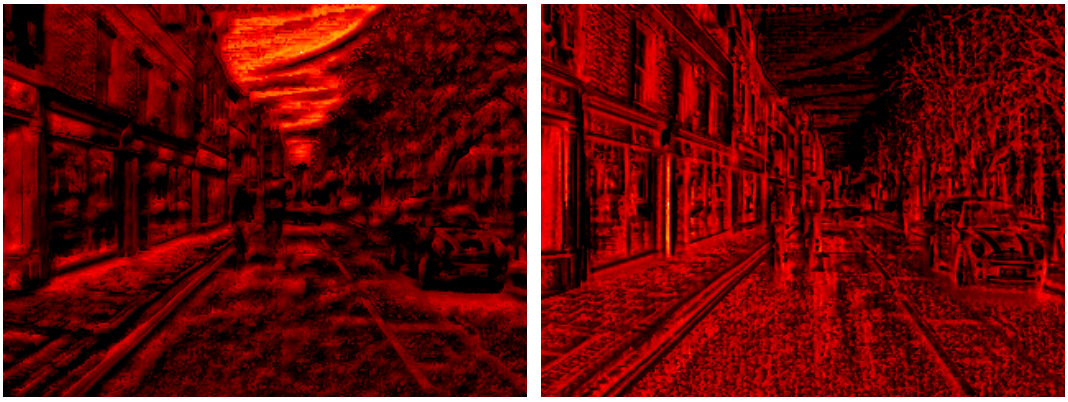
$$\begin{aligned} \min_{M,\xi,\zeta} \quad & \|M\|_F^2 + \alpha \sum_{(i,j)\in\mathcal{E}_p} \xi_{ij} + \beta \sum_{(i,k)\in\mathcal{E}_n} \zeta_{ik} \\ \text{s.t.} \quad & d_M(f_i, f_j) \le 1 + \xi_{ij}, \qquad \forall (i,j) \in \mathcal{E}_p \\ & d_M(f_i, f_k) \ge 2 + \zeta_{ik}, \qquad \forall (i,k) \in \mathcal{E}_n \\ & M \in \mathcal{M}, \end{aligned} \tag{8.4}$$

where $\mathcal{M} = \{M | M \succeq 0, M = M^T\}$ is the convex cone of symmetric positive-semidefinite matrices, and α and β penalize violating the margins. This program can be equivalently rewritten as the unconstrained, convex minimization

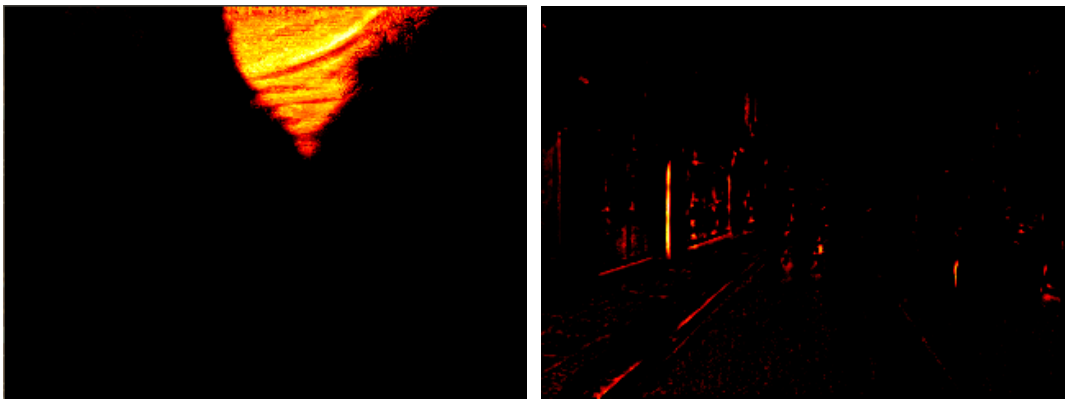
$$\begin{aligned} \min_{M\in\mathcal{M}} \mathrm{Tr}(M^T M) + \alpha & \sum_{(i,j)\in\mathcal{E}_p} \max(0, \mathrm{Tr}(M^T \Delta_{ij}) - 1) \\ & + \beta \sum_{(i,k)\in\mathcal{E}_n} \max(0, 2 - \mathrm{Tr}(M^T \Delta_{ik})), \end{aligned} \tag{8.5}$$

and can be efficiently optimized using the projected subgradient method (Ratliff et al., 2007).

(a)



(b)



(c)

Figure 8.3: Comparing similarity metrics. (a) A scene with two pixels of interest selected. (b) The inverted heatmaps of Euclidean distances of the respective pixel of interest to every other pixel in the image (bright means small distance). (c) The inverted heatmaps from our learned Mahalanobis metric.

We define Υ_{ij} and Ψ_{ik} to be subgradients of the respective summands:

$$\Upsilon_{ij} = \begin{cases} \Delta_{ij}, & \text{Tr}(M^T \Delta_{ij}) - 1 > 0 \\ 0, & \text{otherwise,} \end{cases}, \tag{8.6}$$

$$\Psi_{ik} = \begin{cases} -\Delta_{ik}, & 2 - \text{Tr}(M^T \Delta_{ik}) > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{8.7}$$

The subgradient update, with small step-size η_t, is then

$$M_{t+1} \leftarrow \mathcal{P}_{\mathcal{M}}(M_t - \eta_t(M_t + \alpha \sum_{(i,j)\in\mathcal{E}_p} \Upsilon_{ij} + \beta \sum_{(i,k)\in\mathcal{E}_n} \Psi_{ik})), \tag{8.8}$$

where $\mathcal{P}_{\mathcal{M}}$ projects to the closest matrix on the convex cone $\mathcal{M}$. Since Υ_{ij} and Ψ_{ik} are symmetric by construction, the closest projection, with respect to the Frobenius norm, is computed by reconstructing the matrix with its negative eigenvalues set to zero (Golub and Van Loan, 1996). To further improve run-time efficiency, we also constrain M to be a diagonal matrix.

As illustrated in Figure 8.3-c, our learned metric measures visual similarity much better than Euclidean distance. Although the computed distances may not be optimal across the entire image, we observe correct behavior over a local area. Thus, we use optical flow to initialize the area in which to compute distances over.

8.3.2 Obtaining Training Data

Learning the metric requires annotated examples of pairs of pixels that do and do not correspond. One way to generate these examples is to use the annotated images and sample pairs of pixels that belong to the same semantic category to create $\mathcal{E}_p$ and use pairs between the different categories to create $\mathcal{E}_n$. The result will be general metric for measuring similarity between generic object categories, which is a much harder problem than measuring similarities between instances of paired pixel correspondences. Furthermore, our similarity metric should work well between correspondences under different viewpoints, as this is the mode of operation while the robot is moving.

We instead generate our pairs of *training* data using pixel correspondences across multiple frames and viewpoints. These correspondences can be easily obtained through a variety of different keypoint-based algorithms. Specifically, we use the publicly available SfM package, Bundler (Snavely et al., 2006). Given a collection of images, Bundler produces a 3-D reconstruction of the scene (which we ignore) as well as the corresponding pixels across multiple frames. As illustrated in Figure 8.4, we use pairs of pixels from the same correspondence to construct $\mathcal{E}_p$ and use pairs that do not correspond to construct $\mathcal{E}_n$ when learning the metric *offline*. In addition to hard margin constraints, it may be

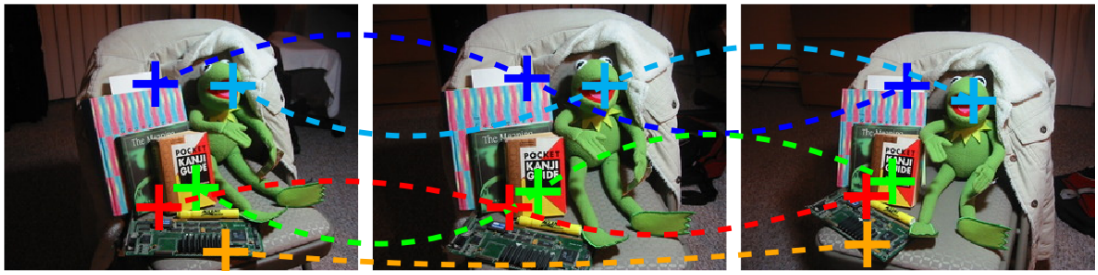


Figure 8.4: Generating data for training the metric. Pairs of pixels of true correspondences (along the same colored line) form $\mathcal{E}_p$ and pairs of pixels that do not correspond (between different colors lines) form $\mathcal{E}_n$.

beneficial to also incorporate (convex) *relative* margin constraints over triplets of pixels i, j, k

$$d_M(f_i, f_j) \leq d_M(f_i, f_k) + \delta_{ijk}, \tag{8.9}$$

where $\delta_{ijk} \in \mathbb{R}^+$ encodes the relative benefit (*e.g.*, distance) of true correspondences (i, j) over incorrect correspondences (i, k). These relative constraints also enable easy access to additional training data which can be used to learn more powerful metrics in higher dimensional spaces; we leave this for future work. Now that we can accurately measure visual similarity, in the following section we describe how to combine predictions over time.

8.4 Temporal Consistency

8.4.1 Optical Flow for Data Propagation

We are interested in general scene analysis without making strong assumptions on the movement/frame-rate of the camera and/or the type of object motions in the scene. Furthermore, we require the procedure to be as efficient as possible so it can potentially be used onboard a mobile robot. To generate initial hypothesis between frames, we use the efficient Anisotropic Huber-L_1 dense optical flow algorithm from (Werlberger et al., 2009) to obtain flow fields[1] $\overleftarrow{V}_x, \overleftarrow{V}_y$ that project pixel coordinates from $(u^{(t)}, v^{(t)}) \in I^{(t)}$ to $(u^{(t-1)}, v^{(t-1)}) \in I^{(t-1)}$ via

$$\begin{bmatrix} u^{(t-1)} \\ v^{(t-1)} \end{bmatrix} = \begin{bmatrix} u^{(t)} \\ v^{(t)} \end{bmatrix} + \begin{bmatrix} \overleftarrow{V}_x(u^{(t)}, v^{(t)}) \\ \overleftarrow{V}_y(u^{(t)}, v^{(t)}) \end{bmatrix}. \tag{8.10}$$

Although we use a state-of-the-art optical flow algorithm, it is not perfect and some velocity vectors may not match exactly and/or some correspondences between frames

[1]Here, the subscripts x, y are overloaded to indicate direction and are not related to features nor labels.

Algorithm 5 Causal Temporal Smoothing

1: **Inputs:** Metric M, previous frame $I^{(t-1)}$ with its temporally smoothed predictions $\hat{y}^{(t-1)}$, and current frame $I^{(t)}$ with its independent predictions $y^{(t)}$.
2: Compute $\overrightarrow{V}$ and $\overleftarrow{V}$ between $I^{(t-1)}$ and $I^{(t)}$ using (Werlberger et al., 2009)
3: Use $\overleftarrow{V}$ to warp $I^{(t-1)} \rightarrow \tilde{I}^{(t)}$ and $\hat{y}^{(t-1)} \rightarrow \tilde{y}^{(t)}$, except for invalid locations as defined by Equation (8.11)
4: Compute per-pixel features for $\tilde{I}^{(t)}$ and $I^{(t)}$
5: **for** $i \in I^{(t)}$ **do**
6: Define $\hat{y}_i^{(t)}$ using Equation (8.12) (and, implicitly, Equations 8.1 and 8.3)
7: **end for**
8: **return** Temporally smoothed predictions $\hat{y}^{(t)}$

might be missing. To help recover from missing flow information, we warp/transfer *small patches* of data at a time, instead of a single pixel. That is, for each pixel $(u^{(t)}, v^{(t)}) \in I^{(t)}$ and its correspondence $(u^{(t-1)}, v^{(t-1)}) \in I^{(t-1)}$, we use the forward-in-time flow vector $[-\overleftarrow{V}_x(u^{(t)}, v^{(t)}), -\overleftarrow{V}_y(u^{(t)}, v^{(t)})]^T$ to transfer *both* the RGB values and temporally smoothed predictions, $\hat{y}^{(t-1)}$, of each pixel in a 5×5 patch centered at $(u^{(t-1)}, v^{(t-1)})$. For each warped pixel, we accumulate RGB values and previous temporal predictions into the respective coordinates. After all pixels from the previous frame have been warped, the RGB values and predictions are appropriately averaged by the number of projections that fell into each coordinate, resulting in a warped RGB image $\tilde{I}^{(t)}$ and warped predictions $\tilde{y}^{(t)}$ into the current time t.

Estimated optical flows are imperfect, and in these scenarios we do not want to propagate label predictions over time. A standard method to detect optical flow failures is to ensure that the flows forward and backward in time are consistent. We consider the flow at pixel $(u^{(t)}, v^{(t)})$ to be invalid if

$$\left\| \begin{bmatrix} \overrightarrow{V}_x(u^{(t-1)}, v^{(t-1)}) \\ \overrightarrow{V}_y(u^{(t-1)}, v^{(t-1)}) \end{bmatrix} + \begin{bmatrix} \overleftarrow{V}_x(u^{(t)}, v^{(t)}) \\ \overleftarrow{V}_y(u^{(t)}, v^{(t)}) \end{bmatrix} \right\|_2 \geq \kappa, \tag{8.11}$$

where $\overrightarrow{V}$ is the flow from $I^{(t-1)}$ to $I^{(t)}$ and $\kappa = 13$ is a small threshold which is related to the size of the neighborhood used in the update described in the following subsection.

8.4.2 Causal Temporal Smoothing

Now we have all the necessary components for recursive temporal smoothing: a metric for measuring visual similarity between two pixels, w_{ij}, and a method to warp pixels between frames. At the recursive step of our procedure, we have the warped RGB image, $\tilde{I}^{(t)}$, its warped temporally smoothed predictions, $\tilde{y}^{(t)}$, and the predicted per-pixel probabilities from the scene analysis algorithm $\hat{y}^{(t)}$ for current frame $I^{(t)}$. For each pixel $i \in I^{(t)}$ and $j \in \tilde{I}^{(t)}$, we compute the pixel features (RGB, texture gradients, local binary pattern)

f_i, f_j that were used to learn our metric. Using an exponential smoothing update, we define a pixel's new temporally smoothed predictions using the update rule

$$\hat{y}_i^{(t)} = \frac{1}{Z_i}\left(\sum_{j \in N_i^{(t)}} w_{ij}\tilde{y}_j^{(t)} + cy_i^{(t)}\right), \tag{8.12}$$

where $N_i^{(t)}$ is a 5×5 spatial neighborhood in $I^{(t)}$ centered at pixel i, $c = 0.25$ is our prior belief on the independent prediction from the scene analysis algorithm, and

$$Z_i = \sum_{j \in N_i^{(t)}} w_{ij} + c, \tag{8.13}$$

ensures that $\hat{y}_i^{(t)}$ sums to one. The procedure then repeats to smooth predictions $y^{(t+1)}$ using $\hat{y}^{(t)}$. The entire procedure is summarized in Algorithm 5.

8.5 Experimental Analysis

We evaluate our approach over three sophisticated scene analysis algorithms over three different datasets (Chapter 2). All results were obtained using the same smoothing parameters across the different algorithms/datasets, and classified videos from each dataset are available at http://www.cs.cmu.edu/~dmunoz/.

8.5.1 Algorithms and Datasets

Firstly, we analyze the CamVid dataset where the per-frame predictions come from HIM (Chapter 6), which does not use any temporal information or additional detectors and is comparable (Sturgess et al., 2009) or exceeds (Brostow et al., 2008) other techniques which use these additional sources of information. For evaluating temporal consistency, we trained two separate models. The first is trained on the standard CamVid training fold and then evaluated on the test sequence 05VD. The second model is evaluated on the 16E5 sequence and trained on the remaining images not from this sequence.

Secondly, we analyze the NYU Scenes dataset where the per-frame predictions come from a deep learning architecture (Farabet et al., 2013), which were provided by the authors. This algorithm learns features using a multi-scale convolutional neural network and then performs inference by selecting labels based on a purity criterion over regions in a hierarchical segmentation.

Thirdly, we analyze the MPIVehicleScenes dataset where the per-frame predictions come from a per-pixel, boosting classifier (Wojek and Schiele, 2008), which were provided by the authors and is based on JointBoost (Torralba et al., 2007).

Table 8.1: Breakdown of computation time for temporal smoothing.

Computation	Time (s)
Optical flows (GPU)	0.02
Smoothing (CPU)	0.75
Total	0.77

Table 8.2: Per-class F_1 scores and accuracy on CamVid

	05VD		16E5	
Class	**Per-frame**	**Temporal**	**Per-frame**	**Temporal**
sky	.303	**.682**	.237	**.639**
tree	.352	**.563**	.336	**.518**
road	.197	**.546**	.150	**.529**
sidewalk	.277	**.512**	.188	**.357**
building	.165	**.232**	.275	**.395**
car	.127	**.456**	.304	**.597**
pole	.201	**.386**	.252	**.285**
person	.138	**.218**	**.324**	.193
bicycle	**.323**	.165	**.348**	.043
fence	.325	**.712**	.335	**.668**
sign	**.367**	.029	**.378**	.039
Accuracy	.261	**.591**	.286	**.530**

8.5.2 Efficiency

There are two main components of our approach: 1) forward and backward, dense optical flow computation and 2) temporal smoothing (which includes warping, pixel features, and the weighted averaging). The average computation times between two frames are shown in Table 8.1. The experiments were performed using a GeForce GTX 590 GPU and an Intel X5670 CPU. We observe that dense optical flow computation time can be brought down from the order of seconds with using a CPU to 10s of milliseconds with using a GPU. As our approach relies on many, simple numeric computations, we would expect a similar efficiency improvement with a GPU implementation.

In practice, the computational bottleneck is often from the scene analysis algorithm, depending on how expressive the features and/or model are. For example, both of the structured prediction algorithms take about one second to process a frame, which also includes feature computation. However, we demonstrate with the third dataset that we can maintain temporally consistent predictions with a simple per-pixel classifier.

Table 8.3: Per-class F_1 scores and accuracy on NYU Scenes

Class	Per-frame	Temporal
building	.231	**.547**
car	.207	**.630**
door	**.046**	.000
person	.094	.080
pole	.169	.139
road	.152	**.575**
sidewalk	**.373**	.274
sign	**.127**	.000
sky	.002	.019
tree	.353	**.630**
window	.102	.101
Accuracy	.209	**.500**

Table 8.4: Per-class F_1 scores and accuracy on MPIVehicleScenes

Class	Per-frame	Temporal
background	.420	**.730**
road	**.503**	.321
lane-marking	**.779**	.319
vehicle	.075	**.276**
sky	.206	**.571**
Accuracy	.407	**.533**

8.5.3 Analysis

Videos comparing per-frame and the temporally smoothed classifications for the sequences are available in the supplementary multimedia attachment; qualitative examples from each sequence are shown in Figure 8.5, Figure 8.6, and Figure 8.7. The videos demonstrate the substantial benefit of using temporal smoothing, especially on the CamVid sequences which are captured at a much higher frame rate.

It is important to remember that our approach relies on the output of the inner scene analysis algorithm and cannot fix misclassifications due to the biases of the base algorithm. Hence, for quantitative evaluation we first only consider pixels from which the prediction obtained by the scene analysis algorithm differs with our temporal smoothing. We compute a confusion matrix over these differing pixels and report the per-class F_1 scores as well as the per-pixel accuracy in Table 8.2, Table 8.3, and Table 8.4; improvements greater than 0.03 are bolded. This evaluation measures whether we are worse or better off with using temporal smoothing.

The behavior across datasets is consistent: categories which occupy large areas of the

Table 8.5: Overall pixel accuracies (%)

Dataset	Independent	Smoothed
CamVid-05VD	84.60	**86.85**
CamVid-16E5	87.37	**88.84**
NYU Scenes	71.11	**75.31**
MPIVehicleScenes	93.10	**93.55**

image (*e.g.*, sky, trees, cars, buildings) are significantly improved upon and predictions on some of the smaller objects (*e.g.*, signs, people, lane-markings) are sometimes over-smoothed. There are various reasons as to why performance may decrease. 1) Optical flow estimation on small objects may fail due to large displacements and/or excessive blurring, resulting in neighborhoods that are not adequately initialized. 2) As these objects occupy a small number of pixels, over-smoothing from spatially adjacent objects will cause a large drop in performance. Similarly, it is challenging to accurately annotated these intricate objects, and imperfections in the ground truth can further skew evaluation. 3) These small object categories are typically challenging to classify. When the predicted label distributions from the scene analysis algorithm are less confident, *i.e.*, have high entropy, the resulting weighted combination may be incorrect. Nonetheless, the overall improvement in accuracy shows a clear benefit of using temporal smoothing rather than per-frame classification.

The comparisons of overall per-pixel accuracy for each sequence are shown in Table 8.5. Due to the sparseness of the CamVid annotations, the quantitative improvements are not as drastic as we would expect, however, there is a noticeable gain. We also observe a large quantitative improvement in the NYU Scenes sequence, even in the presence of large camera motion. The improvement in the MPIVehicleScenes dataset is modest, however, this can be attributed to a small label set of 5 categories (vs. 11 and 33 from the other two) which often have little confusion. Furthermore, we note the predictions are qualitatively much smoother in appearance, even from using a per-pixel classifier.

8.6 Summary

This chapter proposes an efficient meta-algorithm for the problem of spatio-temporal consistent 2-D scene parsing from streaming video. Our approach is based on recursive weighted filtering in a small neighborhood, where large displacements are captured by dense optical flow, and we propose an efficient algorithm to learn image-based similarities between pixels. As we do not require information about future frames, our causal algorithm can handle streaming images in a very efficient manner. Furthermore, we

demonstrate that our approach can be wrapped around various structured prediction algorithms to improve predictions without a difficult redefinition of an inference process.

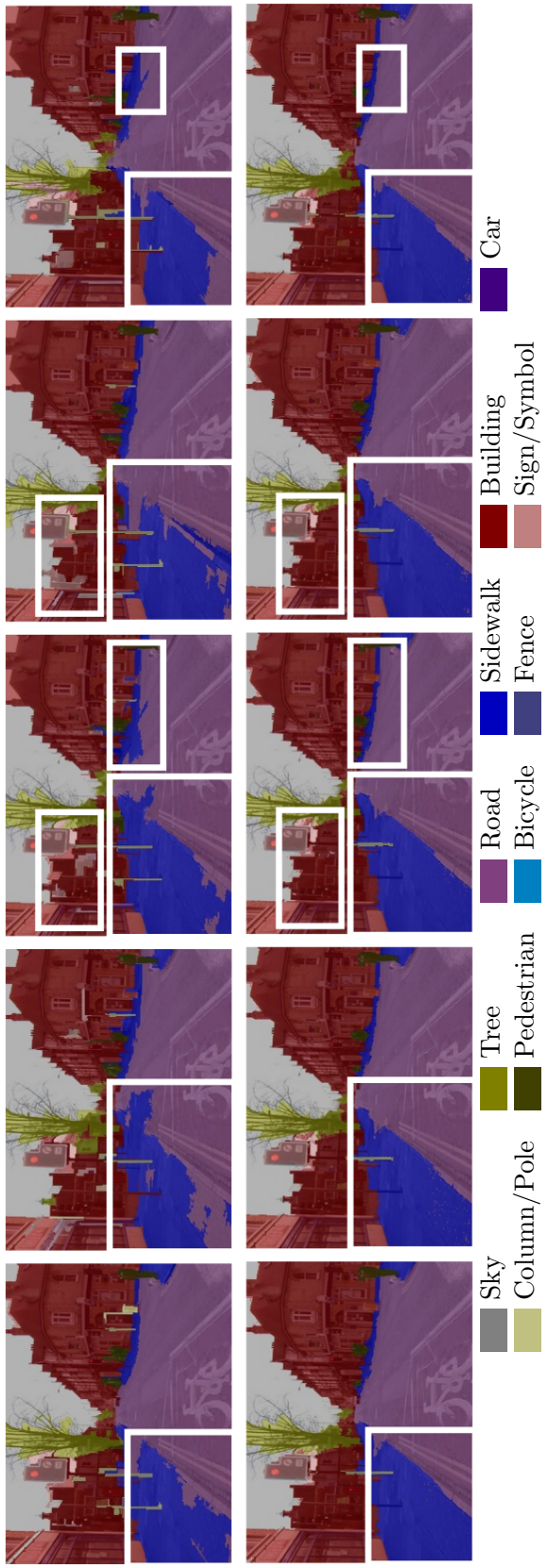


Figure 8.5: Temporal classifications on CamVid-05VD frames 1872-1876. Top: per-frame. Bottom: temporally smoothed. Inconsistent predictions are highlighted.

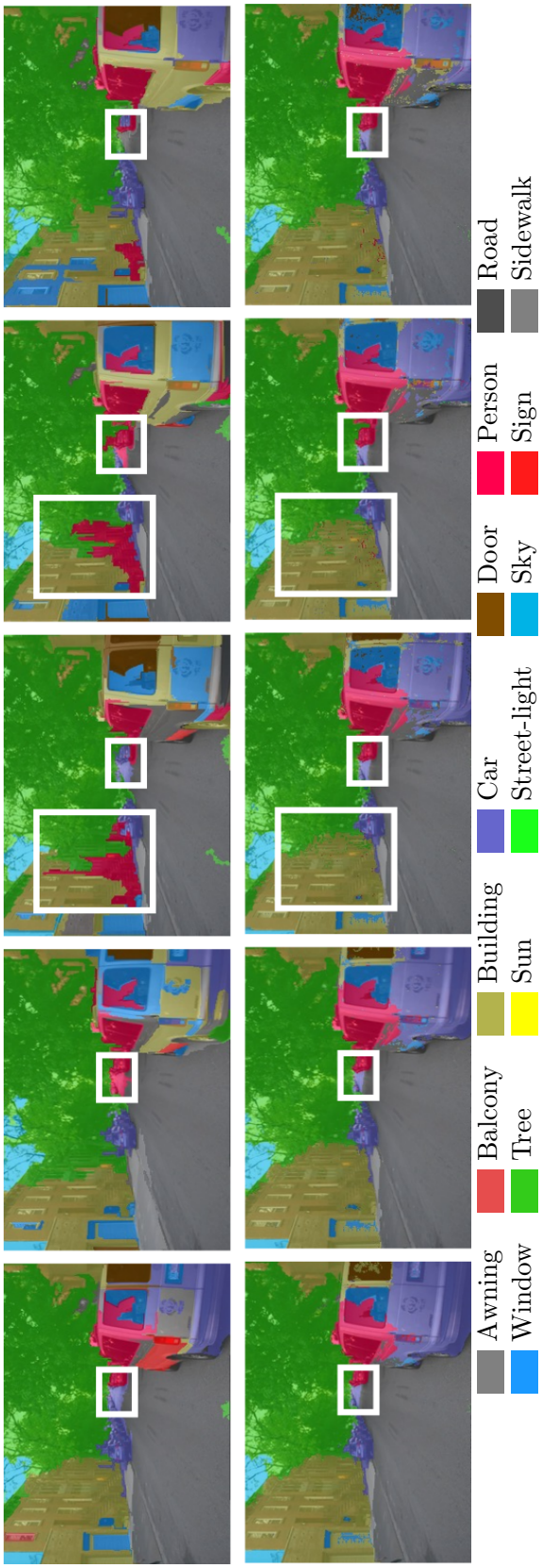


Figure 8.6: Temporal classifications on NYU Scenes frames 55-59. Top: per-frame. Bottom: temporally smoothed. Inconsistent predictions are highlighted.

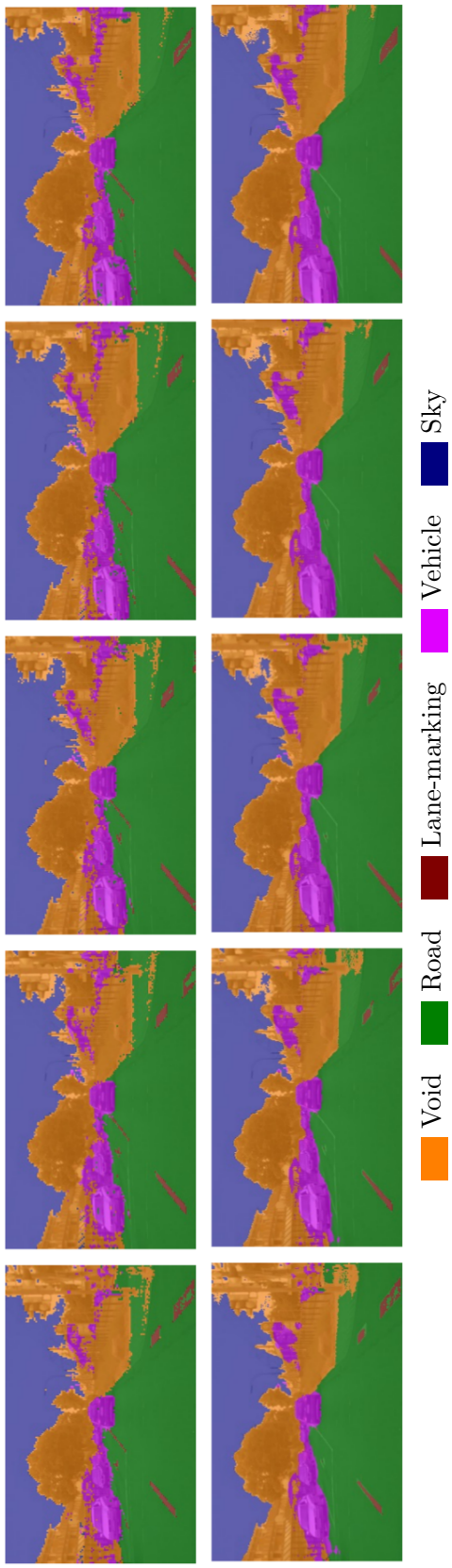


Figure 8.7: Temporal classifications on MPIVEHICLESCENES frames 265-269. Top: per-frame. Bottom: temporally smoothed. Inconsistent predictions are highlighted.

Chapter 9

Efficient 3-D Scene Parsing from Streaming Data

9.1 Introduction

In this chapter, we address the problem of scene parsing from 3-D data when the data is continuously streamed from a sensor on a moving vehicle, as shown in Figure 9.1. In order to obtain high performance predictions, it has been shown that is necessary to use models that encode the structure/relationships of the predictions (Anguelov et al., 2005, Munoz et al., 2009a, Koppula et al., 2011). However, in the streaming-data setting, the efficient use of these structured models is a challenging problem due to both theoretical and practical issues. As these algorithms rely on analyzing the entire scene, rather than individual points/voxels, it is unclear how to update the various components of the inference process when 3-D points are being continuously streamed from the sensor. For example, many approaches (Xiong et al., 2011, Douillard et al., 2011b, Lai and Fox, 2010, Golovinskiy et al., 2009) rely on representing the scene with a segmentation and analyzing the resulting groups/regions/segments instead of points. When data is streaming from the sensor, it is unclear how to efficiently insert new data into an existing segmentation without having to recompute the solution from scratch. Furthermore, structured prediction techniques rely on analyzing the entire scene at once, and it is difficult to efficiently update, rather than recompute, the joint solution with the newly streamed data (Kohli and Torr, 2007).

In practice, we are often forced to make a compromise in the inference process for the sake of efficient predictions. For example, instead of using a segmentation that obeys object boundaries, we might choose a technique that is less precise but more efficient. Additionally, instead of using expressive contextual models, we might limit ourselves to less expressive models with efficient approximate inference algorithms, or

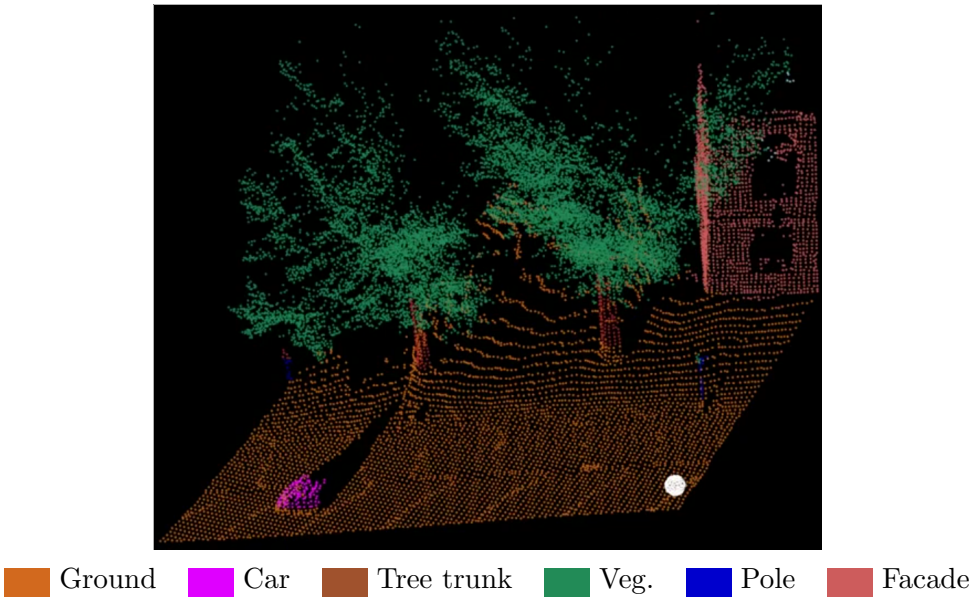


Figure 9.1: A screenshot of classifying streaming 3-D data, where the white ball is the sensor location. Many videos are available at http://www.cs.cmu.edu/~dmunoz/

even use a simple classifier. In this work, we demonstrate that we do not need to compromise efficiency for performance, or *vice versa*, and that we can generate state-of-the-art contextual classifications at a high enough rate to handle streamed sensor data on a mobile robot. Specifically, we demonstrate that a simple and efficient, yet imprecise, representation of the scene, when used in conjunction with the region-based scene parsing technique from Chapter 6 is able to efficiently predict state-of-the-art classifications.

The descriptions of our approach are broken down as follows. In Section 9.2, we describe a data structure that will enable us to efficiently extract regions, perform contextual neighborhood searches, and classify data over large regions of interest around the robot. In Section 9.3, we describe our representations of the scene and how they are used by the scene parsing algorithm. And in Sections 9.4, 9.5, and 9.6, we thoroughly analyze the different aspects of our approach and demonstrate its efficiency and efficacy on real-world datasets.

9.1.1 Related Work

In contrast to techniques that perform efficient object detection/classification from streaming 3-D data (Himmelsbach et al., 2009, Mertz et al., 2012, Teichman and Thrun, 2012), which often filter out a large portion of the data, we address the problem of efficiently understanding entire 3-D point cloud *scenes*. Related works (Lalonde et al., 2007, Bansal et al., 2011, Wellington and Stentz, 2003) have similarly focused on scene analysis for

robot mobility; however, we address longer range scene understanding, which is important for urban-scale semantic mapping. Additionally, recent works (Hadjiliadis and Stamos, 2010, Stamos et al., 2012) have investigated efficient techniques for classifying streaming point cloud data based on hand-designed filters. The key difference of our work from all the above is that we address the problem of efficient *structured prediction* and can use context in our predictions from a rich set of semantic object categories that would otherwise be difficult to encode using only point cloud descriptors. This work greatly improves upon our earlier work (Munoz et al., 2009b) on structured prediction from streaming data that uses a graphical model. We use a completely different data representation and adapt the hierarchical inference machine framework (Chapter 6) for improved efficiency.

The two key ingredients of this approach that affect its implementation as an online algorithm are 1) the set of 3-D operations that need to be performed on the point cloud, and 2) the representation of the scene that is fed to the scene parsing algorithm. We stress that the operations and representation are essential and universal to any 3-D scene parsing technique. First, in order to efficiently compute feature descriptors from the point cloud, it is necessary to have a data structure that can perform efficient range search operations over a subvolume in the space. Example standard descriptors, which we also use in our experiments, that require this operation are spin images (Johnson and Hebert, 1999) and local curvature statistics (Medioni et al., 2000). Second, many techniques use a segmentation algorithm to analyze over 3-D *regions*, instead of individual points (Golovinskiy et al., 2009, Lai and Fox, 2010, Xiong et al., 2011, Koppula et al., 2011). With our inference algorithm, we use multiple (four) segmentations of the point cloud to form a hierarchical segmentation as input. We address these two topics for the streaming data scenario in the following two sections.

9.2 Data Structures for Streaming Data

9.2.1 Scrolling Grids

One of the prevalent data structures for classifying streaming 3-D data is a scrolling grid representation (Wellington and Stentz, 2003, Lalonde et al., 2007, Bansal et al., 2011). Briefly, this data structure quantizes a pre-specified fixed volume of space into a grid composed of n^3 small voxels of prespecified resolution. When robot moves, the indices of the voxels are shifted/scrolled using a circular buffer. As the size of the grid and resolution of the voxels are known, it is straightforward to insert a point into a voxel inside the grid. Similarly, determining the voxels that constitute a queried subvolume of space can be computed by calculating the min and max extrema voxels and iterating

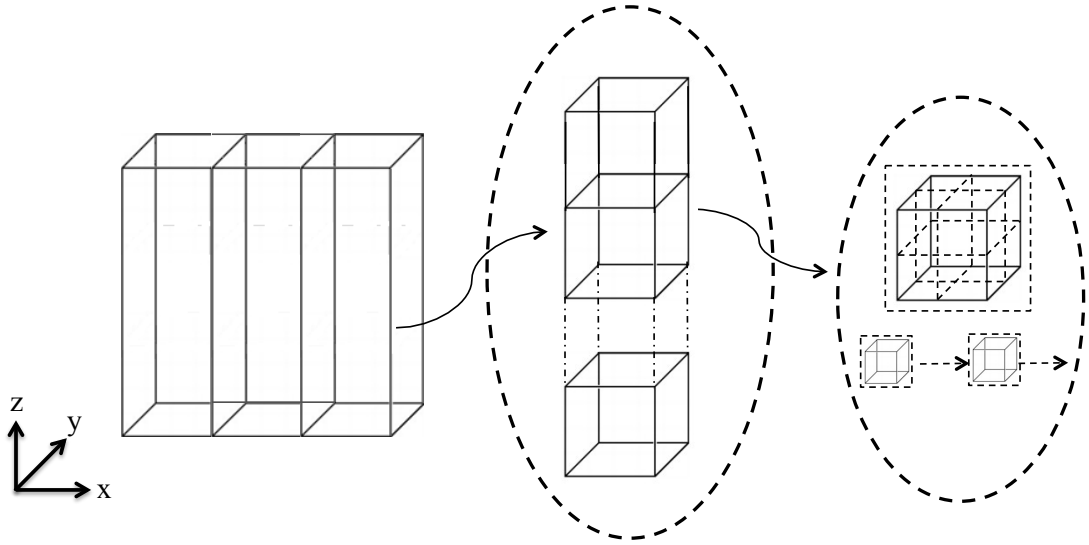


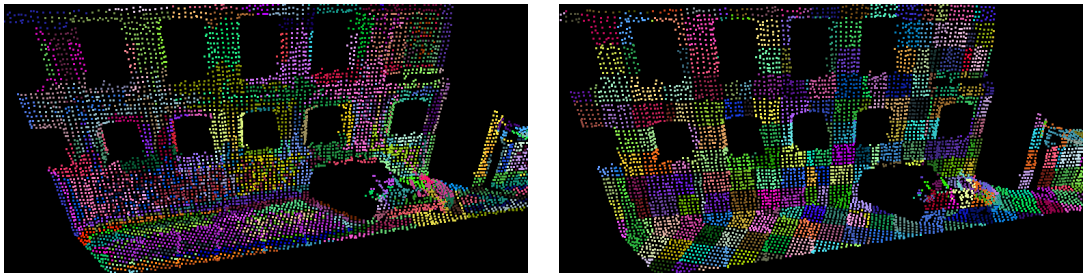
Figure 9.2: Visualizations of our data structures. Left: the world is sparsely quantized into infinitely tall *pillars*. Middle: each pillar is sparsely quantized into coarse *blocks*. Right: each block contains a linked list of its occupied *voxels*.

over the subvolume. This data structure is efficient for querying small subvolumes of space; however, when making long-range queries, which is necessary to model contextual interactions among physical objects in the scene, the queried, typically sparse, subvolume becomes computationally expensive to densely iterate over.

9.2.2 Sparse Global Grids

Because we need to randomly query very large subvolumes of space, most of which are sparse, we designed a voxel-based data structure to handle this sparsity and still enable efficient long-range query operations. Instead of maintaining a subset of streamed data within a fixed volume, we propose to store all streamed data in a sparse, voxel-based global map. To classify a local map of interest around the robot, we can efficiently query a large subvolume of the space to process with the scene parsing algorithm. Furthermore, as this local map is a subset of the data structure, it still maintains the efficient range search operations necessary for local neighborhood operations needed for feature computation.

In addition to large subvolume queries, we also require efficient insertion of streamed data over time. Similar to an octree, we consider multiple partitions of the 3-D space to efficiently ignore empty space. Figure 9.2 illustrates the following explanation. In this work, we use $(0.25\text{ m})^3$ *voxels* as the atomic unit which we assign object categories to. We coarsely partition the 3-D space of voxels into cube-shaped *blocks*, each of which



(a) (b)

Figure 9.3: Comparison of (a) F-H and (b) grid segmentations.

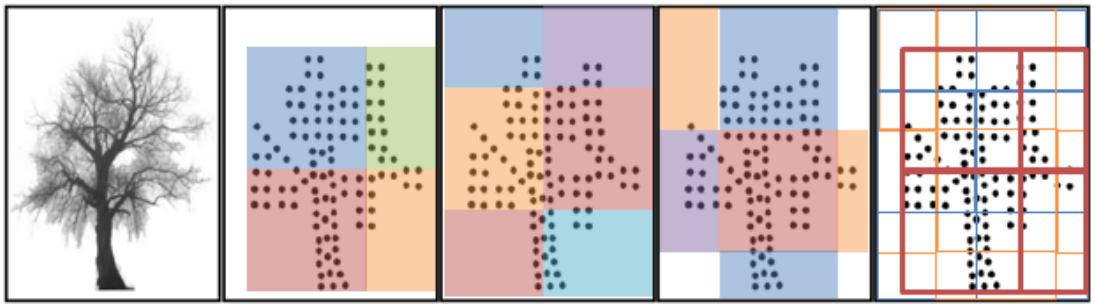
is uniquely indexed based on its global location and stored in a hash map. We can loop over these coarse blocks and retrieve the voxels within to perform queries over large subvolumes, skipping over empty blocks which can reduce search time when the space is sparse. As the block resolution will affect efficiency, we analyze this parameter in Section 9.4. We specify the block resolution as an integer multiple of the voxel resolution.

As we want to classify potentially very tall structures in the scene, we want to query local maps of infinite height (z-axis) around the robot. Again, because the space is sparse over large volumes of space, we further group the blocks into *pillars*, each of which is a linked list of non-empty blocks of the same x-y indices and is stored on a hash map. Thus we can efficiently form a local map around a robot by looping over pillars based on an x-y value to retrieve only the non-empty blocks and, thus, the voxels within.

To summarize: each pillar contains a linked list of blocks of the same x-y indices; each block contains a constant-sized 3-D array of voxels and a linked list of non-empty voxels within the block; each voxel contains accumulated statistics of the respective 3-D *points* that fell within the voxel. In our implementation a global map contains a list of non-empty voxels, a hash map of blocks, and a hash map of pillars. Voxels, blocks, and pillars are created and updated as new 3-D points are inserted into the global map.

9.3 Segmentation

The input to many scene analysis algorithms is a 3-D segmentation. The F-H (Felzenszwalb and Huttenlocher, 2004) segmentation algorithm (described in Section 3.2) is efficient and prevalent for segmenting 3-D point clouds in batch (Strom et al., 2010, Triebel et al., 2010); an example F-H segmentation is shown in Figure 9.3-a. While F-H is efficient in theory and practice, it does require non-negligible computational costs. First, the algorithm relies on a graph representation, and constructing edges among neighboring nodes in 3-D space relies on local range searches which require non-trivial amounts of time. Second, a notion of similarity between nodes is needed and requires



(a) (b) (c) (d) (e)

Figure 9.4: Visualization of a multi-grid segmentation. (a) An image of the scene. (b) A gridded segmentation of the scene, where each dot represents a voxel and regions are colored. (c,d) Two grids spatially offset from (b). (e) A multi-grid formed by the union of the (b,c,d), where the boundary colors (red, blue, orange) indicate the respective originating grid.

some form of feature computation. Third, in order to incorporate context in predictions, many algorithms (Koppula et al., 2011, Xiong et al., 2011) rely on using *contextual neighborhoods*, *i.e.*, adjacencies between regions within some fixed radius. As illustrated in Figure 9.3-a, regions resulting from F-H can be irregularly shaped, and accurately computing adjacent regions involves additional range searches. Note that an expanded bounding box approximation would be too crude as regions can be non-convex and/or span extremely large areas of space. Furthermore, filtering points that do lie within some radius of any point within the free-form region may also be costly. Finally, in the streaming data scenario, it is unclear how to efficiently update the previous segmentation with each newly inserted node without having to recompute the segmentation from scratch[1].

Instead of performing a precise segmentation that attempts to obtain object boundaries, we use regions extracted from fixed, gridded partitions of the 3-D space, as shown in Figure 9.3-b. We refer to this gridded segmentation as a *grid*. This simple approach addresses all of the previous computational concerns: there are no associated setup/construction/feature computations, contextual neighbors can be efficiently found due to all regions having bounded shape, and newly inserted points do not affect the existing segmentation.

When we arbitrarily partition the space, the resulting grid-regions may contain more than one object and/or cut through the boundary of another object. As the per-voxel classification is generated from the finest level segmentation, there will be unrecoverable

[1]Although there exists efficient data structures for modifying minimum spanning trees that have complexity sublinear in the number of edges for each online update (Frederickson, 1983), this would be impractical with streaming 3-D data.

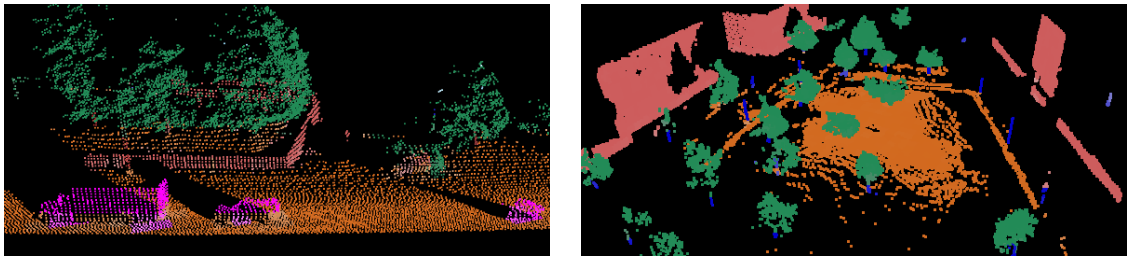


Figure 9.5: Example 3-D point cloud classifications from the VMR OAKLAND-V2 (Xiong et al., 2011) (left) and FREIBURG (Behley et al., 2012) (right) datasets.

errors if there exist multiple objects within one region. To address this quantization artifact, similar to how we use a hierarchical segmentation that consists of multiple segmentations at different resolutions in scale, we also consider grids at multiple spatial displacements/offsets from each other. That is, for a grid of fixed resolution, we create multiple grids whose region boundaries are spatially offset from each other in the following way. Suppose an initial grid is constructed with $l \times l \times l$ m^3 resolution regions and that we are considering γ different displacements of the initial grid. In the i^{th} grid, for $i \in \{0, \ldots, \gamma\}$, each of the 3 world coordinates for any region corner has the value $o + \frac{il}{\gamma} + kl$, where $o \in \mathbb{R}$ is the choice of origin and $k \in \mathbb{Z}$ specifies the corner. We refer to the union of the regions from each grid as a *multi-grid*. Although a multi-grid is not a proper "segmentation" due to elements (voxels) being contained within multiple regions, we refer to one multi-grid as a single segmentation in that it is a set of (overlapping) regions. Finally, since a voxel may be contained within multiple regions across displaced grids, the voxel's final label distribution is the unweighted average over all the label distributions of the respective regions it falls into. Figure 9.4. illustrates a multi-grid with $\gamma = 2$.

9.4 Efficiency Analysis

In the remaining sections, we compare various performance metrics with using F-H segmentation vs. simple (multi-)grids. Our analysis is performed on the 3-D point cloud datasets VMR OAKLAND-V2 (Xiong et al., 2011) and FREIBURG (Behley et al., 2012). Examples of classified scenes from each dataset are shown in Figure 9.5. For the computation analysis in this section, we use the training and validation folds from the VMR OAKLAND-V2 dataset. For classification analysis, we evaluate *voxel* classification error and assign the ground truth label to each voxel as the mode ground truth label from its respective points. All timing results were obtained on an Intel i7-2960XM processor.

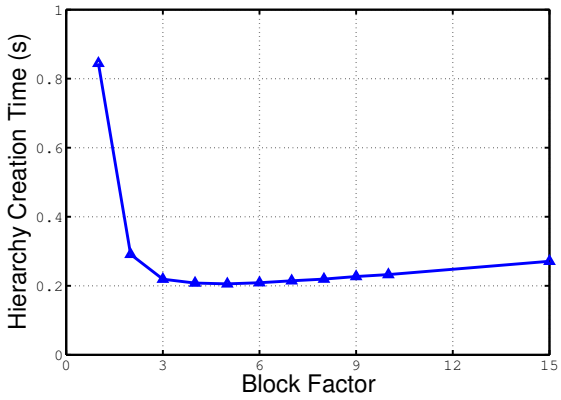


Figure 9.6: Average region hierarchy construction time, on validation data, with respect to block factor ($\frac{\text{block-resolution}}{\text{voxel-resolution}}$).

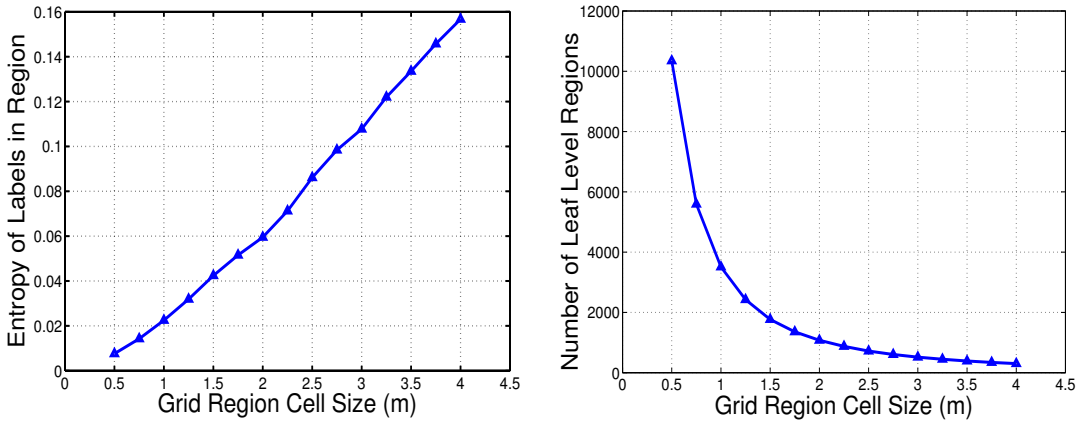
9.4.1 Setup

For each segmentation algorithm, F-H and (multi-)grids, we construct a 4-level hierarchy, from fine to coarse, by varying parameters that affect scale. When constructing the graph for F-H we use a spatial neighborhood of 0.5 m radius to create edges between two voxels, and we use the Euclidean distance between two feature vectors that encode local geometry and orientation (Medioni et al., 2000). The specifics of the grid partitions are discussed in the following subsections.

We compute the same four types of region features over the two different hierarchical segmentations. 1) A bag-of-words representation through feature quantization using soft k-means assignment (Coates et al., 2011) over two per-voxel descriptors: a) 5×5 spin images of $(0.2 \text{ m})^2$ cell resolution, b) local geometry and orientation features computed over three local neighborhoods of radii 0.5 m, 0.8 m, and 1.1 m, respectively. 2) Relative elevations using a 2.5-D elevation map. 3) The shape of the region through spectral analysis of the voxel coordinates that constitute the region, weighted by the number of points that fell into the voxel (Xiong et al., 2011). 4) The region's bounding box statistics (Xiong et al., 2011).

9.4.2 Block Resolution

Our global grid uses $(0.25 \text{ m})^3$ voxels as the atomic element for classification. As previously mentioned, we perform efficient range searches using coarse neighboring blocks to skip over empty volumes of space. We select the resolution of the blocks by analyzing the construction time of our region grid hierarchy, which is a function of the range searches needed to compute the feature descriptors and contextual neighborhoods. In Figure 9.6 we plot the average computation time with respect to coarsening block resolution, which



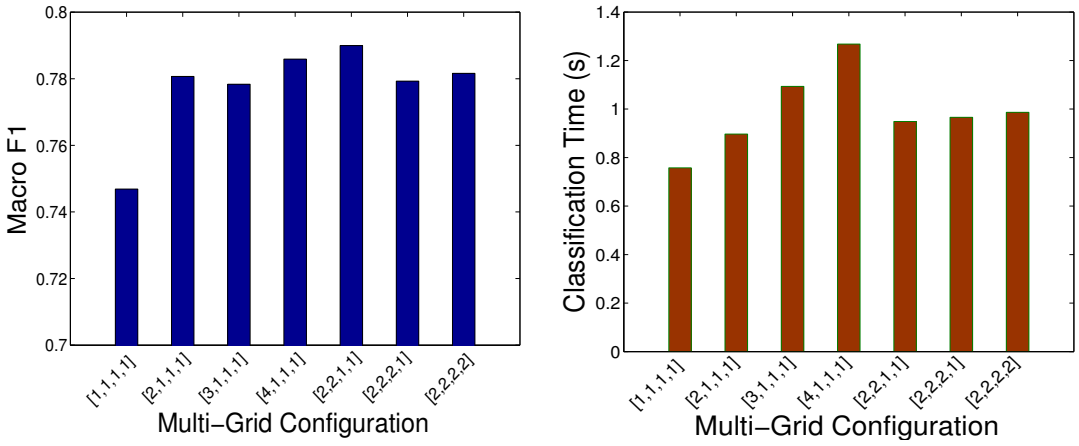
(a) (b)

Figure 9.7: Analysis, on validation data, of region grid resolution at the finest level. (a) Ground truth label entropy vs. region size. (b) Number of regions vs. region size.

is quantified by the ratio $\frac{\text{block-resolution}}{\text{voxel-resolution}}$ and is referred to as a "block factor". As expected, we observe a block factor of 1, meaning iterating over every neighboring voxel, is the slowest. In contrast, we see computation time start to increase when the block resolution coarsens to a factor more than 5. Hence, we use a block factor of 5 in the remaining experiments.

9.4.3 Finest Segmentation Resolution

The final voxel classifications are generated from the finest level in the region hierarchy. Ideally, we would choose the finest segmentation so that each voxel is a unique region in order to avoid any quantization artifacts; however, this precision comes at the cost of more samples to classify and increases inference time. On the other hand, using larger regions runs the risk of grouping together different labels within one region. To quantify this mixture of labels within a region, we can compute the average ground truth label entropies for regions with different sizes (from the training set). This value directly relates to the error rate for assigning a single label to a region containing a mixture of labels. In Figure 9.7 we plot the trade-offs of entropies (a) and number of generated regions (b) for different region grid sizes. We observe an exponential drop in the number of regions as the cell size increases, which is good for efficiency, and an increasing entropy, which hurts performance. As a compromise to balance efficiency and accuracy, we choose $(1.5\text{ m})^3$ as the grid region resolution in our finest level segmentation. The resolutions of the three coarser segmentations in the hierarchy are less sensitive as it is only the finest level segmentation that assigns the per-voxel labels. We use increasingly coarse regions of $(3.5\text{ m})^3$, $(7.5\text{ m})^3$, $(10\text{ m})^3$ resolution, respectively, for the remaining levels.



(a) (b)

Figure 9.8: Analysis, on validation data, of (a) classification performance and (b) computation time with respect to different multi-grid configurations.

9.4.4 Multi-grid Configuration

A multi-grid is the union of multiple grids of the same resolution with spatial displacement from each other. The more grids we have, the more robust we are to arbitrary quantization artifacts. However, this improvement in precision comes at the cost of having more regions in the scene to analyze. In Figure 9.8, we analyze, on the validation fold, the behavior of using multi-grids (with various sizes) at different levels in the region hierarchy. As our hierarchy contains four levels, we specify the multi-grid configuration of each level with a 4-tuple $[\gamma_1, \gamma_2, \gamma_3, \gamma_4]$, where γ_ℓ is the number of grids in the multi-grid of level ℓ in the hierarchy and $\ell = 1$ is the finest segmentation level. From Figure 9.8-a, we observe a large improvement in performance when simply using 2 grids in the leaf level ($[2, 1, 1, 1]$) vs. using only one ($[1, 1, 1, 1]$) multi-grid. Figure 9.8-b shows that this improvement in performance comes at an extra cost of 0.14 s when classifying a scene, on average. However, we also observe diminishing returns in average/macro per-class F_1 performance as we increase the number of grids with respect to computation time. Therefore, we use the $[2, 1, 1, 1]$ multi-grid configuration in the remaining experiments.

The most costly computation in constructing the region hierarchy is determining the contextual neighborhoods for each region. Typically, we perform very large range searches, up to 10 m, in order to model long-range interactions. If we consider the number of grids in a multi-grid, γ, the overall computation time for determining neighboring context over all regions is roughly raised by the number of additional offset grids. To avoid this extra cost, when computing contextual neighborhoods in multi-grids on the regions that are offset by a relatively small distance, we perform the following approxi-

Table 9.1: Breakdown of average computation times for constructing the hierarchical regions for a Grid (a) and a $[2, 1, 1, 1]$ Multi-grid (b).

Level	Number of Regions	Segmentation (ms)	Features (ms)	Context (ms)	Total (ms)	Context (%)
0	1862.6	10.0	13.8	54.1	77.9	69.45
1	426.7	10.0	10.0	26.1	46.1	56.62
2	107.3	8.2	8.8	12.3	29.3	41.98
3	63.6	6.4	8.4	12.6	27.4	45.99
Total	2460.2	34.6	41.0	105.1	180.7	58.16

(a) Grid

Level	Number of Regions	Segmentation (ms)	Features (ms)	Context (ms)	Total (ms)	Context (%)
0	3748.3	19.7	27.7	69.2	116.6	59.35
1	426.7	10.5	10.3	26.4	47.2	55.93
2	107.3	7.4	9.0	12.5	28.9	43.25
3	63.6	6.6	8.5	12.8	27.9	45.88
Total	4345.9	44.2	55.5	120.9	220.6	54.81

(b) Multi-grid

mation. We know that regions that overlap have some fixed, equally spaced offset from each other, and that the offsets are small with respect to the context range. Therefore, for regions that overlap each other, the contextual neighborhoods cover essentially the same 3-D space. Hence, instead of computing multiple neighborhoods for every offset region, we compute one contextual neighborhood and share it with its overlapping regions. In Table 9.1, we decompose the average timings for constructing a 4-level region hierarchy using a grid and a $[2, 1, 1, 1]$ multi-grid segmentation. We demonstrate that the use of the neighborhood approximation achieves comparable timing as with using a single grid.

9.5 Classification Analysis

9.5.1 Addressing Quantization Artifacts

Because our grid representation uses a fixed partitioning of the space, the segmentation is not invariant under rotations/translations of the scene. Note that although precise segmentation algorithms, such as F-H, are invariant to these transformations, the quantized voxels may not be if they are too coarse. We address this problem when training the models for both the F-H and (multi-)grid hierarchies. For each training point cloud, we generate additional training scenes by rotating the original scene around the z-axis

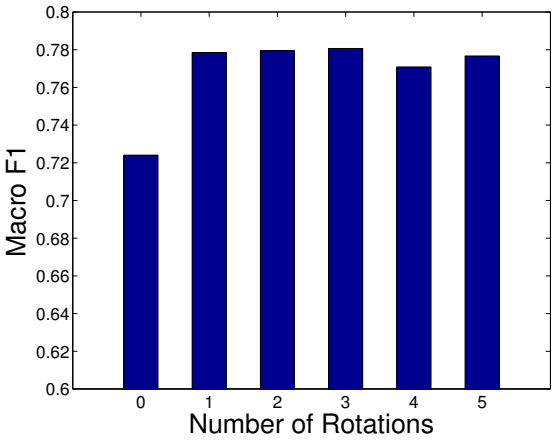


Figure 9.9: Average per-class F_1, on validation data, with respect to the number of times the training data is rotated.

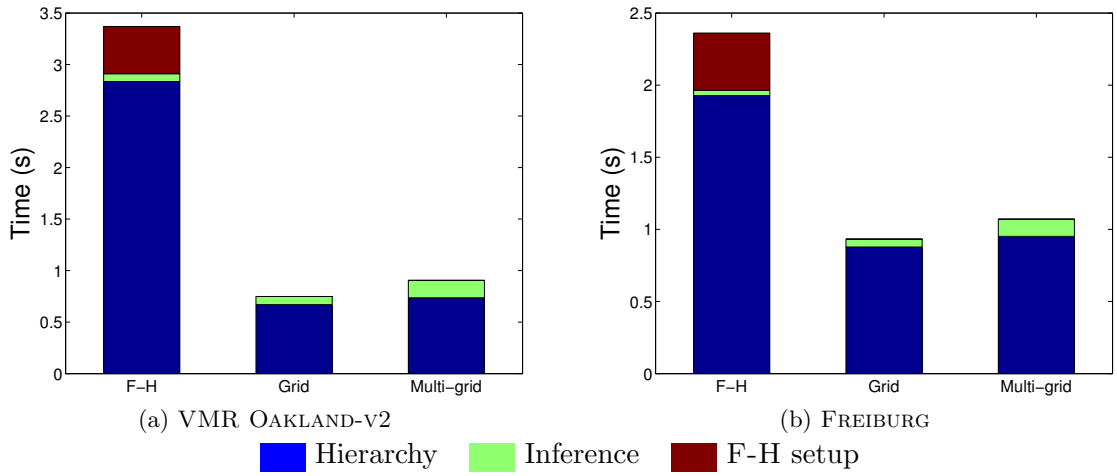
with equally spaced angles between $[0, \pi/2]$. In Figure 9.9, we quantify on the VMR OAKLAND-V2 validation set the performance with respect to the number of times we rotate each training scene; we use 3 rotations in the remaining experiments.

9.5.2 Classification Performance

We now evaluate, on the evaluation folds from each dataset, the efficiency and classification performances of the 3 models trained on different hierarchical region representations: 1) F-H, 2) Grid, and 3) $[2, 1, 1, 1]$ Multi-grid.

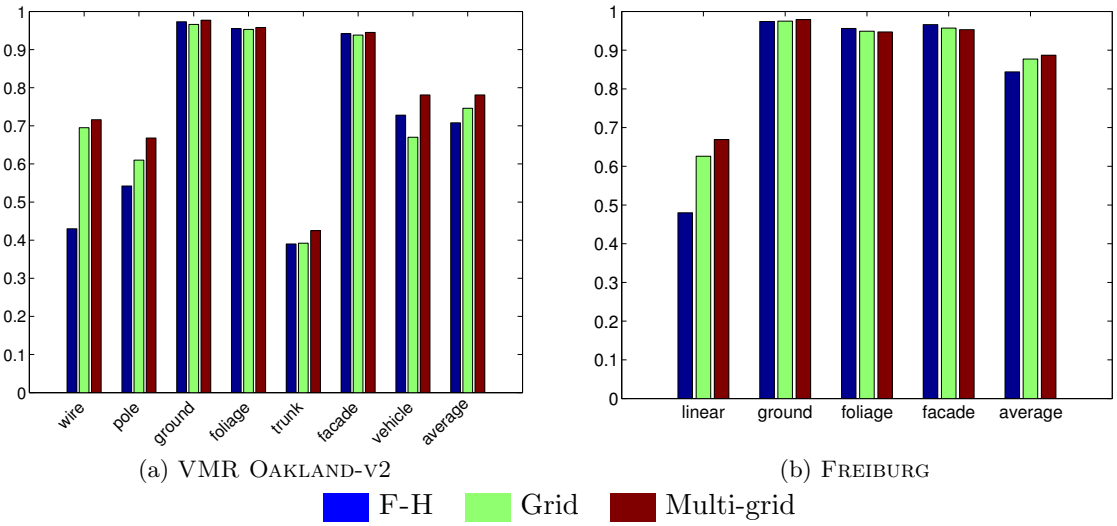
In Figure 9.10, we break down the computation for each model on the two different datasets. We observe that the grid-based model timings are inversely related with the F-H model. That is, the grid-based region constructions are much faster than F-H; however, F-H compresses the scene into a smaller number of regions which results in a faster inference time. Overall, we observe the average computation time with a multi-grid is 2.5-3x faster than using the more precise F-H segmentation, per static scene.

In Figure 9.11, we present the classification rates for the different models, in terms of F_1 scores for each class. We observe that using a grid-based segmentation can exceed the performance of using a precise F-H segmentation. This follows from the property that the scene analysis algorithm (Munoz et al., 2010b) is robust to imperfect segmentations due to explicitly modeling the distributions of labels within regions. Additionally, we can further improve performance by using a multi-grid to account for discretization artifacts from a single grid. In conjunction with the previous timing information, we conclude that this is an efficient and effective approach to perform full 3-D scene analysis.



(a) VMR Oakland-v2 (b) Freiburg

Figure 9.10: Average timings of each component during the entire inference procedure. Hierarchy construction includes feature computation time.



(a) VMR Oakland-v2 (b) Freiburg

Figure 9.11: Per-class F_1 for the datasets. "average" is the mean over the classes.

9.6 Streaming Classification

We demonstrate the practical benefits of using our efficient representation in the streaming data scenario. Both datasets contain sensor logs collected while the robot was moving. The VMR Oakland-v2 log was collected from a push-broom laser scanner mounted on the side of a moving vehicle, and the sequence is broken down into three smaller logs. The Freiburg log was collected on a wheeled robot moving in a campus environment. The sensor was a pan-tilt laser that scans a 360° field of view, and data was collected in

Table 9.2: Video sequence statistics

	VMR Oakland-v2	Freiburg
Avg. Number of 3-D Points / Scene	44,198	452,330
Avg. Number of Voxels / Scene	10,904	34,031
Total Number of Classified Scenes	398	1,059
Total Distance Traveled (m)	2,950	723

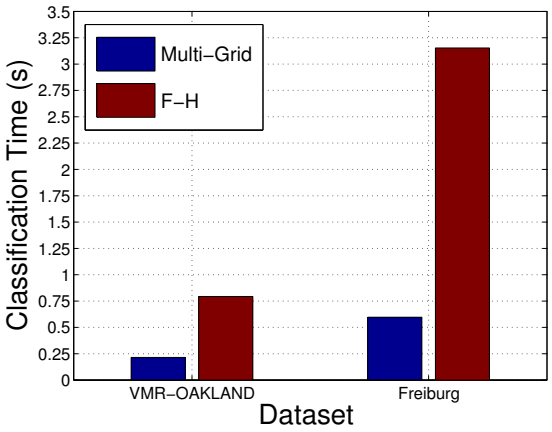


Figure 9.12: Average classification time per scene using multi-grid and F-H segmentation on streams of VMR Oakland-v2 and Freiburg datasets.

a stop-and-go manner from 77 different scanning locations. Real-time classifications of each dataset are available at http://www.cs.cmu.edu/~dmunoz/.

We process each log in the same manner: after inserting the last $10,000$ streamed 3-D *points* into the global voxel grid, we construct a local map of 20 m L_∞ radius, in the x-y plane, centered at the mean coordinate of the newly inserted $10,000$ points. Due to the profile scanning pattern in the VMR Oakland-v2 dataset, the resulting local map is approximately 20×20 m^2; however, it is a full 40×40 m^2 in the Freiburg dataset. We refer to this local map as a "scene" and then construct the region hierarchy and classify the scene with our scene analysis algorithm.

In Table 9.2, we break down the average number of 3-D points and voxels contained within each scene for each dataset, as well as the total number of classifications needed to process each sequence and how far the robot traveled. On average, the Freiburg scenes contains 3x the number of voxels to classify than in VMR Oakland-v2 and each scene covers a much larger area of space. The supplementary video shows the processing of each log using the multi-grid model. The video is screen captured in real-time and demonstrates the ability of our approach to efficiently process data for use on board mobile robots. In Figure 9.12, we compare the average classification time, per scene,

when using a simple multi-grid representation vs. F-H segmentation. Using the F-H representation is 3-5x more expensive than using the efficient multi-grid representation and would greatly limit a robot's speed in practice.

9.7 Summary

This chapter describes a simple modification of the segmentation algorithm used in Chapter 6 for 3-D scene parsing from streaming data. At its heart, we demonstrated that we do not use sophisticated segmentations that attempt to model the object discontinuities. Instead, we can simply use multiple, overlapping, fixed-sized partitionings of the 3-D space while modeling the distribution of object categories inside each resulting region/block. The main benefit of this segmentation is that it is agnostic to whether the data processed in a batch or online setting. Hence, the resulting segmentation can easily facilitate data that is streaming in from the sensor. We demonstrated that we can use this simple gridded segmentation approach to achieve the best of both worlds: state-art-of-the-art classifications with extremely efficient computations.

Chapter 10

Future Directions

This thesis proposes a structured prediction framework for accurate and efficient scene parsing from images and/or 3-D point clouds. The proposed sequential decoding approach enables the computation of rich intermediate representations of the scene and breaks down the complex inference procedure to a series of simpler subproblems. How to improve upon this framework raises some challenging problems.

10.1 Learning Structure

The hierarchical structure and representation of the scene is inspired by many works in vision for scene parsing (Ohta et al., 1978, Bouman and Shapiro, 1994, Feng et al., 2002, He et al., 2004, Kumar and Hebert, 2005). However, the hierarchical regions we use are often redundant due to multiple, inexact segmentations. Anecdotally, we have investigated how to improve the segmentation over the sequence of predictions to limit this redundancy. Unfortunately, despite a quantitative improvement in the *segmentation* performance (with respect to object boundaries), this do not result an improvement in *classification* performance. Alternatively, instead of always using *all* of these regions, it might be more efficient and effective to have the model automatically identify the portions of the scene that require additional processing. That is, the model incorporates its current beliefs of the scene to guide its future processing.

10.2 Learning Context

The way we encoded neighboring predictions was designed with *a priori* knowledge of the world, *e.g.*, in 3-D point clouds we discretized the predictions by elevation. When applying this framework to other domains, it may be unclear the best way to encode the context information. Alternatively, it would be more powerful to remove this hand-

Figure 10.1: Example ground truth annotation of an urban scene using the LabelMe tool (Russell et al., 2007).

designed representation as a prerequisite, and instead have the model learn how to compress the neighboring context/predictions into a more discriminative representation.

10.3 Learning Features

As we saw in Chapter 6, feature descriptors can have a major impact in the overall classification performance. Indeed, if we were able to compute local features that linearly separated each class, we could use simple predictors such as linear regression or SVMs. This general problem of feature learning for scene parsing has shown to produce promising results (Socher et al., 2011, Ren et al., 2012, Farabet et al., 2013). However, many techniques decouple the procedure of learning an intermediate representation and then defining a pooled/meta representation on top, *e.g.*, pooling the activation codes to train another classifier. Coupling the predictions with the representation may net more accurate predictions.

10.4 Semi-supervised Structured Prediction

The stacking trick (Wolpert, 1992) can be thought of as an agnostic technique to regularize any model in a way that might not be possible through canonical regularization of its parameters. However, in the presence of increasing amounts of data, the benefit of regularizing the model diminishes and there may be no need to use stacking. Unfortunately, obtaining large amounts of annotated *scenes* can be difficult in practice, especially in cases of complex scenes as illustrated in Figure 10.1. Semi-supervised learning for structured prediction (Altun et al., 2005, Brefeld and Scheffer, 2006, Lee et al., 2006) is an attractive class of techniques to use. Unfortunately, our experience with these techniques applied to scene parsing led to negative results (Munoz et al., 2010a), which also follows others' observations in both theory (Lafferty and Wasserman, 2007) and practice (Li and Zhou, 2011). Personally, I believe how to effectively leverage large amounts of unlabeled data for scene parsing is the most important problem to address for actual usage of these techniques in the field. In practice, how to effectively use abundant amounts of data to regularize the model or lower the sample complexity (Kakade and Foster, 2007) remains difficult.

10.5 Task-based Scene Parsing

The vast majority of current evaluation methods for scene parsing examine the per-pixel/point classification performance. This is not too useful in the real-world for a couple of reasons. First, this metric does not consider the spatial extent of specific object instances, but rather an agglomeration of pixels/points. Secondly, and more fundamentally, many current techniques only vary in a few percentage points in terms of classification accuracy on the pixels – do these extra bits really translate into a superior understanding of the scene? Alternatively, the evaluation can shift towards scoring whether the parsed scene sufficiently enables the completion of a task, *e.g.*, did the robot succeed to getting from point A to point B? This raises the question of how specific of a representation of the scene is necessary to complete a task? Does knowing that an "object" is specifically a "chair" make any difference? Furthermore, can we do away with *a priori* classes/categories (Malisiewicz, 2011) and using interaction with the world to automatically learn the sufficient representation needed to complete a task?

Bibliography

Michal Aharon, Michael Elad, , and Alfred Bruckstein. K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation. *IEEE Transactions on Signal Processing*, 54(11), 2006.

Yasemin Altun, David McAllester, and Mikhail Belkin. Maximum Margin Semi-Supervised Learning for Structured Variables. In *Advances in Neural Information Processing Systems*, 2005.

Dragomir Anguelov, Ben Taskar, Vassil Chatalbashev, Daphne Koller, Dinkar Gupta, Geremy Heitz, and Andrew Y. Ng. Discriminative Learning of Markov Random Fields for Segmentation of 3D Range Data. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. From Contours to Regions: An Empirical Evaluation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

David Arthur and Sergei Vassilvitskii. k-means++: The Advantages of Careful Seeding. In *ACM-SIAM Symposium on Discrete Algorithms*, 2007.

Vijay Badrinarayanan, Fabio Galasso, and Roberto Cipolla. Label Propagation in Video Sequences. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

Mayank Bansal, Bogdan Matei, Ben Southall, Jayan Eledath, and Harpreet Sawhney. A LIDAR Streaming Architecture for Mobile Robotics with Application to 3D Structure Characterization. In *IEEE International Conference on Robotics and Automation*, 2011.

Adrian Barbu. Training an Active Random Field for Real-Time Image Denoising. *IEEE Transactions on Image Processing*, 18(11), 2009.

Emre Baseski, Nicolas Pugeault, Sinan Kalkan, Dirk Kraft, Florentin Worgotter, and Norbert Kruge. Indoor Scene Segmentation using a Structured Light Sensor. In

International Conference on Computer Vision Workshops (3D Representation and Recognition), 2007.

Jens Behley, Volker Steinhage, and Armin B. Cremers. Performance of Histogram Descriptors for the Classification of 3D Laser Range Data in Urban Environments. In *IEEE International Conference on Robotics and Automation*, 2012.

Yoshua Bengio. Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning*, 2(1), 2009.

Paul J. Besl and Ramesh C. Jain. Invariant Surface Characteristics for 3D Object Recognition in Range Images. *Computer Vision, Graphics, and Image Processing*, 33 (1), 1986.

Leon Bottou and Yann LeCun. Large Scale Online Learning. In *Advances in Neural Information Processing Systems*, 2004.

Charles A. Bouman and Michael Shapiro. A Multiscale Random Field Model for Bayesian Image Segmentation. *IEEE Transactions on Image Processing*, 3(2), 1994.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast Approximate Energy Minimization via Graph Cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(1), 2001.

Ulf Brefeld and Tobias Scheffer. Semi-Supervised Learning for Structured Output Variables. In *International Conference on Machine Learning*, 2006.

Leo Breiman. Random Forests. *Machine Learning*, 45(1), 2001.

William Brendel and Sinisa Todorovic. Learning Spatiotemporal Graphs of Human Activities. In *IEEE International Conference on Computer Vision*, 2011.

Gabriel J. Brostow, Jamie Shotton, Julien Fauqueur, and Roberto Cipolla. Segmentation and Recognition Using Structure from Motion Point Clouds. In *European Conference on Computer Vision*, 2008.

Gabriel J. Brostow, Julien Fauqueur, and Roberto Cipolla. Semantic Object Classes in Video: A High-Definition Ground Truth Database. *Pattern Recognition Letters*, 30 (2), 2009.

Gal Chechik, Varun Sharma, Uri Shalit, and Samy Bengio. Large Scale Online Learning of Image Similarity Through Ranking. *Journal of Machine Learning Research*, 11, 2010.

Albert Y. C. Chen and Jason J. Corso. Temporally Consistent Multi-class Video-object Segmentation with the Video Graph-Shifts Algorithm. In *IEEE Workshop on Applications of Computer Vision*, 2011.

Stanley F. Chen and Ronald Rosenfeld. A Survey of Smoothing Techniques for ME Models. *IEEE Transactions on Speech and Audio Processing*, 8(1), 2000.

Adam Coates, Honglak Lee, and Andrew Y. Ng. An Analysis of Single-Layer Networks in Unsupervised Feature Learning. In *International Conference on Artificial Intelligence and Statistics*, 2011.

William W. Cohen and Vitor R. Carvalho. Stacked Sequential Learning. In *International Joint Conference on Artificial Intelligence*, 2005.

Alvaro Collet, Manuel Martinez, and Siddhartha S. Srinivasa. The MOPED Framework: Object Recognition and Pose Estimation for Manipulation. *The International Journal of Robotics Research*, 30(10), 2011a.

Alvaro Collet, Siddhartha S. Srinivasa, and Martial Hebert. Structure Discovery in Multi-modal Data: a Region-based Approach. In *IEEE International Conference on Robotics and Automation*, 2011b.

Corinna Cortes and Vladimir Vapnik. Support-vector Networks. *Machine Learning*, 20 (3), 1995.

Antonio Criminisi and Jamie Shotton. *Decision Forests for Computer Vision and Medical Image Analysis*. Springer, 2013.

Mark Cummins and Paul Newman. Appearance-only SLAM at Large Scale with FAB-MAP 2.0. *The International Journal of Robotics Research*, 30(9), 2011.

Hal Daume III, John Langford, and Daniel Marcu. Search-based Structured Prediction. *Machine Learning*, 75(3), 2009.

Jason Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit Dhillon. Information-theoretic Metric Learning. In *International Conference on Machine Learning*, 2007.

Roderick de Nijs, Sebastian Ramos, Gemma Roig, Xavier Boix, Luc van Gool, and Kolja Kuhnlenz. On-line Semantic Perception Using Uncertainty. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

Chaitanya Desai, Deva Ramanan, and Charless C. Fowlkes. Discriminative Models for Multi-Class Object Layout. *International Journal of Computer Vision*, 95(1), 2011.

Thomas G. Dietterich, Adam Ashenfelter, and Yaroslav Bulatov. Training Conditional Random Fields via Gradient Tree Boosting. In *International Conference on Machine Learning*, 2004.

Cristian Dima, Nicolas Vandapel, and Martial Hebert. Classifier Fusion for Outdoor Obstacle Detection. In *IEEE International Conference on Robotics and Automation*, 2004.

Bertrand Douillard, Dieter Fox, Fabio T. Ramos, and Hugh F. Durrant-Whyte. Classification and Semantic Mapping of Urban Environments. *International Journal on Robotics Research*, 30(1), 2011a.

Bertrand Douillard, James Patrick Underwood, Noah Kuntz, Vsevolod Vlaskine, Alastair James Quadros, Peter Morton, and Alon Frenkel. On the segmentation of 3D LIDAR Point Clouds. In *IEEE International Conference on Robotics and Automation*, 2011b.

Miroslav Dudik, Steven J. Phillips, and Robert E. Schapire. Maximum Entropy Density Estimation with Generalized Regularization and an Application to Species Distribution Modeling. *Journal of Machine Learning Research*, 8, 2007.

Andreas Ess, Tobias Muller, Helmut Grabner, and Luc van Gool. Segmentation-based Urban Traffic Scene Understanding. In *British Machine Vision Conference*, 2009.

Scott E. Fahlman and Christian Lebiere. The Cascade-Correlation Learning Architecture. In *Advances in Neural Information Processing Systems*, 1990.

Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning Hierarchical Features for Scene Labeling. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013.

Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient Graph-Based Image Segmentation. *International Journal of Computer Vision*, 59(2), 2004.

Xiaojuan Feng, Christopher K. I. Williams, and Stephen N. Felderhof. Combining Belief Networks and Neural Networks for Scene Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4), 2002.

Thomas Finley and Thorsten Joachims. Training Structural SVMs when Exact Inference is Intractable. In *International Conference on Machine Learning*, 2008.

Greg N. Frederickson. Data Structures for On-line Updating of Minimum Spanning Trees. In *ACM Symposium on Theory of Computing*, 1983.

Yoav Freund and Robert E Schapire. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, 55(1), 1997.

Jerome H. Friedman. Greedy Function Approximation: A Gradient Boosting Machine. *Annals of Statistics*, 29(5), 2001.

Andrea Giachetti, Marco Campani, and Vincent Torre. The Use of Optical Flow for Road Navigation. In *IEEE International Conference on Robotics and Automation*, 1998.

Jacob Goldberger, Sam Roweis, Geoff Hinton, and Ruslan Salakhutdinov. Neighbourhood Components Analysis. In *Advances in Neural Information Processing Systems*, 2004.

Aleksey Golovinskiy, Vladimir G. Kim, and Thomas Funkhouser. Shape-based Recognition of 3D Point Clouds in Urban Environments. In *IEEE International Conference on Computer Vision*, 2009.

Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. John Hopkins University Press, 1996.

Stephen Gould, Paul Baumstarck, Morgan Quigley, Andrew Y. Ng, and Daphne Koller. Integrating Visual and Range Data for Robotic Object Detection. In *European Conference on Computer Vision Workshops (Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications)*, 2008a.

Stephen Gould, Jim Rodgers, David Cohen, Gal Elidan, and Daphne Koller. Multi-Class Segmentation with Relative Location Prior. *International Journal of Computer Vision*, 80(3), 2008b.

Stephen Gould, Richard Fulton, and Daphne Koller. Decomposing a Scene into Geometric and Semantically Consistent Regions. In *IEEE International Conference on Computer Vision*, 2009.

Alexander Grubb and J. Andrew Bagnell. Generalized Boosting Algorithms for Convex Optimization. In *International Conference on Machine Learning*, 2011.

Matthias Grundmann, Vivek Kwatra, Mei Han, and Irfan Essa. Efficient Hierarchical Graph-based Video Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

Abhinav Gupta, Alexei A. Efros, and Martial Hebert. Blocks World Revisited: Image Understanding Using Qualitative Geometry and Mechanics. In *European Conference on Computer Vision*, 2010.

Olympia Hadjiliadis and Ioannis Stamos. Sequential Classification in Point Clouds of Urban Scenes. In *International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*, 2010.

Raia Hadsell, Sumit Chopra, and Yann Lecun. Dimensionality Reduction by Learning an Invariant Mapping. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

Peter L. Hammer. Some Network Flow Problems Solved With Pseudo-Boolean Programming. *Operations Research*, 13, 1965.

Xuming He, Richard S. Zemel, and Miguel A. Carreira-Perpinan. Multiscale Conditional Random Fields for Image Labeling. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

Geremy Heitz, Stephen Gould, Ashutosh Saxena, and Daphne Koller. Cascaded Classification Models: Combining Models for Holistic Scene Understanding. In *Advances in Neural Information Processing Systems*, 2008.

Ralf Herbrich and Thore Graepel. A PAC-Bayesian Margin Bound for Linear Classifiers: Why SVMs Work. In *Advances in Neural Information Processing Systems*, 2000.

Michael Himmelsbach, Thorsten Luettel, and Hans-Joachim Wuensche. Real-time Object Classification in 3D Point Clouds Using Point Feature Histograms. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009.

Derek Hoiem, Alexei A. Efros, and Martial Hebert. Recovering Surface Layout from an Image. *International Journal of Computer Vision*, 75(1), 2007.

Hanzhang Hu, Daniel Munoz, J. Andrew Bagnell, and Martial Hebert. Efficient 3-D Scene Analysis from Streaming Data. In *IEEE International Conference on Robotics and Automation*, 2013.

Allison Janoch, Sergey Karayev, Yangqing Jia, Jonathan T. Barron, Mario Fritz, Kate Saenko, and Trevor Darrell. A Category-Level 3-D Object Dataset Putting the Kinect to Work. In *International Conference on Computer Vision Workshops (Consumer Depth Cameras in Computer Vision)*, 2011.

Edwin T. Jaynes. Information Theory and Statistical Mechanics. *Physical Review*, 106 (4), 1957.

Andrew E. Johnson and Martial Hebert. Using Spin-Images for Efficient Object Recognition in Cluttered 3D Scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5), 1999.

Vladimir Jojic, Stephen Gould, and Daphne Koller. Accelerated Dual Decomposition for MAP Inference. In *International Conference on Machine Learning*, 2010.

Sham Kakade, Yee Whye Teh, and Sam Roweis. An Alternate Objective Function for Markovian Fields. In *International Conference on Machine Learning*, 2002.

Sham M. Kakade and Dean P. Foster. Multi-View Regression via Canonical Correlation Analysis. In *Conference on Learning Theory*, 2007.

Jyrki Kivinen and Manfred K. Warmuth. Exponentiated Gradient versus Gradient Descent for Linear Predictors. *Information and Computation*, 132(1), 1997.

Pushmeet Kohli and M. Pawan Kumar. Energy Minimization for Linear Envelope MRFs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

Pushmeet Kohli and Philip H. S. Torr. Dynamic Graph Cuts for Efficient Inference in Markov Random Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(12), 2007.

Pushmeet Kohli, Lubor Ladicky, and Philip H. S. Torr. Robust Higher Order Potentials for Enforcing Label Consistency. *International Journal of Computer Vision*, 82(3), 2009.

Vladimir Kolmogorov. Convergent Tree-reweighted Message Passing for Energy Minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10), 2006.

Vladimir Kolmogorov and Ramin Zabih. What Energy Functions Can Be Minimized via Graph Cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26 (2), 2004.

Hema Swetha Koppula, Abhishek Anand, Thorsten Joachims, and Ashutosh Saxena. Semantic Labeling of 3D Point Clouds for Indoor Scenes. In *Advances in Neural Information Processing Systems*, 2011.

Zhenzhen Kou and William W. Cohen. Stacked Graphical Models for Efficient Inference in Markov Random Fields. In *SIAM International Conference on Data Mining*, 2007.

Daniel Kuettel, Matthieu Guillaumin, and Vittorio Ferrari. Segmentation Propagation in ImageNet. In *European Conference on Computer Vision*, 2012.

Alex Kulesza and Fernando Pereira. Structured Learning with Approximate Inference. In *Advances in Neural Information Processing Systems*, 2007.

Sanijv Kumar and Martial Hebert. Discriminative Random Fields. *International Journal of Computer Vision*, 68(2), 2006.

Sanjiv Kumar and Martial Hebert. A Hierarchical Field Framework for Unified Context-Based Classification. In *IEEE International Conference on Computer Vision*, 2005.

Sanjiv Kumar, Jonas August, and Martial Hebert. Exploiting Inference for Approximate Parameter Learning in Discriminative Fields: An Empirical Study. In *International Conference on Energy Minimization Methods in Computer Vision and Pattern Recognition*, 2005.

In So Kweon, Martial Hebert, and Takeo Kanade. Sensor Fusion of Range and Reflectance Data for Outdoor Scene Analysis. In *NASA Workshop on Space Operations, Automation, and Robotics*, 1988.

Lubor Ladicky. *Global Structured Models Towards Scene Understanding.* PhD Thesis, Oxford Brookes University, 2011.

Lubor Ladicky, Chris Russell, Pushmeet Kohli, and Philip H. S. Torr. Associative Hierarchical CRFs for Object Class Image Segmentation. In *IEEE International Conference on Computer Vision*, 2009.

Lubor Ladicky, Paul Sturgess, Karteek Alahari, Chris Russell, and Philip H. S. Torr. What, Where & How Many? Combining Object Detectors and CRFs. In *European Conference on Computer Vision*, 2010.

John Lafferty and Larry Wasserman. Statistical Analysis of Semi-Supervised Regression. In *Advances in Neural Information Processing Systems*, 2007.

John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *International Conference on Machine Learning*, 2001.

Kevin Lai and Dieter Fox. Object Recognition in 3D Point Clouds Using Web Data and Domain Adaptation. *International Journal on Robotics Research*, 29(8), 2010.

Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Detection-based Object Labeling in 3D Scenes. In *IEEE International Conference on Robotics and Automation*, 2012.

Jean-Francois Lalonde, Nicolas Vandapel, and Martial Hebert. Data Structures for Efficient Dynamic Processing in 3-D. *International Journal on Robotics Research*, 26(8), 2007.

Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11), 1998.

Yann LeCun, Sumit Chopra, Raia Hadsell, Marc'Aurelio Ranzato, and Fu-Jie Huang. A Tutorial on Energy-Based Learning. In *Predicting Structured Data*. MIT Press, 2006.

Chi-Hoon Lee, Shaojun Wang, Feng Jiao, Dale Schuurmans, and Russell Greiner. Learning to Model Spatial Dependency: Semi-Supervised Discriminative Random Fields. In *Advances in Neural Information Processing Systems*, 2006.

Victor Lempitsky, Andrea Vedaldi, and Andrew Zisserman. A Pylon Model for Semantic Segmentation. In *Advances in Neural Information Processing Systems*, 2011.

Thomas Leung and Jitendra Malik. Representing and Recognizing the Visual Appearance of Materials using Three-dimensional Textons. *International Journal of Computer Vision*, 43(1), 2001.

Yu-Feng Li and Zhi-Hua Zhou. Towards Making Unlabeled Data Never Hurt. In *International Conference on Machine Learning*, 2011.

Lin Liao, Tanzeem Choudhury, Dieter Fox, and Henry Kautz. Training Conditional Random Fields using Virtual Evidence Boosting. In *International Joint Conference on Artificial Intelligence*, 2007.

Joseph J. Lim, Pablo Arbelaez, Chunhui Gu, and Jitendra Malik. Context by Region Ancestry. In *IEEE International Conference on Computer Vision*, 2009.

Beyang Liu, Stephen Gould, and Daphne Koller. Single Image Depth Estimation from Predicted Semantic Labels. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

Ce Liu, Jenny Yuen, and Antonio Torralba. SIFT Flow: Dense Correspondence Across Scenes and Its Applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5), 2011.

Stuart P. LLoyd. Least Squares Quantization in PCM. *IEEE Transactions on Information Theory*, 28(2), 1982.

David G. Lowe. Distinctive Image Features from Scale-invariant Keypoints. *International Journal of Computer Vision*, 60(2), 2004.

Julien Mairal, Francis Bach, and Jean Ponce. Task-Driven Dictionary Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(4), 2012.

Tomasz Malisiewicz. *Exemplar-based Representations for Object Detection, Association and Beyond*. PhD Thesis, Carnegie Mellon University, 2011.

Llew Mason, Jonathan Baxter, Peter Bartlett, and Marcus Frean. Functional Gradient Techniques for Combining Hypotheses. In *Advances in Large Margin Classifiers*. MIT Press, 1999.

Larry Matthies, Chuck Bergh, Andres Castano, Jose Macedo, and Roberto Manduchi. Obstacle Detection in Foliage with Ladar and Radar. In *International Symposium on Robotics Research*, 2003.

Gerard Medioni, Mi-Suen Lee, and Chi-Keung Tang. *A Computational Framework for Segmentation and Grouping*. Elsevier, 2000.

Christoph Mertz, Luis Ernesto Navarro-Serment, David Duggins, Jay Gowdy, Robert MacLachlan, Paul Rybski, Aaron Steinfeld, Arne Suppe, Christopher Urmson, Nicolas Vandapel, Martial Hebert, and Chuck Thorpe. Moving Object Detection with Laser Scanners. *Journal of Field Robotics*, 30(1), 2012.

Branislav Micusik, Jana Kosecka, and Gautam Singh. Semantic Parsing of Street Scenes from Video. *International Journal on Robotics Research*, 31(4), 2012.

Ondrej Miksik, Daniel Munoz, J. Andrew Bagnell, and Martial Hebert. Efficient Temporal Consistency for Streaming Video Scene Analysis. In *IEEE International Conference on Robotics and Automation*, 2013.

Daniel Munoz, J. Andrew Bagnell, Nicolas Vandapel, and Martial Hebert. Contextual Classification with Functional Max-Margin Markov Networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009a.

Daniel Munoz, Nicolas Vandapel, and Martial Hebert. Onboard Contextual Classification of 3-D Point Clouds with Learned High-order Markov Random Fields. In *IEEE International Conference on Robotics and Automation*, 2009b.

Daniel Munoz, J. Andrew Bagnell, and Martial Hebert. On Two Methods for Semi-Supervised Structured Prediction. Technical Report CMU-RI-TR-10-02, Robotics Institute, Carnegie Mellon University, 2010a.

Daniel Munoz, J. Andrew Bagnell, and Martial Hebert. Stacked Hierarchical Labeling. In *European Conference on Computer Vision*, 2010b.

Daniel Munoz, J. Andrew Bagnell, and Martial Hebert. Co-inference for Multi-modal Scene Analysis. In *European Conference on Computer Vision*, 2012.

Elizbar A. Nadaraya. On Estimating Regression. *Theory of Probability and Its Applications*, 9(1), 1964.

David Nitzan, Alfred E. Brain, and Richard O. Duda. The Measurement and Use of Registered Reflectance and Range Data in Scene Analysis. *Proceedings of the IEEE*, 65(2), 1977.

Yu-ichi Ohta, Takeo Kanade, and Toshiyuki Sakai. An Analysis System for Scenes Containing Objects with Substructures. In *International Joint Conference on Pattern Recognitions*, 1978.

Nathan Ratliff. *Learning to Search: Structured Prediction Techniques for Imitation Learning*. PhD Thesis, Carnegie Mellon University, 2009.

Nathan Ratliff, J. Andrew Bagnell, and Martin Zinkevich. Online Subgradient Methods for Structured Prediction. In *International Conference on Artificial Intelligence and Statistics*, 2007.

Xiaofeng Ren, Liefeng Bo, and Dieter Fox. RGB-(D) Scene Labeling: Features and Algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

Stephane Ross and J. Andrew Bagnell. Efficient Reductions for Imitation Learning. In *International Conference on Artificial Intelligence and Statistics*, 2010.

Walter Rudin. *Functional Analysis*. McGraw-Hill, 1991.

Bryan Russell, Antonio Torralba, Kevin Murphy, and William T. Freeman. LabelMe: A Database and Web-based Tool for Image Annotation. *International Journal of Computer Vision*, 77(1-3), 2007.

Sunita Sarawagi and Rahul Gupta. Accurate Max-Margin Training for Structured Output Spaces. In *International Conference on Machine Learning*, 2008.

Ashutosh Saxena, Min Sun, and Andrew Y. Ng. Make3D: Learning 3-D Scene Structure from a Single Still Image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5), 2008.

Roman Shapovalov, Alexander Velizhev, and Olga Barinova. Non-associative Markov networks for 3D Point Cloud Classification. In *Photogrammetric Computer Vision and Image Analysis*, 2010.

Naum Zuselevich Shor. *Minimization Methods for Non-Differentiable Functions and Applications*. Springer-Verlag, 1985.

Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. TextonBoost for Image Understanding: Multi-Class Object Recognition and Segmentation by Jointly Modeling Texture, Layout, and Context. *International Journal of Computer Vision*, 81(1), 2009.

Nathan Silberman and Rob Fergus. Indoor Scene Segmentation using a Structured Light Sensor. In *International Conference on Computer Vision Workshops (3D Representation and Recognition)*, 2011.

Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo Tourism: Exploring Photo Collections in 3D. *ACM Transactions on Graphics*, 25(3), 2006.

Richard Socher, Cliff Lin, Andrew Y. Ng, and Christopher D. Manning. Parsing Natural Scenes and Natural Language with Recursive Neural Networks. In *International Conference on Machine Learning*, 2011.

Boris Sofman, Ellie Lin, J. Andrew Bagnell, John Cole, Nicolas Vandapel, and Anthony Stentz. Improving Robot Navigation Through Self-Supervised Online Learning. *Journal of Field Robotics*, 23(12), 2006.

Nathan Srebro and Adi Shraibman. Rank, Trace-Norm and Max-Norm. In *Conference on Learning Theory*, 2005.

Ioannis Stamos, Olympia Hadjiliadis, Hongzhong Zhang, and Thomas Flynn. Online Algorithms for Classification of Urban Objects in 3D Point Clouds. In *International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*, 2012.

Bastian Steder, Giorgio Grisetti, and Wolfram Burgard. Robust Place Recognition for 3D Range Data Based on Point Features. In *IEEE International Conference on Robotics and Automation*, 2010.

Johannes Strom, Andrew Richardson, and Edwin Olson. Graph-based Segmentation for Colored 3D Laser Point Clouds. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010.

Paul Sturgess, Karteek Alahari, Lubor Ladicky, and Philip H. S. Torr. Combining Appearance and Structure from Motion Features for Road Scene Understanding. In *British Machine Vision Conference*, 2009.

Rick Szeliski, Ramin Zabih, Daniel Scharstein, Olga Veksler, Vladimir Kolmogorov, Aseem Agarwala, Mashall Tappen, and Carsten Rother. A Comparative Study of Energy Minimization Methods for Markov Random Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(6), 2007.

Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-Margin Markov Networks. In *Advances in Neural Information Processing Systems*, 2003.

Ben Taskar, Vassil Chatalbashev, and Daphne Koller. Learning Associative Markov Networks. In *International Conference on Machine Learning*, 2004.

Alex Teichman and Sebastian Thrun. Tracking-Based Semi-Supervised Learning. *International Journal on Robotics Research*, 31(7), 2012.

Federico Tombari and Luigi Di Stefano. 3D Data Segmentation by Local Classification and Markov Random Fields. In *International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*, 2011.

Antonio Torralba, Kevin P. Murphy, and William T. Freeman. Contextual Models for Object Detection using Boosted Random Fields. In *Advances in Neural Information Processing Systems*, 2004.

Antonio Torralba, Kevin P Murphy., and William T. Freeman. Sharing Visual Features for Multiclass and Multiview Object Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(5), 2007.

Rudolph Triebel, Jiwon Shin, and Roland Siegwart. Segmentation and Unsupervised Part-based Discovery of Repetitive Objects. In *Robotics: Science and Systems Conference*, 2010.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large Margin Methods for Structured and Interdependent Output Variables. *Journal of Machine Learning Research*, 6, 2005.

Zhuowen Tu and Xiang Bai. Auto-context and Its Application to High-level Vision Tasks and 3D Brain Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(10), 2010.

Olga Veksler. *Efficient Graph-based Energy Minimization Methods in Computer Vision.* PhD Thesis, Cornell University, 1999.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked Denoising Autoencoders: Learning Useful Representations in a

Deep Network with a Local Denoising Criterion. *Journal of Machine Learning Research*, 11, 2010.

Paul A. Viola and Michael J. Jones. Robust Real-time Face Detection. *International Journal of Computer Vision*, 57(2), 2004.

Martin J. Wainwright. Estimating the "Wrong" Graphical Model: Benefits in the Computation-Limited Setting. *Journal of Machine Learning Research*, 7, 2006.

Martin J. Wainwright and Michael I. Jordan. Graphical Models, Exponential Families, and Variational Inference. *Foundations and Trends in Machine Learning*, 1(1-2), 2008.

Martin J. Wainwright, Tommi S. Jaakkola, and Alan S. Willsky. MAP Estimation via Agreement on Trees: Message-passing and Linear Programming. *IEEE Transactions on Information Theory*, 51(11), 2005.

Kilian Q. Weinberger and Lawrence K. Saul. Distance Metric Learning for Large Margin Nearest Neighbor Classification. *Journal of Machine Learning Research*, 10, 2009.

Carl Wellington and Anthony Stentz. Learning Predictions of the Load-Bearing Surface for Autonomous Rough-Terrain Navigation in Vegetation. In *International Conference on Field and Service Robotics*, 2003.

Paul J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD Thesis, Harvard University, 1974.

Manuel Werlberger, Werner Trobin, Thomas Pock, Andreas Wedel, Daniel Cremers, and Horst Bischof. Anisotropic Huber-L1 Optical Flow. In *British Machine Vision Conference*, 2009.

Christian Wojek and Bernt Schiele. A Dynamic Conditional Random Field Model for Joint Labeling of Object and Scene Classes. In *European Conference on Computer Vision*, 2008.

Christian Wojek, Stefan Roth, Konrad Schindler, and Bernt Schiele. Monocular 3D Scene Modeling and Inference: Understanding Multi-Object Traffic Scenes. In *European Conference on Computer Vision*, 2010.

David H. Wolpert. Stacked Generalization. *Neural Networks*, 5(2), 1992.

Jianxiong Xiao and Long Quan. Multiple View Semantic Segmentation for Street View Images. In *IEEE International Conference on Computer Vision*, 2009.

Eric Xing, Andrew Y. Ng, Michael Jordan, and Stuart Russell. Distance Metric Learning, with Application To Clustering with Side-information. In *Advances in Neural Information Processing Systems*, 2003.

Xuehan Xiong, Daniel Munoz, J. Andrew Bagnell, and Martial Hebert. 3-D Scene Analysis via Sequenced Predictions over Points and Regions. In *IEEE International Conference on Robotics and Automation*, 2011.

Chenliang Xu, Caiming Xiong, and Jason J. Corso. Streaming Hierarchical Video Segmentation. In *European Conference on Computer Vision*, 2012.

Chenxi Zhang, Liang Wang, and Ruigang Yang. Semantic Segmentation of Urban Scenes Using Dense Depth Maps. In *European Conference on Computer Vision*, 2010.

Lei Zhang and Qiang Ji. Image Segmentation with a Unified Graphical Model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8), 2010.