

Geometric Context From Single And Multiple Views

**Alexander John Flint
Exeter College**

Supervised by Prof Ian Reid and Prof David Murray
Active Vision Laboratory
Robotics Research Group
Department of Engineering Science
University of Oxford

Trinity Term 2012

This thesis is submitted to the Department of Engineering Science, University of Oxford, for the degree of Doctor of Philosophy. This thesis is entirely my own work, and, except where otherwise indicated, describes my own research.

Alexander John Flint
Exeter College

Doctor of Philosophy
Trinity Term 2012

Geometric Context From Single And Multiple Views

Abstract

In order for computers to interact with and understand the visual world, they must be equipped with reasoning systems that include high-level quantities such as objects, actions, and scenes. This thesis is concerned with extracting such representations of the world from visual input. The first part of this thesis describes an approach to scene understanding in which texture characteristics of the visual world are used to infer scene categories. We show that in the context of a moving camera, it is common to observe images containing very few individually-salient image regions, yet overall texture structure often allows our system to derive powerful contextual cues about the environment. Our approach builds on ideas from texture recognition, and we show that our algorithm out-performs the well-known Gist [61] descriptor on several classification tasks.

In the second part of this thesis we are interested in scene understanding in the context of multiple calibrated views of a scene, as might be obtained from a Structure-from-Motion or Simultaneous Localization and Mapping (SLAM) system. Though such systems are capable of localizing the camera robustly and efficiently, the maps produced are typically sparse point-clouds that are difficult to interpret and of little use for higher-level reasoning tasks such as scene understanding or human-machine interaction. In this thesis we begin to address this deficiency, presenting progress towards modeling scenes using semantically meaningful primitives such as floor, wall, and ceiling planes.

To this end we adopt the indoor Manhattan representation, which was recently proposed for single-view reconstruction [13]. This thesis presents the first in-depth description and analysis of this model in the literature. We describe a probabilistic model relating photometric features, stereo photo-consistencies, and 3D point clouds to Manhattan scene structure in a Bayesian framework. We then present a fast dynamic programming algorithm that solves exact MAP inference in this model in time linear in image size. We show detailed comparisons with the state-of-the art in both the single- and multiple-view contexts.

Finally, we present a framework for learning within the indoor Manhattan hypothesis class. Our system is capable of extrapolating from labelled training examples to predict scene structure for unseen images. We cast learning as a structured prediction problem and show how to optimize with respect to two realistic loss functions. We present experiments in which we learn to recover scene structure from both single and multiple views — from the perspective of our learning algorithm these problems differ only by a change of feature space. This work constitutes one of the most complicated output spaces (in terms of internal constraints) yet considered within a structure prediction framework.

Contents

1	Introduction	1
1.1	Motivation	2
1.2	Our Approach	4
1.2.1	Photometric Models	4
1.2.2	Coarse Geometric Models	5
1.3	Exegesis	6
2	Literature Review	8
2.1	Introduction	8
2.2	Context in Computer Vision	8
2.2.1	Holistic Approaches	9
2.2.2	Geometric Context	12
2.2.3	Manhattan–World Approaches	14
2.2.4	Texture Recognition	19
2.3	Context in Robotics	21
2.3.1	Ego–centric approaches	21
2.3.2	Map–centric approaches	23
2.4	Conclusions	25
3	Appearance–Only Context Models	27
3.1	Introduction	28
3.2	Background	29
3.3	Textons for Scene Understanding	30
3.3.1	Textons Select Salient Image Elements	31
3.3.2	Textons Focus Attention on Salient Image Regions	33
3.3.3	Textons Correlate With Scene Structure	33
3.4	Learning and Inference	35
3.5	Place Recognition	36
3.6	Camera Orientation Classification	37
3.7	Computational Concerns	40
3.7.1	Separable Kernels	42
3.7.2	Parallelization over filters	42

3.7.3	Parallelization over pixels	42
3.7.4	Timing results	43
3.8	Conclusions	43
4	The Geometry of Indoor Environments	45
4.1	Introduction	46
4.2	Indoor Manhattan Environments	49
4.3	Floorplans	50
4.3.1	Images of Floorplans	52
4.4	Parametrisation	53
4.4.1	Parametrising the floor and ceiling location	54
4.4.2	Parametrising walls	57
4.4.3	The Seam Representation	60
4.5	Deductions	61
4.5.1	Classification of Corners	61
4.5.2	Recovering Metric Scene Structure	63
4.5.3	Recovering Orientation and Depth	64
4.5.4	Column-wise Decomposability	65
4.6	Conclusion	66
5	Identifying Manhattan Orientations	67
5.1	Introduction	68
5.2	Background	69
5.2.1	Single View Approaches	69
5.2.2	Multiple View Approaches	70
5.3	Overview of Proposed Approach	71
5.4	Generative Model	71
5.5	Estimating the Manhattan Coordinate Frame	74
5.5.1	Complete Data Likelihood	75
5.5.2	The E-step	75
5.5.3	M-step	77
5.5.4	Summary	80
5.6	Results	80
5.7	Identifying The Vertical Direction	84
5.8	Identifying The Floor And Ceiling Planes	84
5.9	Extension: Relaxing The Manhattan World Assumption	85
5.10	Conclusion	86
6	Inference From Single and Multiple Views	91
6.1	Introduction	92
6.2	The Payoff Formulation	93

6.3	Probabilistic Model	95
6.3.1	Scene Prior	95
6.3.2	Photometric Sensor Model	96
6.3.3	Multiple–View Sensor Model	98
6.3.4	Point Cloud Sensor Model	103
6.3.5	Joint Model	105
6.4	MAP Inference Using Dynamic Programming	106
6.4.1	Preliminaries	108
6.4.2	Optimal Substructure	111
6.4.3	First Algorithm	114
6.4.4	First Refinement: Auxiliary Sub–problems	117
6.4.5	Second Refinement: From $O(L^3)$ to $O(L^2)$	119
6.5	Results	123
6.5.1	Payoff Matrices	130
6.6	Other Approaches	133
6.6.1	Branch and Bound	134
6.6.2	Graph Cuts	134
6.7	Discussion	135
6.7.1	Non–memoryless Scene Priors	135
6.7.2	Conclusion	137
7	Learning to Construct Indoor Manhattan Models	138
7.1	Introduction	138
7.2	Background	140
7.3	Model	142
7.3.1	Hypothesis Class	142
7.3.2	Feature Space	143
7.3.3	Loss Functions	147
7.4	Learning Algorithm	149
7.4.1	Inference (Prediction)	150
7.4.2	Loss–Augmented Inference (Separation)	151
7.5	Multiple View Results	151
7.6	Single View Results	154
7.7	Discussion	159
7.7.1	Condition in the joint feature space, not the input feature space.	159
7.7.2	Condition the loss terms.	160
7.7.3	Check that the hypothesis class contains the ground truth.	160
7.7.4	Equivalence of (7.34) and (7.42)	161
7.8	Conclusion	162

8 Conclusion	164
8.1 Key Results	164
8.2 Future Work	166
8.2.1 Applications of Indoor Manhattan Models	166
8.2.2 Relaxing Geometric Constraints	168
8.2.3 Extensions Of The Probabilistic Model	169
8.3 Final Remarks	170

Acknowledgements

This thesis is the culmination of many years of work and would not have been possible without the help and support of many people.

I would like to thank my supervisors, Professors Ian Reid and David Murray for their guidance and patience, and good spirit throughout the long English winters.

If there are any ideas of any value within this thesis, credit should undoubtedly go to the amazing people I had the privilege to work with at the Active Vision Lab in Oxford. In particular I would like to thank Christopher Mei and Gabe Sibley for inspiration, encouragement, and the most interesting coffee-time discussions I have had the privilege of participating in. In addition, I owe much to discussions and collaborations with Michael Osborne, Matthew Blaschko, Eric Sommerlade, and Victor Prisacariu.

My research was funded by a Clarendon Scholarship, for which I owe gratitude to the wonderful people at the Clarendon Foundation and the Oxford University Press.

Throughout this work I have enjoyed the unfailing support of my family. To my mother, Kate, my father, Roger, and my brother, Hugh: I owe you everything. Thank you for guiding me here, guiding me through, and always guiding me onwards.

I would also like to thank partners-in-crime Akshat Rathi, Priya and Preethi Vijayakumar, and Michelle Fernandez, for making my time at Oxford what it was.

Notation

Throughout this thesis, the following conventions will be used for typesetting mathematics unless otherwise indicated:

- **Sets** are written in calligraphic type: \mathcal{S} .
- **Fields** are written in upper case: \mathbb{F} . In particular, \mathbb{R} denotes the reals and \mathbb{Z} denotes the integers.
- **Vectors** are written in lower case bold: \mathbf{x} . Vectors are usually column vectors, with elements specified by subscript index (e.g. $\mathbf{x} = (x_1, x_2, x_3)$).
- **Matrices** are written in upper case: H . The entry in the i^{th} row and j^{th} column is $H_{i,j}$.
- **Conditional probabilities** are written $P(A \mid B)$. In most cases A and B will correspond to the event that certain variables take on certain values, for example: $P(x \mid y)$.
- **Expectations** of x with respect to a distribution p are written $\mathbb{E}_p[x]$.
- **Gradients** of y with respect to x are written $\frac{\partial y}{\partial x}$.
- **Inner products** for geometric quantities are written $\mathbf{x} \cdot \mathbf{y}$. In the more general context of Hilbert spaces we switch to the notation $\langle \mathbf{x}, \mathbf{y} \rangle$.
- Where there is a need to differentiate the estimated and true values of a quantity y , the former will be written \hat{y} and the latter y^* .

1

Introduction

Humans are visual in nature and as such we have constructed our world around visual cues. We use written street signs rather than, say, smells or sounds to navigate the road network because our biological vision system has both higher fidelity and wider bandwidth than other senses. In order for computers to understand and interact with our world we must either develop automated vision systems or re-signpost our world. Computer vision is the endeavour to take the former route.

This thesis addresses the problem of developing a computer vision system that understands the geometry of the environment at a semantically meaningful level. By “semantically meaningful” we mean at the level of objects and events relevant to humans and human interactions, as in the difference between an city roadmap (which we would say is semantically meaningful on account of its high-level representation of quantities such as roads and addresses) and a set of 3D points in space. Our goal is to recover semantically meaningful models of the world from visual input.

To this end we explore two types of models. The first relates low-level photometric information directly to high-level concepts such as the function of an environment and the location of objects within it. Building on recent progress in contextual computer vision, we show how to derive visual context directly from an environment’s texture structure. The second type of model we explore relates images to high-level concepts via an intermediate representation of

high-level scene geometry. Contrary to much previous work in geometric computer vision, we select a representation that leads naturally to semantic-level reasoning tasks such as place recognition and object detection, for which we argue that the indoor Manhattan model is an attractive choice. Much of this thesis concerns the estimation of indoor Manhattan models from single and multiple images.

1.1 Motivation

The research presented here has been motivated from a variety of angles over time. This thesis was originally motivated by the application area of augmented reality (AR). In the broadest sense, AR systems aim to overlay relevant information onto a human's visual field-of-view. One important challenge is to build a 3D model of the environment and track the sensor with respect to it in order to create the illusion of annotations that move with 3D objects in the environment. This problem is known as simultaneous localisation and mapping (SLAM), and has seen great progress over the past several decades. In particular, visual SLAM, in which the sensing apparatus is a camera, has taken great strides, moving from single rooms to entire cities, and from costly off-line processing to real-time systems. However, in order to select relevant information and have annotations interact seamlessly with other visual stimulus, the AR system must have some understanding of the visual world beyond the low-level point clouds generated by SLAM systems.

It is neither surprising nor objectionable that so much effort towards AR has focussed on visual SLAM, but now that high-performance SLAM systems *are* available, the question that naturally presents itself is: *are we done yet?* In this thesis we will argue that while SLAM is an important first step, many challenging and general computer vision problems remain to solve before AR systems can be fully realised. For example, the relevance of certain types of information to the wearer varies with the type of environment that the wearer is moving within. Directions to a restaurant may be relevant while outdoors but directions to the kitchen are less likely to be helpful within the wearer's own house. Within the home, the augmentations most useful in

the kitchen are likely to be different to those suitable to the bedroom. Furthermore, the likely locations of various objects is tightly coupled to the surfaces and boundaries within an environment. A human would not search for lost keys on the ceiling first, nor outside a window, but to make this inference an AR system must first identify key surfaces and their orientations. These problems are not immediately solved by the ability to locate oneself within an arbitrary 3D coordinate frame, though a major theme of this thesis is how to use such information together with other image evidence for high-level reasoning tasks.

Although this thesis was originally motivated by applications within AR, much inspiration was drawn from the single-view computer vision literature, principally from the literature dedicated to *scene understanding*, which is the problem of understanding images in terms of objects, actions, and semantically meaningful geometric primitives. Scene understanding has a long history within the single-view computer vision literature but a shorter history within the multiple-view literature. One motivation for this research is the extension of traditional scene understanding techniques to the context of multiple views. This is a challenging problem because single-view techniques are commonly cast in terms of rectangular arrays of pixels, which do not naturally incorporate the kind of geometric information available in a multiple view context.

There is a very large literature on single-view scene understanding, and a corresponding plethora of models to choose from. The focus of this thesis is on the extraction of high-level geometric information within indoor-type environments, with relevance to scene categorisation and object localisation. The hope is that this problem is large enough to be interesting, small enough to be tractable, and perhaps insightful enough to be instructive for future research in the relatively new domain of multiple-view scene understanding.

There are many reasons that utilising multiple views is an attractive direction for scene understanding research. Firstly, video on the internet has become increasingly important over the past several years, increasing the relevance of computer vision systems that process video streams rather than single images. Secondly, mobile phones have become more attractive devices for vision systems as the processors and cameras now ubiquitously attached to them con-



Figure 1.1: An example of an indoor Manhattan scene.

tinue to become more powerful. Finally, depth sensing cameras have entered the mainstream consumer hardware market and are likely to become more common over the coming years.

1.2 Our Approach

In this thesis we present two approaches to deriving semantic-level scene information from visual input.

1.2.1 Photometric Models

The first model we present is a purely photometric approach and builds on ideas from texture analysis, in particular the texton model. This model associates each pixel with a discrete texture element, which is then related to the semantic-level variable of interest. We show how textons can be leveraged toward the problems of scene recognition and contextual object search. Despite the simplicity of this approach, we show that it is effective for the proposed problems.

In contrast to systems based on edges, interest points, or local descriptors, utilising texture information allows us to leverage visual information in non-salient image regions. This is particularly important for environments in which there are few individually distinctive image patches, yet the overall scene appearance conveys significant information.

We present qualitative results motivating the use of texture structure for the stated task, followed by a formal probabilistic model and quantitative comparisons with the state-of-the-art.

1.2.2 Coarse Geometric Models

The second part of this thesis focusses on capturing geometric information in a form well-suited to semantic-level reasoning. This represents a departure from much of the scene understanding literature, in which geometric information is captured implicitly, rather than via explicit geometric assumptions. Our decision to work with explicit geometric models is motivated by the observation that much about scenes and objects is tied to the high-level shape of the environment, particularly the location of major surfaces and boundaries.

As a simple illustration of this, consider the image shown in Figure 1.1. Despite the low information content of the image – it is defined by just a handful of vertices – a human can make many high-level inferences about this environment, including answers to questions such as (in order of increasing difficulty):

1. What is the direction of gravity?
2. Where would doors most likely be found?
3. Where would a person be most likely to stand and at what scale?
4. Is this an office or a house?
5. What is the absolute scale of the environment?

While the image does not provide enough information for definitive answers to any of these questions, it does provide surprisingly strong evidence given that it consists of just a few hundred bits of information. Thus it seems that coarse geometric information is particularly valuable for high-level inference.

The image in Figure 1.1 shows one example of an indoor Manhattan environment [13], which is the hypothesis class we adopt for geometric reasoning in this thesis. Under this model the

world is composed of a floor plane, ceiling plane, and a set of vertical walls that meet at vertical edges. Indoor manhattan environments constitute a subset of the more general set of Manhattan environments, in which any arrangement of surfaces in three dominant directions is permitted. It is an attractive model for a number of reasons. Firstly, as argued above, it captures geometric information of high value for semantic-level reasoning. Secondly, despite the strong geometric assumptions, a surprisingly broad range of environments can be represented exactly or approximately as indoor Manhattan models. Thirdly, strong geometric assumptions assist in the interpretation of ambiguous image evidence, since salient information in one region can constrain the interpretation of other regions. Finally, an efficient inference algorithm exists for this hypothesis class, which is the subject of later chapters.

There is no hard distinction between a “geometry-less” and a “geometry-laden” approach to scene understanding. In either case one is drawing probabilistic inferences from images; the distinction is in how those inputs are combined, which conditional independences are assumed, and how the hypothesis class is formulated. Our choice to incorporate geometry corresponds to a particular choice of independence assumptions, which differ from those typically chosen in photometrically inspired models. This thesis is concerned with the recovery of geometry for the sake of the high-level inference that it enables, not for the sake of the geometry itself, and our choice of model reflects this.

1.3 Exegesis

The remainder of this thesis is organised as follows. Chapter 2 presents a review of literature relevant to this thesis. Due to the connection of our work to the traditionally disparate fields of SLAM and scene understanding, the literature review breaks down into scene understanding work that has touched on geometry, and SLAM work that has touched on scene understanding.

Chapter 3 presents an appearance-only approach to scene understanding in the context of a moving camera. We work with textons as the basic observation unit and present a probabilistic model connecting these to scene categories and object locations.

We then begin the presentation of our work concerning indoor Manhattan models. Chapter 4 presents the model formally, and covers various geometric observations on which later chapters rest. There is little prior work on the indoor Manhattan model, so we dedicate a full chapter to defining the model formally and describing several useful representations. We then turn in Chapter 5 to identifying the orientation of three cardinal Manhattan directions given multiple calibrated views. We show that estimating scene rotation jointly from all available views is a dramatic improvement over the standard approach of identifying vanishing points separately in each image. We also show an order of magnitude improvement over approaches that use surface normals estimated from a reconstructed point cloud.

Chapter 6 presents a probabilistic model relating sensor data to the geometric quantities we wish to extract. We give separate probabilistic models for photometric features, stereo data, and point clouds, then we show how to combine these into a fully Bayesian model, allowing our system to be easily tailored to a variety of contexts. We present a dynamic programming algorithm that solves both maximum–aposteriori and maximum–likelihood inference within our model. This inference procedure is both fast and exact, making a compelling alternative to previous approaches that explode combinatorially in model complexity.

Chapter 7 presents a learning routine in which we learn to reconstruct Manhattan environments based on training examples. We cast the learning problem in a discriminative framework and use state–of–the–art tools from the structured prediction literature. We show that learning with respect to single versus multiple views differs only by a change in feature space, and we present extensive results for both contexts.

Finally, chapter 8 summarises key results and suggests directions for future work.

2

Literature Review

2.1 Introduction

In this section we provide a review of the literature relevant to this thesis. We begin by examining the literature on context in single-view computer vision. This area is of great interest since many of the techniques may be transferable to our multiple-view setting. We then turn to a review of the use of context in robotics applications, a subject that has received somewhat sparse attention in the literature and has, as yet, no cohesive formulation or problem statement.

2.2 Context in Computer Vision

Over the past decade there has been renewed interest in the use of contextual information for computer vision tasks. There are many definitions of what “context” means and how to represent it. Here we review the dominant paradigms that have emerged around contextual reasoning in scene understanding. Many single-image contextual approaches have been targeted at a specific problem — namely that of object recognition — and while this is not aligned exactly with our own goal it is still instructive to review these contributions because the ideas they propose for inferring context are often separable from the specific task to which the authors choose to apply them.

2.2.1 Holistic Approaches

Gist Features

The work of Torralba *et al.* [61] has been very influential in expounding the value of contextual reasoning for vision. In their work, they compute a feature vector composed of statistics from the entire image and use this to reason about the contents of the image. This feature vector is termed the “gist” of the image and is computed as follows. First, an input image is passed through a bank of Gabor filters at n orientations and m scales, producing nm response images. Next, each response image is divided into a $k \times k$ grid. Finally the average over each grid cell is computed for each response image, and these values are concatenated to form the final feature vector of length nmk^2 .

In early work, Torralba *et al.* used the gist vector to learn about scene categories. They showed that images could be classified into categories such as “road”, “forest”, and “bedroom” by applying a support vector machine (explained below) directly to the gist vector. In later work [61] they show how the location and scale of objects can be predicted from the gist vector. The Expectation–Maximisation algorithm [1] is employed to learn a mixture of Gaussians that relates the gist features to the probability of an object being present at various locations and scales. An example prediction is shown in Figure 2.1.

Support vector machines (SVM) were proposed by Vapnik [63] just over a decade ago and have since gained widespread support both inside and outside the machine learning community. SVMs learn to discriminate two classes by finding the separating hyperplane in feature space for which the distance to training examples is maximised.

The Expectation–Maximisation algorithm is a widely used tool for parameter estimation in the presence of unobserved variables [1]. Direct inference in such cases would require marginalisation over all unobserved variables, but the integrals involved typically make this approach infeasible. The EM algorithm overcomes this by iterating between two stages. First, the expected likelihood is computed with respect to the unobserved variables (the E-step), and second, the

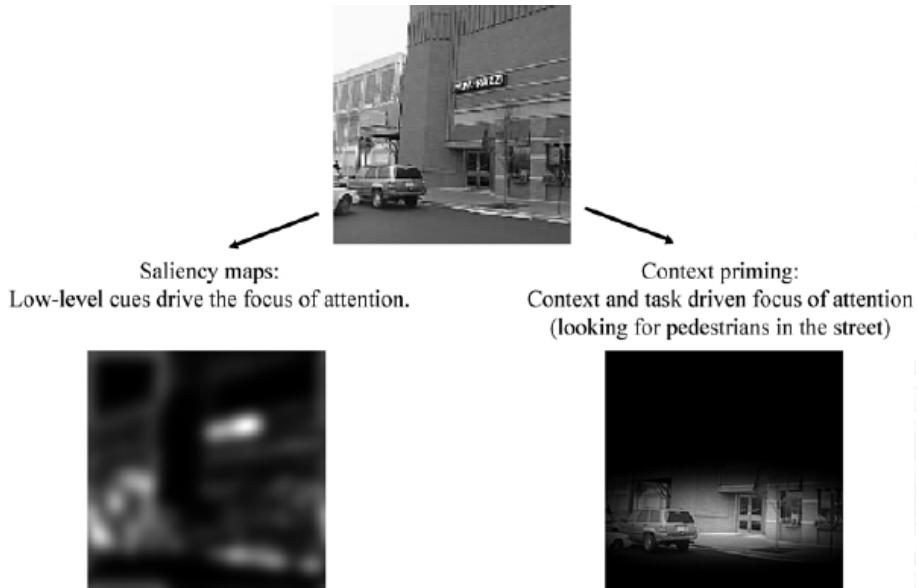


Figure 2.1: Torralba *et al.* [61] is able to improve upon traditional interest point detectors by learning the relationship between gist features and likely object locations. *Reprinted with express permission of Prof Antonio Torralba.*

model parameters are maximised with respect to the distribution computed in the first step (the M-step). These steps are repeated until convergence.

Scene Classification

Fei–Fei and Perona [17] classified images explicitly into semantic categories such as “coastal”, “inner city”, or “bedroom”. Although their work focuses on the classification task itself, the output scene label is clearly a useful piece of contextual information for integration into a broader scene understanding system.

Fei–Fei and Perona represent an image as a bag of words, where the “words” are obtained by clustering SIFT features (explained below) obtained at regularly sampled locations across all training images. Once the codebook of words has been learnt, each image is collapsed to a simple sequence of integers representing the index of the codebook entries identified within it.

Analogous to document understanding approaches in which each section is associated with an inferred topic, Fei–Fei and Perona model a hidden “topic” variable for each image feature. In their model they also include a theme variable, which induces a class–conditional multinomial

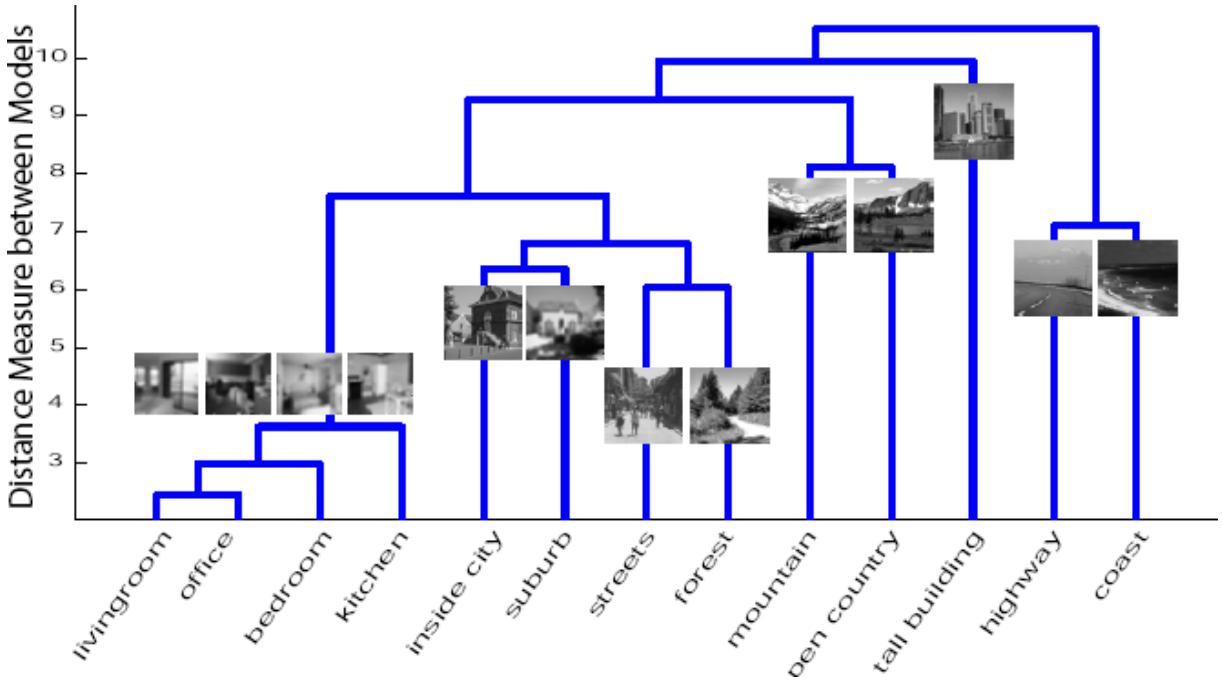


Figure 2.2: Dendrogram showing the relationships between scene categories learnt by the system of Fei-Fei and Perona [17]. The model matches human intuition well. *Reprinted with express permission of Prof Fei-Fei Li.*

distribution over topics.

On a dataset of 15 scene categories, each with several hundred images taken from the web as well as from datasets released by other researchers, the system obtains an average classification accuracy of 64.0%.

The scale-invariant feature transform (SIFT) was first proposed by Lowe [45] for use in object recognition. Given some input image, SIFT generates a set of interest points and corresponding feature vectors that are robust to changes in lighting, scale, and camera viewpoint. To select salient features at a range of scales, SIFT builds a scale-space representation of the image [44] and selects locations that are well-localised in both the spatial and scale dimensions. Features are generated from a histogram over gradient orientations in a patch around each selected interest point.

2.2.2 Geometric Context

An explicit understanding of scene geometry can assist image understanding in numerous ways. Several authors have shown how to obtain coarse 3D reconstructions from a single image, which allows rich geometric reasoning for inference about such things as the objects and actions likely to occur within the scene, and the scale and position at which these might be found.

Pixel-wise Geometry Estimates

Hoiem *et al.* [34] approach geometric context as a per-pixel labelling problem in which the labels identify geometric properties of the 3D surface from which each pixel was captured. Although there are many 3D scenes that could have generated any particular image, Hoiem *et al.* note that some scenes are more probable than others given our knowledge of the world. To keep this difficult inference task tractable, the authors limit the pixel labels to “sky”, “ground”, and “vertical”, the last of which is further sub-divided into “left-facing”, “right-facing”, and “front-facing”. While there are some real-world scenes that cannot be represented by these geometric primitives, the authors show that they are able to model many useful and interesting types of scenes.

The authors take a machine learning approach to the reconstruction problem. First, the image is segmented into superpixels using the algorithm of Felzenszwalb *et al.* [19]. Next, pairs of adjacent superpixels are merged in order to gradually grow the small superpixels into larger regions. To do this, an affinity metric between adjacent superpixels is learnt using a boosted decision tree classifier, the input to which is a feature vector containing cues such as colour, texture, location, shape, and vanishing points. At evaluation time many different segmentations are generated by considering the superpixels in different orders. For each ordering, the first k superpixels are assigned to unique segments, then the remaining superpixels are assigned to the segment with which they have the greatest affinity. Each segment is classified into one of the geometric classes listed above using another a boosted decision tree classifier with the same

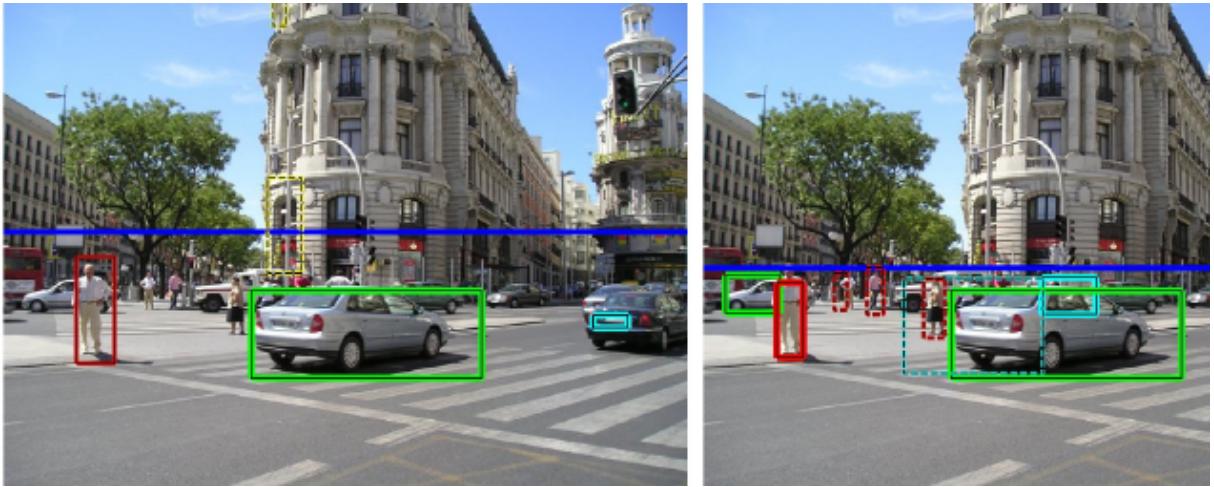


Figure 2.3: The left image shows car and pedestrian detections when using local features only. The right image shows detections when geometric context is leveraged: several false–positives are eliminated and several new correct detections are made. *Reprinted with express permission of Dr Derek Hoiem.*

input cues as for the affinity metric learning. Finally, superpixels are labelled according to the consensus vote amongst the segments to which they belong.

The authors further show that the geometric labels obtained in this way can be used as priors for object detection, since detections in unlikely places and scales can be suppressed, while those in geometrically consistent positions can be amplified. Figure 2.3 shows the improvement in detection performance resulting from the use of geometric context.

Make3D

Another approach to deriving geometric context has been proposed by Saxena *et al.* [55]. Rather than the fixed set of orientations employed by Hoiem *et al.*, Saxena *et al.* only assume that the scene is piece–wise planar. They allow these planar patchlets to take on arbitrary orientations, which they represent with a normal vector.

Saxena *et al.* apply a machine learning methodology to this problem. They begin by dividing the image into superpixels, each of which is associated with a feature vector containing the output of a set of filters, including colour, texture, and edges. Each superpixel also includes the features of neighbouring superpixels so that the inference process can reason in terms of

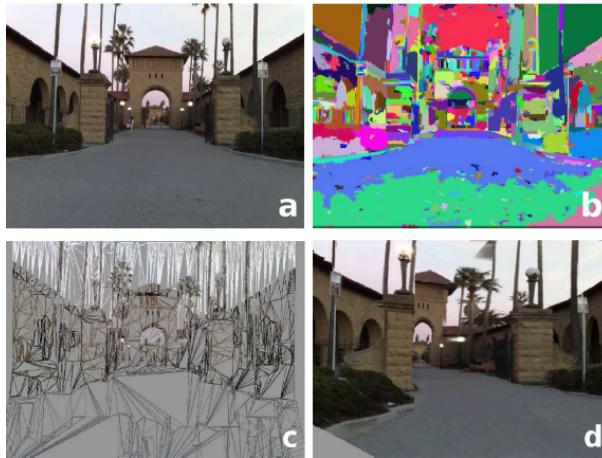


Figure 2.4: Results from the geometric context system of Saxena *et al.* [55]. From top left to bottom right the panes show (a) the original, (b) the superpixels, (c) the inferred 3D model, (d) a re-projection of the 3D scene. *Reprinted with express permission of Dr Ashutosh Saxena.*

broader image properties rather than local statistics alone. In addition, each superpixel boundary is associated with a separate set of features. These are generated by running segmentations based on different image properties and recording whether the boundary is present in each.

The superpixels are organised into a Markov Random Field (MRF), with edges between pairs that share a boundary in the image. The node potentials are learnt from the superpixel features and edge potentials are learnt from the boundary features. The authors allow for coplanar, connected, and disconnected relationships across boundaries, with the former preferred *a priori* over the latter. The MRF parameters are learnt through Multi-Conditional Learning and inference is performed by solving a linear program.

Within a dataset of 152 internet images the system was able to generate a qualitatively correct model (as judged by a human) 64.9% of the time. One such result is illustrated in Figure 2.4.

2.2.3 Manhattan–World Approaches

In recent years there has been growing interest in leveraging the Manhattan world assumption, in which a scene is modelled using surfaces oriented in three mutually orthogonal directions. This idea was originally proposed by Coughland and Yuille [10], who were interested in recovering camera orientation from a single image. It has since been used in a variety of tasks

including vanishing-point detection [41], single-view reconstruction [13, 20], and multiple-view reconstruction [25, 23]. In general, the Manhattan world assumption is appealing for scene understanding tasks as it reduces the size of the hypothesis class one must search over.

Cuboid Models

Hedau *et al.* [28] reason about indoor environments by modelling the scene as the interior of an axis-aligned cuboid. This is perhaps the simplest possible instantiation of the indoor Manhattan assumption, but the authors show that even this restrictive model can be helpful when interpreting and recognising objects in photos of indoor environments. Their approach proceeds in two stages. First, three mutually orthogonal vanishing points are estimated from observed line segments. Next, a cuboid is identified by casting two rays from each vanishing point.

The authors use a structured prediction model to learn how to select bounds for the cuboid. Given training examples they use train an SVM to rank cuboid hypotheses within a feature space defined over inputs (image features) and outputs (cuboids). This approach follows a similar methodology to the approach we propose in Chapter 7 of this thesis, though we learn within a much more general hypothesis class.

In later work [29] the authors connect the cuboid model of spatial boundaries to a model of objects within the room. Objects are modelled as axis-aligned cuboids resting on the floor. Several such cuboids are hypothesised, and the support for each is determined by an SVM trained with HOG features. The authors further show that by reasoning jointly about room boundaries and the objects within, the performance of both inference algorithms is improved.

Indoor Manhattan Scenes

Lee *et al.* [13] have investigated geometric context for the special case of *indoor* Manhattan environments, which are a sub-class of general Manhattan environments and have the following properties:

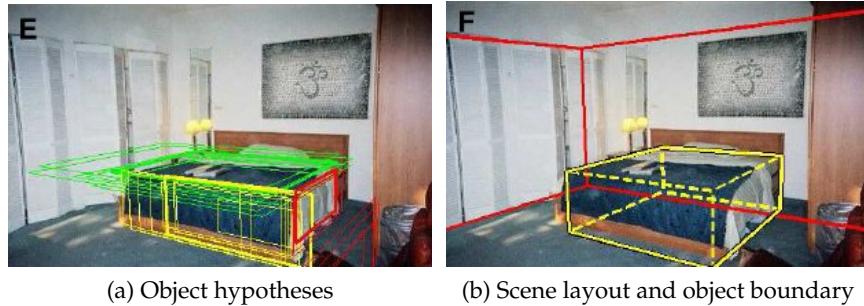


Figure 2.5: Results from the cuboid model of Hedau *et al.* [28, 29]. *Reprinted with express permission of Varsha Hedau.*

- There is a floor plane and a ceiling plane, which are parallel to one another and extend indefinitely in all directions.
- There are planar wall sections, which are orthogonal to the floor, extend all the way from the floor to the ceiling, and terminate in vertical boundaries.
- The wall sections are oriented in one of two mutually orthogonal directions.
- Many objects within rooms are aligned with the floor and/or walls and hence there will be many edges sharing vanishing points with floor, walls, and ceiling.

Although many real environments contain exceptions to these rules, the authors argue that their model is expressive enough to represent approximately or exactly many real-world scenes. This leads to a particularly simple model in which scenes are represented by a ground plane orientation and a set of wall segments.

Lee *et al.* show how such a model can be derived from a single image. They begin by sampling two pairs of lines in a RANSAC-like fashion to generate vanishing points. Each pair is checked for mutual orthogonality (using a prior on camera focal length) and for support amongst the other detected line segments. The intersection of the two pairs gives two vanishing points, from which the third can be derived. This results in a set of straight lines and their associated vanishing points.

The authors show that, given the assumptions above, any set of lines representing wall segments for which the associated vanishing points are known either give rise to exactly one Man-

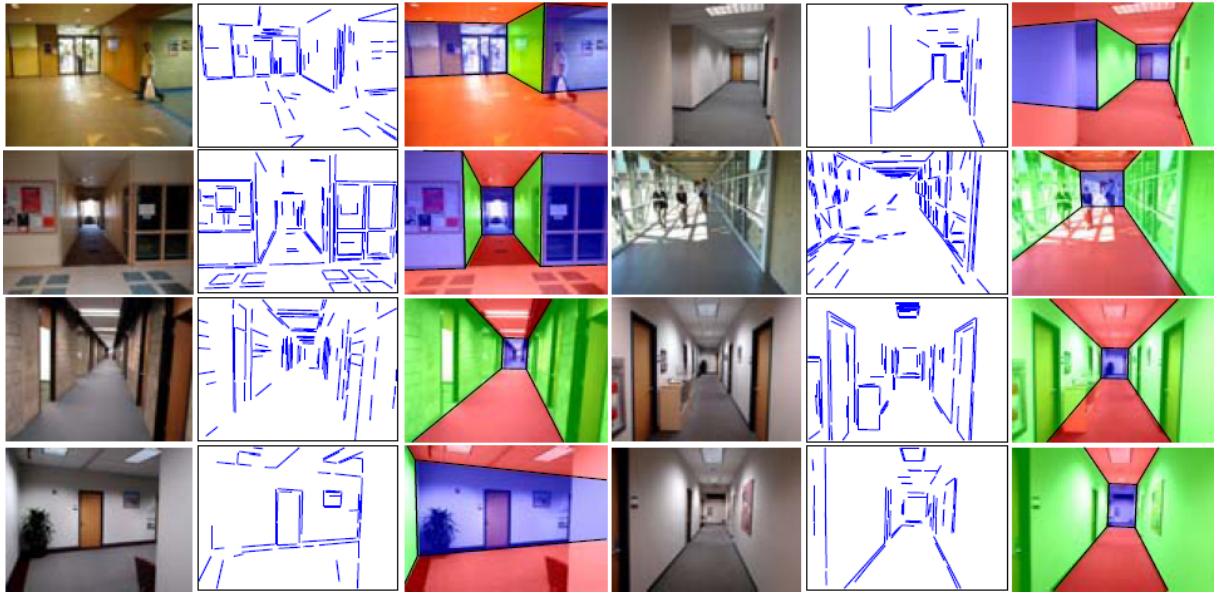


Figure 2.6: Inferred scene structure obtained for single images by Lee *et al.* [13]. Each triplet shows the original image, the detected line segments, and the final scene geometry. *Reprinted with express permission of David Lee.*

hattan world model or violate a set of easily–checkable rules. They therefore proceed to enumerate all valid hypotheses by running a branch–and–bound search over all combinations of line segments. This remains tractable because the validity check eliminates most combinations at an early stage of branching. Of the valid hypotheses, they choose the one which is maximally consistent with surface orientation estimates separately inferred from the image. Figure 2.6 shows some of the building structures their system was able to infer.

The second half of this thesis is concerned with models in this category.

Manhattan World Stereo

Furukawa *et al.* [25] used the Manhattan world assumption to improve the robustness of multiple–view reconstruction. Large textureless surfaces are common within indoor environments — for example walls, ceilings, and floors. Such regions challenge many 3D reconstruction techniques since identifying correspondences between views is challenging. To overcome this, Furukawa *et al.* built a 3D reconstruction system that leverages the Manhattan world assumption to extrapolate the layout of textureless regions from constraints imposed by edges

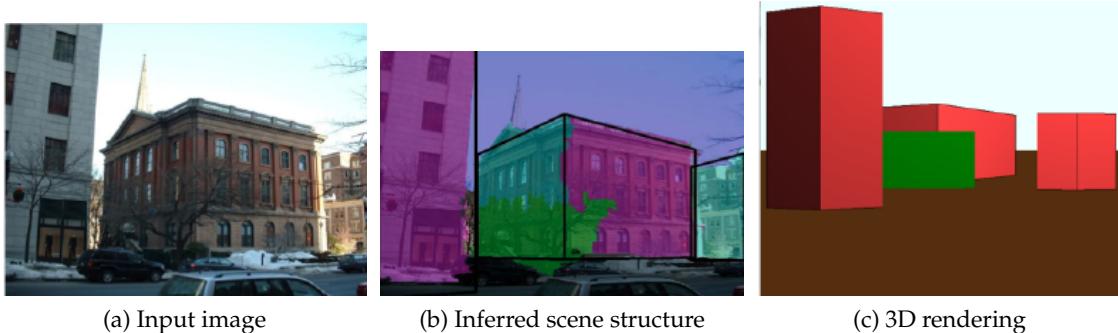


Figure 2.7: Results from the blocks world model of Gupta *et al.* [26]. *Reprinted with express permission of Dr Abhinav Gupta.*

and corners, which are easier to localise in multiple views. They model this task as an energy minimisation problem in which each pixel must be labelled as belonging to some axis-aligned plane. A smoothness prior ensures that system chooses the simplest arrangement of planes that match the observed image evidence.

Blocks World

As early as 1965, Roberts [54] proposed building scene representations up from cuboid building blocks. This idea has recently been revisited using modern probabilistic techniques by Gupta *et al.* [26]. Beginning with an empty ground plane, the authors iteratively add cuboids to explain image features such as corners and edges. Utilising a variety of stability and visibility constraints the authors show that they are able to model a variety of scenes.

Horizontal/upright Models

Delage *et al.* [14] presented early single-view reconstruction work that focussed on identifying the boundary between horizontal and upright surfaces. They modelled this floor/wall fracture as a dynamic Bayesian network over image pixels, for which efficient dynamic programming inference algorithms are well known.

Barinova *et al.* [3] adopt a Manhattan-like hypothesis class for outdoor scenes. Their models consist of a ground plane supporting a series of vertical facades, which they infer us-

ing a conditional random field over pixels. The key inference problem in their approach is to recover a polyline separating the ground plane from the series of vertical facades. They use a standard Expectation–Maximization algorithm for maximum–likelihood inference, and component–wise logistic regression to learn the CRF parameters from training data.

Felzenszwalb and Veksler [18] discuss a class of dynamic programming algorithms capable of maximum–likelihood image segmentation under various shape priors. One of the applications to which they apply their algorithm is single–image reconstruction.

Common to each of these approaches is a model that decompose into vertical segments running from left to right in the image. We take advantage of a similar decomposability within indoor Manhattan environments in the inference algorithm that we propose in Chapter 6.

2.2.4 Texture Recognition

Heitz and Koller [31] derived context from texture structure in images. Their model relates the presence of objects, which have specific boundaries, to the appearance of the nearby surfaces and foliage, which have no crisp notion of spatial support. This approach is motivated by the observation that object positions correlate with the appearance of their surroundings. For example, cars and bikes are likely to appear near road–like texture, whereas aeroplanes are likely to appear against a sky–like background.

They begin by running the normalised graph cuts algorithm of Ren and Malik [52] to produce roughly homogeneous regions called superpixels. Each such region is associated with a feature vector comprising various colour and texture statistics, which, in their model, is assumed to have arisen from a latent category variable, the intention being that superpixels will be clustered into meaningful groups like “road” and “sky”. Meanwhile, a set of candidate object detections is generated by invoking a standard object detector and taking all detections above a certain threshold. The “things” are related to the “stuff” by a set of observed variables representing spatial relationships such as “above”, “beneath”, and “next to”.

An advantage of the this model is that while object labels are required for training, no ex-

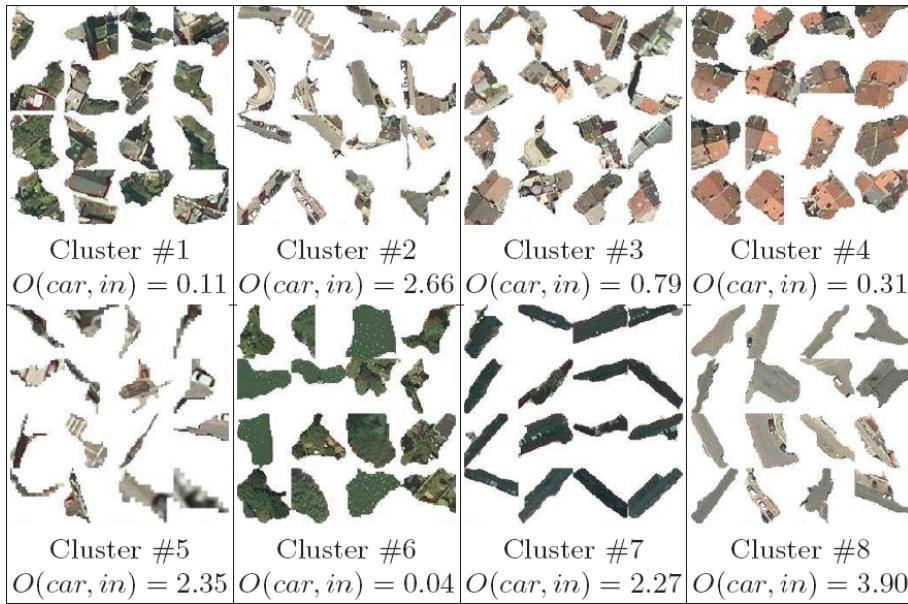


Figure 2.8: Superpixels clusters generated by the model of Heitz and Koller [31]. Shown below each panel is the odds ratios for a car appearing near a superpixel from that class. The odds ratio is higher for road-like regions than for foliage- or water-like regions. *Reprinted with express permission of Prof Daphne Koller.*

plicit segmentations or labels are required for the system to learn how to categorise the “stuff” regions since the system is capable of learning these unsupervised. The authors achieve this using the EM algorithm, which iterates between estimating the superpixel labels and optimising the appearance parameters for each label.

The authors also show how to learn a set of relationships that best describe the dependencies between objects and their surroundings. This involves augmenting the EM algorithm with a greedy search over possible relationships, iteratively adding the best candidate out of a pre-defined pool until convergence. This allows the system to adapt to the most salient relationships for a specific problem, which will differ between, say, images captured from satellites and the photos in the VOC datasets [16]. During evaluation both the object labels and superpixel labels are unknown, so exact inference is intractable. Instead, the authors use an approximate inference technique called Gibbs sampling. Figure 2.8 shows some example superpixel clusters along with their probabilistic relationship to car detections.

2.3 Context in Robotics

Cameras have long been recognised as a valuable sensor for mobile robotics. In addition to visual SLAM, cameras have been used in robotics tasks such as place recognition [12] and scene description [51]. Contextual reasoning has received little attention from the robotics community. This section reviews the literature that does exist on this topic. We divide the work into two categories: map-centric approaches, which integrate sensor data into a map and then reason from this representation, and ego-centric approaches, which organise sensor data as a time-indexed series of observations.

2.3.1 Ego-centric approaches

Martinez *et al.* [49] have demonstrated that a robot can learn to classify its environment into semantic categories such as “corridor” or “room” based on simple range data features. They employ a SICK laser, which gives a 360° scan of the scene within a single horizontal plane. They collect features such as the distance between successive beams, the average beam length, and the eigenvalues of the polygon formed by the beam endpoints, which are concatenated to form a feature vector at each time step.

To relate these features to semantic categories, the authors employ the AdaBoost algorithm [53] using linear classifiers as the weak learners. AdaBoost is a popular classifier that learns by incrementally adding rules that maximally correct the mistakes it has made so far. Martinez *et al.* show that they are able to differentiate rooms, corridors, and doorways at 89% accuracy using this method. Furthermore, in environments that have not been seen before the system obtains 82% accuracy. The results for one environment are depicted in Figure 2.9.

Stachniss *et al.* [59] extends this to use visual cues from a panoramic camera in addition to the laser range data. Stachniss runs an off-the-shelf object detector for each of several object categories that correlate well with location. The chosen objects include computer monitors, coffee machines, and soap dispensers. At each time step, the robot’s current location is categorised

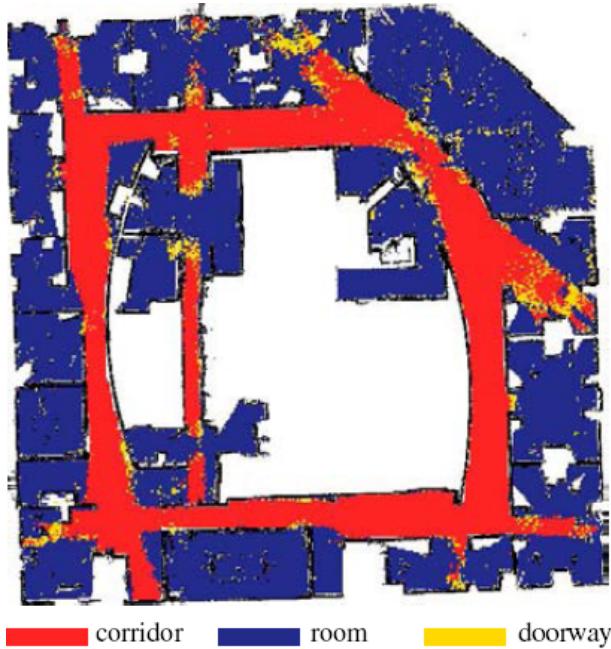


Figure 2.9: Semantic labels assigned by the range data classification system of Martinez *et al.* [49]. 82% of places were correctly classified, despite the environment not having been seen during training. *Reprinted with express permission of Dr Oscar Martinez Mozos.*

using the same AdaBoost classifier as described above, the only difference being that now the number of visual detections of each object type are appended to the AdaBoost input vector. This allows the classifier to select between visual and range data features (or combinations thereof) according to whichever is most salient for the task at hand.

The authors recognise that the robot’s location is highly correlated over successive time steps and so model the robot’s state as a Hidden Markov Model (HMM), with the transition probabilities estimated empirically. Stachniss *et al.* show that the addition of visual cues allow them to differentiate between rooms with a similar shape but different visual appearance (such as bedrooms and living rooms), whereas the original range–data–only approach of Martinez would fail in this case.

Posner *et al.* [51] show how to learn semantic labels such as “grass”, “foliage”, or “wall” for regions within urban environments. Like the systems described above, they reason from a combination of laser and vision features, including colour, location, and orientation properties. Unlike previous approaches they perform a quantisation step to form feature “words”.

Incoming images are segmented based jointly on an off-the-shelf superpixel algorithm and continuity boundaries in the laser data, after which the problem is reduced to determining the appropriate label for each segment. They relate the features for each region to semantic labels using a graphical model that incorporates observation likelihoods as well as a sensor model describing the probability of false positive and false negative observations. While a standard approach would be to assume independence between the feature observations for tractability, the authors note that the observations are far from independent in reality and instead employ the Chow Liu algorithm to find the best tree-structured approximation to the full joint distribution. Inference within this model is performed by computing the full posterior over labels.

Posner *et al.* further refine the labelling by relating the labels of neighbouring patches in an MRF framework. They introduce edges for both spatial and temporal consistency, the former being derived from adjacency between segments within the image, and the latter being derived by reprojecting laser points into consecutive frames. Node potentials are given by the segment classifier described above while edge potentials are learned empirically from hand-labelled training data. Sequential tree-reweighted message passing (TRW-S) is chosen for inference within the MRF due to its efficiency guarantees.

The authors show that their system is able to accurately label a range of outdoor environments with 8 categories (grass, tarmac, dirt path, textured wall, smooth wall, foliage, vehicle). Their classifier obtains the highest precision for the “grass” category (95.5%) and the lowest for “vehicles” (79.7%). Furthermore, their system is able to run in under 4 seconds, which is suitable for periodic on-line operation. An example labelling is illustrated in Figure 2.10.

2.3.2 Map-centric approaches

An alternative approach to deriving context in robotics applications is to integrate new measurements into a map, and then reason about semantics within the map representation. In general this approach enables stronger integration of measurements taken over several time steps, at the cost of relying on the ability to correctly build a map.

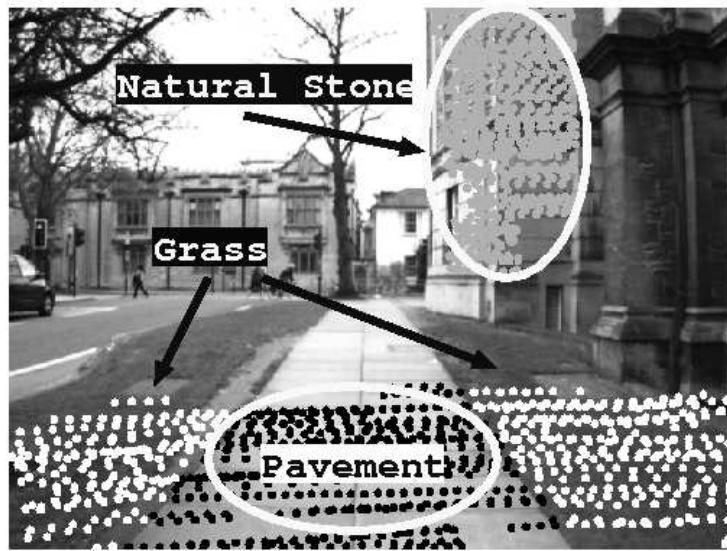


Figure 2.10: Semantic labels output by the system of Posner et al [51]. *Reprinted with express permission of Dr Ingmar Posner.*

Buschka and Saffiotti [7] have taken a map–centric approach to the problem of identifying room boundaries within indoor environments and recognising the resultant rooms. A series of laser range scans are fused into a 2D occupancy grid representing the probability that each cell is occupied by some object or boundary. Rooms boundaries are identified by applying dilation and erosion to the occupancy map, which are standard morphological filters from visual segmentation [24]. The authors demonstrate that this can be performed with fixed computational cost by discarding old parts of the environment as the robot moves through the environment.

The result of their algorithm is a series of “nodes” with topological connections between them, which correspond to the various rooms and corridors within the robot’s environment and the doorways that connect them. The authors proceed to characterise each node by the size and eccentricity (length to breadth ratio) of its bounding box. This gives contextual information in two senses: firstly, the identification of room boundaries allows reasoning in terms of rooms rather than the entire known environment, and secondly, the characterisation of room shapes allows differentiation between rooms and corridors and thus allows different interpretations of sensor data in these semantically distinct workspaces.

Vasudevan *et al.* [66] use an alternative map representation based around the location of objects. They argue that this matches human perception of space. In their maps, each “object” (actually

a SIFT landmark) occupies a separate coordinate frame, with uncertainty represented in the transformations between frames.

They identify doorways by running a line detector and testing various combinations of lines against a set of heuristics, enabling separation of the constituent rooms within an environment. This in turn allows per-room reasoning as was the case with the Buschka and Saffiotti system described above. The difference here is that Vasudevan *et al.* identify doorways directly from visual input whereas Buschka and Saffiotti use occupancy maps.

Vasudevan *et al.* show that this representation leads naturally to reasoning about place context. They argue that place categories (bedrooms, kitchens, bathrooms, *etc.*) can be identified by the objects within them, and hence that their object-centric maps provide the perfect setting for this form of contextual reasoning. Formally, they learn a class-conditional object likelihood by computing the number of times each object is observed in each type of places versus the total number of times the place category has been observed. They then assume independence between object observations and compute the posterior over place categories by multiplying out the likelihoods for each observed object.

2.4 Conclusions

Modern structure-from-motion systems generate accurate metric maps of an environment, and can do so robustly and efficiently using visual input alone [39]. The problem of deriving high-level semantic context from such sensor data has been approached by several authors. Although there is not yet a widely agreed-upon problem formulation, current work in this area shows that semantics are important for many robotics applications, and can be derived from a range of sensor types. Important contributions include that of Martinez [49] and Stachniss [59], who show that place categories can be obtained from photometric and geometric cues, and Posner [51], who shows that scene regions can be associated with semantic categories, again by combining vision and range sensors.

The contextual reasoning problem for single-image vision has received comparatively greater attention. Torralba's seminal gist descriptor [61] has led to the development of many types of contextual cues, including geometric [33, 55], textural [31], and model-based [13] approaches. Our work is motivated largely by the success of the single-image approaches discussed above; in this thesis we show how to extend these ideas to incorporate a moving camera and the geometric information that a SLAM system provides.

3

Appearance–Only Context Models

This chapter explores an approach to scene understanding that relates low-level image information directly to high-level variables describing scene categories and coarse geometry. We build on a well-known texture analysis approach in which pixels are identified with characteristic texture units known as textons. Extending previous work in texture analysis and image recognition, we show that this approach — which has previously been applied to recognition of uniform textures — extends to high-level scene understanding problem in which the input image is composed of many objects and surfaces. We propose a novel recognition model in which we measure the displacement between texton occurrences in an image rather than their absolute location or overall frequency. We apply our ideas to two classification tasks: place recognition and camera orientation classification. Comparisons with the gist descriptor of Torralba *et al.* show superior performance on both tasks.¹

¹This work was published in part in:
Flint, Murray, and Reid, “Learning Textons For Real-Time Scene Context”, in *Proceedings of the First International Workshop on Ego-Centric Vision*, 2009[22]

3.1 Introduction

One important aspect of scene understanding is the ability to differentiate between logical areas within an environment, such as rooms in a house. This problem is important in the domain of augmented reality because knowledge of the location of a user will give a strong indication as to the activities the user might undertake and the objects with which they are likely to interact.

Outside this specific application area, place recognition can be used as a generic source of context for further scene understanding tasks. Object recognition, semantic segmentation, and semantic reconstruction are all likely to benefit from a prior on place categories, since such information would correlate well with the location of objects, the function of surfaces, and the geometric structure of the environment.

The place recognition problem we propose in this chapter associates many images of a logical region — rooms within a house in our case — with a single place category. This means that there may be image pairs observing no common 3D locations, yet are associated with the same category, and hence the system will be expected to recognise them as associated with the same logical place. We also tackle the problem of identifying the approximate tilt of the camera from a single image. This problem requires the categorisation of images into 3 categories: “upwards-facing”, “downward-facing”, and “straight-facing”.

We pursue an approach motivated by the texton model developed by Julesz [37] and introduced to the computer vision community somewhat later [69]. We show that low-level texture elements can be related to the high-level variables of interest to us, and that doing so yields state-of-the-art results for both scene understanding problems.

The remainder of this chapter is organised as follows. First we introduce the notion of textons, which form the basis of our approach. Next we motivate the use of textons for scene understanding problems in general by exploring the relationship between textons and scenes. We then describe our probabilistic model relating textons to place recognition and camera classification in detail. Next we present an empirical comparison between our system and the gist descriptor of Torralba *et al.*, followed by discussion and concluding remarks.

3.2 Background

The notion of textons as atomic texture elements was born in the neuroscience community when in 1981 Julesz [37] introduced textons as part of his theory of human visual attention. Julesz defined textons as elongated blobs, line terminators, and line intersections. Several researchers have proposed definitions of textons for use in computer vision applications (see [69] for an extensive discussion), including both topological and statistical descriptions. We follow the statistical account first proposed by Malik *et al.* [47]. Under this definition, an image is passed through a filter bank, producing a set of response images. Each pixel is then associated with a feature vector that contains each filter's response for that pixel. The feature vectors are then clustered and the resultant cluster centres become the texton exemplars.

At evaluation time the input images are passed through the same set of filters, and each pixel is again associated with a feature vector containing the filter responses at that point. Next, each pixel is labelled according to the index of the texton exemplar (from the training phase) closest to it in the L2 sense. The image is henceforth represented as an array of texton indices (the “texton map”); the remainder of the image data is discarded.

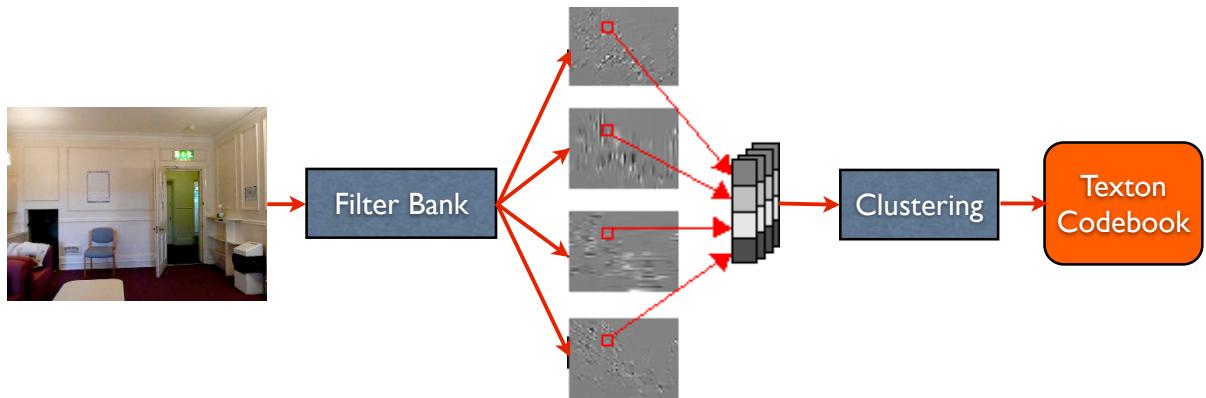


Figure 3.1: The texton generation process.

In the past textons have been successfully used to classify close-up photos of materials such as wood, paper, and glass. Varma and Zisserman [65] tackled this problem using the texton model discussed above. After forming the texton map they proceed by counting the occurrences of each texton, with the resultant histogram over texton frequencies forming the model

by which input images are matched to their category. In their work, Varma and Zisserman apply a nearest neighbour approach to classify the models, using the χ^2 statistic to measure distances. Representing data by a histogram over quantised features in this way has since become known by the terminology “bag-of-features” or “bag-of-words”.

3.3 Textons for Scene Understanding

Textons are a widely used and well-understood within the computer vision community [69, 65, 47]. However, their use has mostly been limited to low-level image analysis tasks, such as texture classification. We propose to use textons as the basis for high-level reasoning tasks including scene classification. In the remainder of this chapter we will develop a novel model by which to utilise textons during inference. We begin in this section by motivating the use of textons for this application.

Many place recognition systems rely upon an interest point detector and descriptor to summarise input images [17, 12]. This has proven effective for outdoor environments as well as indoor environments that contain reasonably distinctive landmarks. However, these systems are fundamentally limited to a feature-centric view of the world in which only local information about interest points is utilised.

We believe that there is un-leveraged information in the texture structure within images, and that this can be used for scene understanding tasks such as place recognition. Many images of indoor scenes contain extremely poor visual information, particularly small, empty environments like corridors and foyers. Figure 3.2 shows some examples of these. Yet despite this information poverty, humans are capable of deducing much from these images. For example, consider the image at the right of Figure 3.2: there is barely one location here that would yield a useful SIFT feature, yet a human can identify the part of the environment that the image represents (a corner between wall and ceiling), and a human familiar with the environment can easily identify the room in which the image was captured. We think it is clear that, in this case, humans are using a holistic understanding of the image in which the edge structure together

with the texture of the surfaces is used to understand the image. We propose to use textons to leverage this valuable information.



Figure 3.2: Frames with low visual salience.

To investigate whether textons provide useful information about scene structure, we collected video sequences of an indoor environment and examined the textons generated by the algorithm described above. Our dataset consisted of 10,555 frames from 5 rooms in a hostel. Some qualitative findings are described in the following sections.

3.3.1 Textons Select Salient Image Elements

Figure 3.3 shows 25 textons generated for this dataset, in order of their frequency of occurrence. The first 7 correspond to untextured regions of the image — *i.e.* patches with near-uniform intensity. This is expected since the majority of pixels lie within object or region boundaries, where either the texture is too fine for the camera to detect (a carpet, for example) or there simply is no texture (a white wall, for example).

The next most frequent textons are those corresponding to edges and bars at various orientations. Many image understanding algorithms explicitly employ a line detector [24], whereas in this case the use of textons has allowed the system to learn to identify these elements unsupervised.

At the other end of the frequency distribution, the least numerous textons are those corresponding to image structures such as junctions and line endpoints. These structures are also sought explicitly in many image understanding algorithms [24], whereas the use of textons selects these structures automatically and unsupervised.

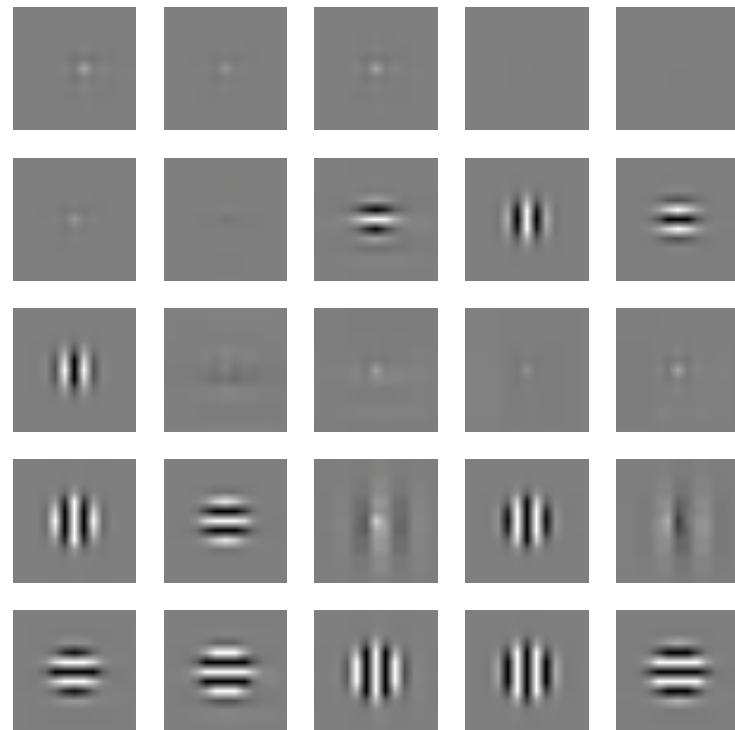


Figure 3.3: Visualisation of textons identified for our dataset during the offline training stage. Each image shows a weighted sum over the filter kernels in our filter bank, with the weights set to the coefficient from one identified cluster centre. Each image therefore shows the “ideal” patch corresponding to one of the textons added to the codebook. The textons are sorted by decreasing frequency from top to bottom, left to right.

3.3.2 Textons Focus Attention on Salient Image Regions

Figure 3.4 shows the frequency of each texton within our entire dataset. The most frequent textons at the left of the graph account for a disproportionately large number of pixels, whereas the textons towards the right account for a tiny minority. We have just seen that it is precisely these minority textons that correspond to the image elements that are most useful in image understanding, so in this sense the use of textons has automatically focused attention on the small fraction of pixels representing the most salient image structures.

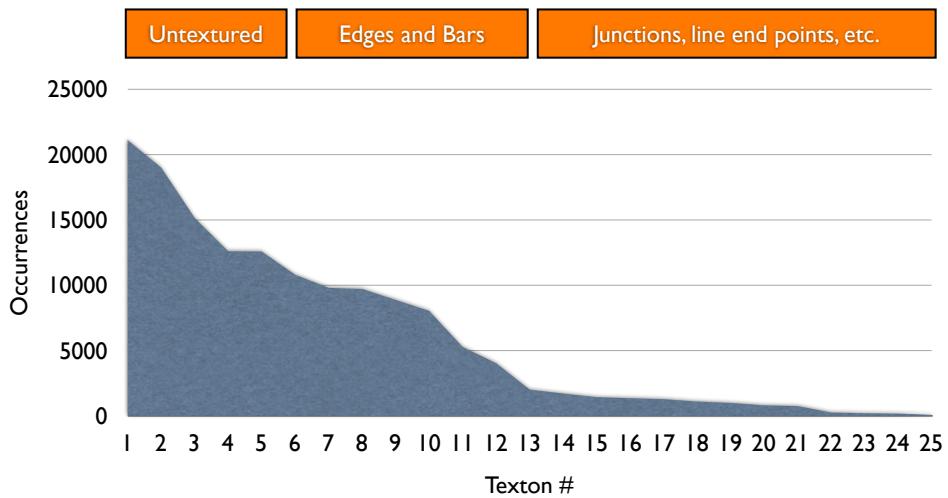


Figure 3.4: Frequency distribution of textons. The texton generation process assigns more textons to describing salient but rare image structures such as junctions and line end points than it assigns to frequent but non-discriminative image regions such as textureless patches.

3.3.3 Textons Correlate With Scene Structure

To further explore the relationship between textons and scene structure we divided all frames within our dataset into three orientation categories: “straight” (for which the camera’s optical axis was within $\pm 22.5^\circ$ of the horizontal), “up” (above 22.5°), and “down” (below 22.5°). Next we computed an average occupancy map for each texton within across each category. Several per-texon heat-maps are shown in Figure 3.5. The first two rows show heat-maps for textons that correspond roughly to “floor” and “wall/ceiling”. We can see that these reflect the image regions in which we would expect to find these scene parts given the various camera

orientations.

Perhaps the most instructive example of the texton/scene structure relationship, however, is that of the textons that represent edge elements, two of which are shown in the lower two rows of Figure 3.5. Consider the third row in this figure: The stratification evident in the heat-map for downward-facing frames indicates a common intersection somewhere above the image, whereas that for the upward-facing frames indicates an intersection below the image, and in the forward-facing frames the lines are close to parallel. This, of course, is exactly what we would expect from our understanding of projective camera geometry, but here our system has automatically and without supervision captured these geometric constraints (or at least some statistical form of them).

These illustrations are simply indications that textons might provide salient information of value for scene understanding. We describe our model that explicitly relates these to each other in the next section.

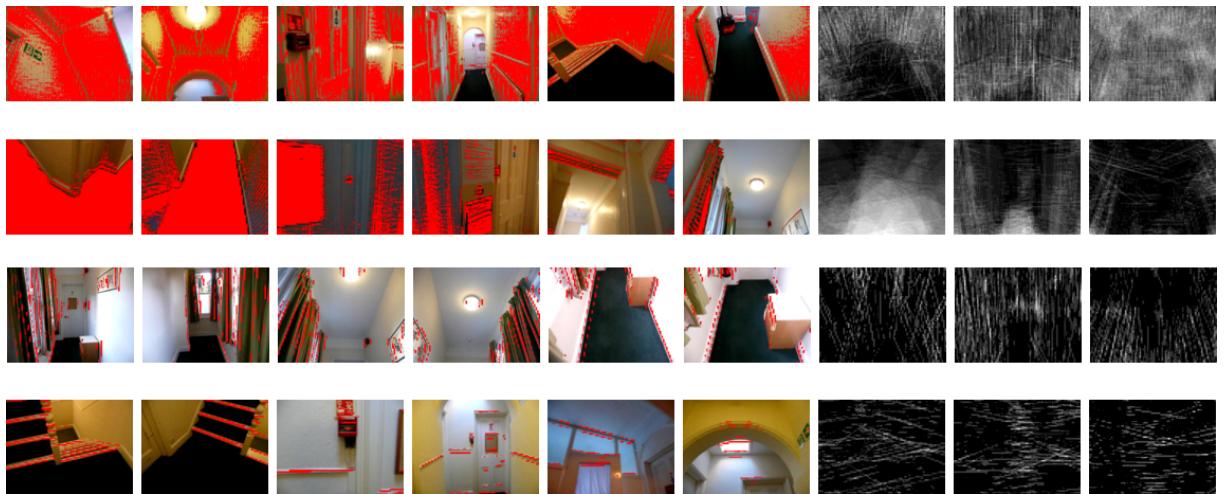


Figure 3.5: Four example textons generated unsupervised for the camera orientation classification problem. From top to bottom the textons represent roughly “wall or ceiling”, “floor”, “vertical edge”, and “horizontal edge”. The six columns on the left show examples of where the texton was found. The three columns on the right show the average occupancy map over our dataset for images taken from an upwards-facing, horizontal, and downwards-facing camera (from left to right in the figure). The layout of the textons correlate strongly with camera orientation, which illustrates how our system is able to distinguish between camera orientations based on texton layout.

3.4 Learning and Inference

Having motivated the use of textons for image understanding we now describe our probabilistic model that relates textons to scene categories. For an input image I , we wish to discover the associated place category c . For each category we are given a set of example images at training time. We wish to model categories c according to the pixel locations in which textons appear within images associated with each category. One popular approach is the bag-of-features model [36] but this would remove all information about the locations of the textons, which as we have seen is a principal source of salient information. Instead we propose a new bag-of-texton-pairs model in which an image is represented as a collection of observed texton pairs $\{(t_i, t_j, s_{i,j})\}$ where t_i and t_j are texton labels and $s_{i,j}$ is the displacement between the image locations at which they were observed. By considering only displacements and not absolute pixel locations in our model we gain some robustness to camera orientation.

For an image I containing N pixels there are N^2 such pairwise observations. Our class-conditional likelihood is

$$P(I | c) = \prod_{i=0}^N \prod_{j=0}^N P(t_i, t_j, s_{i,j} | c) \quad (3.1)$$

where we have assumed independence between observations for tractability. We compute the likelihood (3.1) by estimating the joint distribution $P(t_i, t_j, s_{i,j}, c)$ using a histogram. We could have used Parzen windowing [50] for the continuous variable $s_{i,j}$ but due to the very large number of samples we obtain for each image, we found this to be unnecessary.

For images of reasonable size the cost of enumerating all N^2 texton pairs is prohibitively expensive. We overcome this by overlaying a $M \times M$ grid on the image and counting the occurrences of each texton within each grid cell. We then enumerate all pairs of grid cells and evaluate the texton pairs in aggregate. Hence for grid cells C_a and C_b containing n_i^a and n_j^b instances of texton t_i and t_j respectively, we evaluate $n_i^a n_j^b$ instances of the observation $(t_i, t_j, s_{a,b})$ where $s_{a,b}$ is the distance between the centres of the grid cells. We have lost some precision in the texton locations since each texton is effectively moved to the centre of the grid cell containing it, but our experiments show that we are still able to capture sufficient salient information.

During training we evaluate these aggregated observations by multiplying the entry we make in the histogram by $n_i^a n_j^b$, and during evaluation the aggregate observations correspond to multiplications in the class-conditional log likelihood

$$\log P(I \mid c) = \sum_{a=0}^{M^2} \sum_{b=0}^{M^2} \sum_{i=0}^K \sum_{j=0}^K n_i^a n_j^b \log P(t_i, t_j, s_{a,b} \mid c) \quad (3.2)$$

In both cases the aggregated observations can be evaluated in a single step so the complexity is reduced from $O(N^2)$ to $O(M^4 K^2)$. In practice we found that setting $M=8$, $K=25$ was sufficient to capture much of the salient image information, while allowing our system to run at video frame rate.

3.5 Place Recognition

We applied our system to the problem of place recognition. Our data set consisted of several video sequences captured in a hostel using a low-quality camera with a resolution of 320×240 , which moved rapidly with the user's upper body. The sequences involved frequent motion blur and rapid variations in camera orientation.

We labelled each frame with the place that it was captured in. There were five labels: bedroom, kitchen, common room, garden, and corridor. We gave all frames captured in corridors the same label. This experiment does not correspond to place *category* recognition since most of the labels included frames from only one place instance. However, it is harder than strict landmark-style localisation because, as shown in Figure 3.6, many images with the same label contain non-overlapping views of the room they were captured in, yet the system is expected to recognise all of them as belonging to the same place.

We compared our system with the gist descriptor of Torralba *et al.* and a K-nearest-neighbours baseline, using vectorised grayscale images as feature vectors for the latter. For the gist descriptor we used the same Gabor filter bank that we used in our own system and we estimated the class-conditional likelihood in feature space by building Gaussian mixture models with the



Figure 3.6: Four frames with the “bedroom” label. There are almost no overlapping scene parts but the system is required to (and did successfully) recognise each of them as part of the same place.

Gaussians constrained to be spherical, exactly as described in [61].

Initially we used 230 frames for training and 490 frames for evaluation (our training and evaluation sets were taken from separate sequences). The results from this experiment are shown in the middle row of Table 3.1 and in Figure 3.7. Our system outperformed Torralba’s by a large margin. We suspected that the poor performance of Torralba’s system was due to the training data not sufficiently populating the 32-dimensional feature space. This exemplifies one of the major advantages of our approach: the ability to learn from limited training data. However, to show that this is not the *only* advantage of our approach we ran auxiliary experiments with larger and smaller training sets. When the training set was enlarged our system outperformed Torralba’s by a significant but smaller margin, and when the training set was decreased, the performance of our system decreased only slightly whereas Torralba’s system was unable to estimate the Gaussian mixture due to the sparsity of training samples. The latter case corresponds to just 20 examples per label. These results are shown in the top and bottom rows of Table 3.1.

Figures 3.8 and 3.9 show positive and negative results from our system. Note how our system recognises images containing disjoint views of a room as belonging to the same place.

3.6 Camera Orientation Classification

In this section we show how our system can deduce a coarse camera orientation from a single image. We are interested only in the tilt of the camera with respect to the ground plane. Our intention is to make a rapid but coarse estimate of camera orientation. We pose the problem as

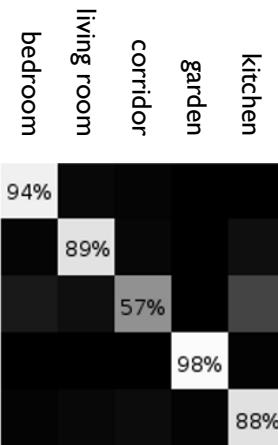


Figure 3.7: Confusion matrix for place recognition. This figure corresponds to the bottom row of Table 3.1.

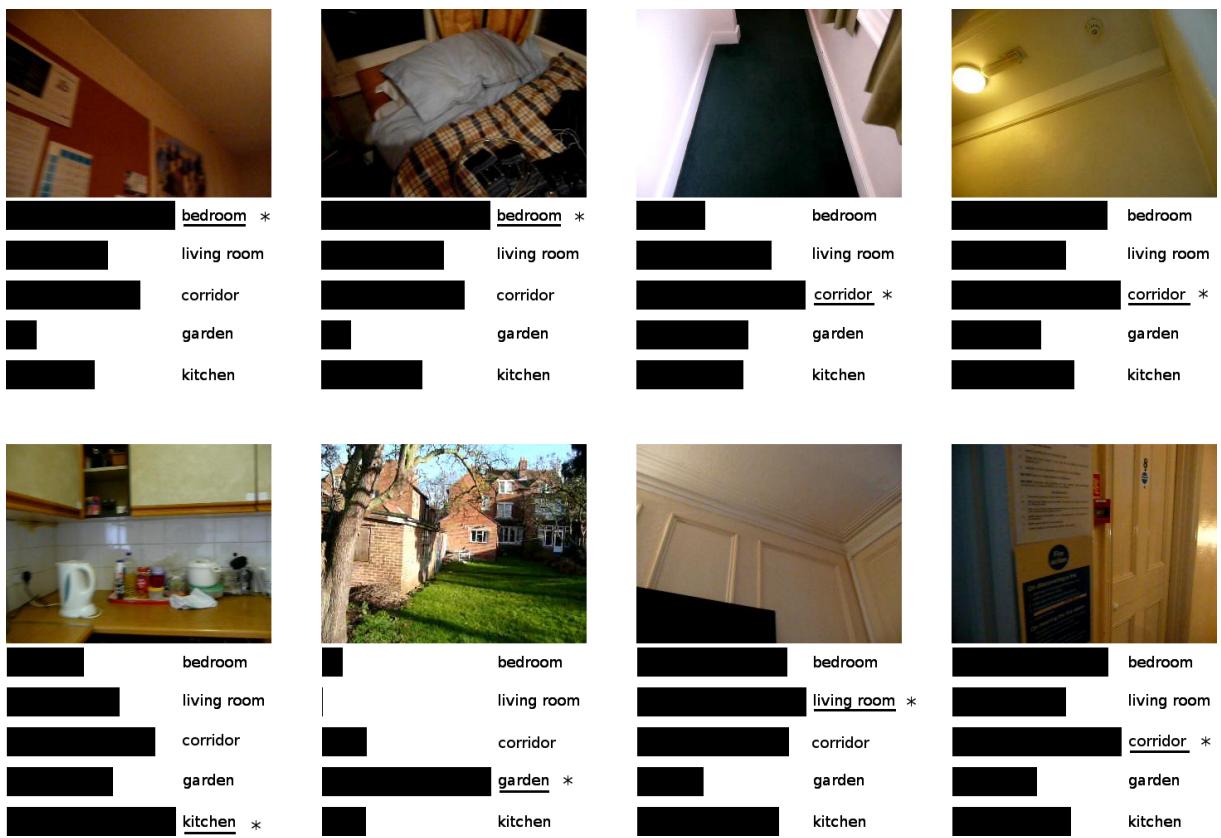


Figure 3.8: Example frames for which our classifier produced the correct output. The ground truth label is underlined and the output from our system is starred. We show the posterior distribution over class labels in black. Note the variation between images with the same label, and the low quality of many images.

No. training frames	This chapter	Torralba <i>et al.</i>	Nearest neighbours
103	81%	—	45%
230	83%	62%	52%
565	85%	70%	55%

Table 3.1: Recognition rate for the place recognition problem with varying numbers of training examples. The reported accuracy is the mean over the on-diagonal elements of the confusion matrix. The left column reports the total number of training examples for all categories. For the experiment with 103 training examples we were unable to estimate the Gaussian mixtures required for the system of Torralba *et al.* due to the sparsity of the training examples. The nearest neighbours baseline was computed by resizing images to 64 by 64 pixels and comparing raw pixel intensities in the L2 sense.

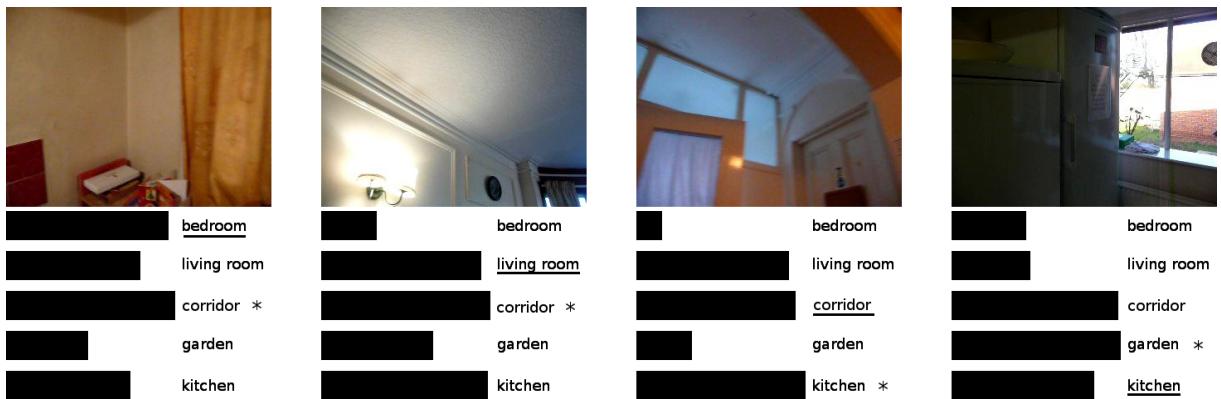


Figure 3.9: Example frames for which our classifier failed. See caption of Figure 3.8.

a classification task with three labels: “up”, “straight”, and “down”. The “straight” label represents images taken with the camera axis parallel to the ground plane, plus or minus 22.5° , and the “up” and “down” labels represent all orientations facing further upwards or downwards respectively (see Figure 3.10).

We captured three sequences in which the camera orientation was fixed within one of the above orientation ranges. We included footage from five different places (the same rooms used in the previous section) but we labelled the frames according to orientation only. This represents a difficult classification task because the system must learn properties that correlate with camera orientation but are not tied to the appearance of a particular room. We then trained our system to distinguish between the three orientation categories as in the previous section.

We again compared with the “gist” of Torralba *et al.* and a KNN baseline. We ran auxiliary experiments with an enlarged training set as in the previous section. The results of these ex-

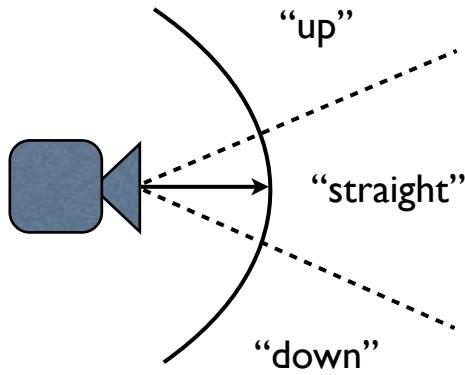


Figure 3.10: Definition of the labels for camera orientation recognition.

periments are shown in Table 3.2 and Figure 3.11. Our system again outperformed both other classifiers by a significant margin. Some example frames for which our system correctly identified the camera orientation are shown in Figure 3.12. Of particular interest is our system’s ability to generalise across images taken with the same camera orientation at several different locations.

No. training frames	This chapter	Torralba <i>et al.</i>	Nearest neighbours
88	70%	61%	59%
728	79%	63%	59%

Table 3.2: Camera orientation classification results using small and large training sets. The reported accuracies are the average of the on-diagonal entries in the confusion matrix. In this case we were able to estimate the Gaussians for Torralba’s system using only 88 training examples because there were fewer categories than in the place recognition problem. The nearest neighbours baseline was computed by resizing images to 64 by 64 pixels and comparing raw pixel intensities in the L2 sense.

3.7 Computational Concerns

The most computationally demanding aspect of our system is the convolutions needed to generate the pixel features. In this section we describe several optimisations for the convolution operation. Timing results are shown in Table 3.3 at the end of this section.

	up	straight	down
up	90%		
straight		72%	
down			72%

Figure 3.11: Confusion matrix for camera orientation classification. This figure shows results corresponding to the bottom row in Table 3.2.

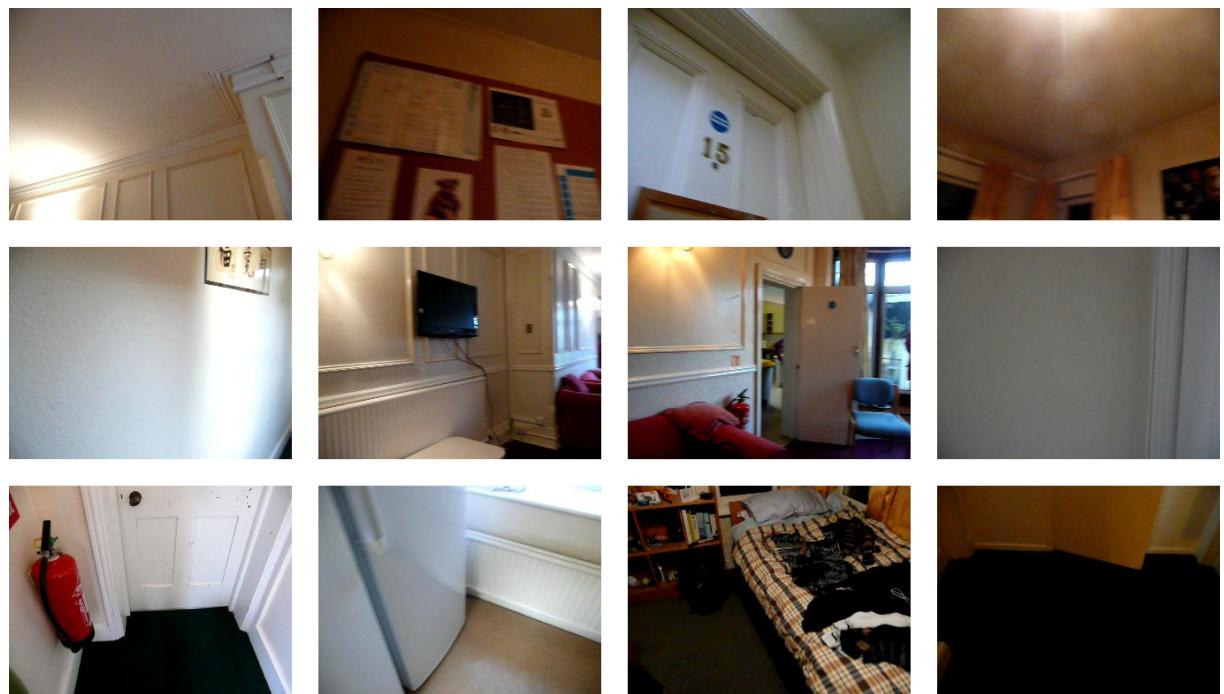


Figure 3.12: Twelve frames for which our system correctly identified the camera orientation. From the top to bottom the rows contain images from the “up”, “straight”, and “down” classes.

3.7.1 Separable Kernels

The Gabor function, which we use to generate pixel features, in its canonical form is a complex-valued function and can be trivially separated. However, in our work we use just the real part,

$$H_{real}(x, y) = g_2(x, y) \cos(kx \cos \theta + ky \sin \theta), \quad (3.3)$$

where g_2 is the two-dimensional Gaussian function. This can be written as the difference between two separable functions as

$$H_{real}(x, y) = (g_1(x) \cos ax)(g_1(y) \cos by) - (g_1(x) \sin ax)(g_1(y) \sin by) \quad (3.4)$$

where g_1 is the one-dimensional Gaussian function and we have used the cosine expansion formula. We implement this by running two separated convolutions and then taking their difference, which is significantly faster than performing the original convolution.

3.7.2 Parallelization over filters

Generating the pixel features requires convolving an input image with 12 different filters. To leverage the parallelization capabilities of modern multi-core CPUs we execute the convolutions in parallel where possible. This approach allows parallelization within a single scale, but requires synchronisation upon completion of each scale due to the down-sampling operation.

3.7.3 Parallelization over pixels

Modern graphics hardware allows efficient parallelization of small-scale computations such as per-pixel operations. This is well suited to performing convolutions since each output pixel is functionally independent of all others. To leverage this we implemented convolutions using a C-like language designed specifically for programming graphics hardware. We note that further improvements may be possible since under the current implementation, much

time is spent transferring data between the CPU and GPU, in comparison to which the time spent performing the convolutions is small. This bottleneck could be partially avoided if we performed the texton labelling in the GPU and transferred only the final texton map back to the CPU.

3.7.4 Timing results

Results from timing evaluations are shown in Table 3.3. The GPU strategy outperforms all others. It improves over the baseline strategy by a factor of 5 for the 320×240 image and by a factor of 6 for the 640×480 image.

It is interesting to note that when the image is enlarged to include four times as many pixels, the three CPU-based strategies degrade by a factor very close to four, whereas the GPU implementation degrades by a factor closer to three. This indicates that this task saturates CPU performance but that a significant overhead remains in the GPU implementation. This matches our observation in the previous section that transmitting data between the CPU and GPU consumes much time, and that further performance improvements are possible.

Strategy	320×240		640×480	
	Per Frame	Frame Rate	Per Frame	Frame Rate
No parallelization	115.0ms	8.69Hz	463.6ms	2.16Hz
Filter-parallel	38.0ms	26.3Hz	148.0ms	6.76Hz
Image-parallel	32.1ms	31.2Hz	125.6ms	8.02Hz
Pixel-parallel (GPU)	23.5ms	42.5Hz	78.1ms	12.80Hz

Table 3.3: Timing results for the four parallelization strategies. Each strategy was evaluated for two image sizes. All strategies utilise separated filters. Results are averages over 10 invocations.

3.8 Conclusions

In this chapter we have shown that texture structure can be leveraged to deduce high-level information about image scenes. We have motivated the use of textons for this purpose with

several qualitative examples, and have presented results from two in-depth experiments. In both cases our system based on relative displacements between textons obtains highly encouraging results. Furthermore, by utilising graphics hardware our system is capable of operating at video frame rate.

4

The Geometry of Indoor Environments

This chapter develops the idea that comprehensive scene understanding at the level of objects, actions, and surfaces requires at least a rudimentary understanding of the high-level geometry of the environment. There are many ways to capture such coarse geometry; here we argue that the indoor Manhattan representation is an excellent choice in terms of the salient geometric information it captures, its simplicity, and the elegant inference procedures it permits. This chapter focusses on analysing the model itself, leaving algorithms and empirical results to later chapters. This constitutes the first comprehensive definition and analysis of indoor Manhattan environments in the literature (though not the first account *per se* [13]). The primary contributions of this chapter are a derivation of the “vertex” and “seam” parametrisations, and the description of several properties of these parametrisations that will prove crucial to the development of the following chapters.¹

¹ This work was published in part in:
Flint, Mei, Murray, and Reid, “A Dynamic Programming Approach To Reconstruction Building Interiors”, in Proceedings of the 2010 European Conference on Computer Vision [20]



Figure 4.1: Two examples of indoor Manhattan environments.

4.1 Introduction

The preceding chapter described an approach to scene understanding in which raw image features were connected directly to high-level quantities such as scene categories. This chapter begins the development of a more sophisticated model in which pixel-level features are connected to high-level quantities via an unobserved intermediate level representing scene geometry. We focus on recovering this intermediate representation, leaving the application to specific classification tasks to future work. There are a number of reasons to see geometry as central to scene understanding. For an intuitive demonstration, consider the information conveyed by the red, green, and blue regions in Figure 4.1. These regions have very low fidelity (the entire model is defined by just a handful of polygons), yet it is clear that they convey considerable information about the likely locations of various objects, the scale at which those objects would appear, and how they would interact.

There are many choices for the representation of the intermediate-level variables; the following desiderata summarise the requirements relevant to our goals.

1. **Salience.** Of all the information present at the image level, the intermediate variables should capture as much of the information that is *relevant* to inferring the high-level quantities of interest as possible.
2. **Compression.** The intermediate representation should discard as much irrelevant infor-

mation as possible. All else equal, we prefer succinct representations of the world for efficiency reasons. See MacKay [46] for a discussion of the connection between compression and learning.

3. **Tractability.** All else equal, we prefer representations that lead to simple and efficient inference algorithms. A simple model for which inference is tractable may be preferable to a slightly more expressive model in which one must rely on approximate inference procedures with poorly-understood convergence properties.
4. **Simplicity.** All else equal, we prefer representations consisting of quantities that are intuitive to humans.
5. **Generality.** The chosen representation should be applicable to a wide variety of real-world environments.

We have chosen to work with the indoor Manhattan representation, in which the world is built up from a floor plane, a ceiling plane, and a sequence of vertical wall segments. This representation was originally proposed by Lee *et al.* [13]. Indoor Manhattan environments are a sub-class of general Manhattan environments, which simply specify surfaces in three mutually orthogonal directions. Indoor Manhattan scenes satisfy the desiderata above for the following reasons.

- **Compression and salience.** As described above, the geometric entities present in the indoor Manhattan representation correlate well the high-level quantities of interest for scene understanding. The models shown in Figure 4.1 consume just a few hundred bytes, yet capture much of the information present in the original image data, which consumes three orders of magnitude more space. The point here is not to save disk space, but to demonstrate that indoor Manhattan models efficiently capture salient information.
- **Simplicity.** Indoor Manhattan models are composed of quantities that have natural semantics, being associated with entities such as the floor, walls, and ceiling of an environment.

- **Generality.** Indoor Manhattan models can represent many environments exactly or approximately; even those that do not conform exactly to the assumptions of the model. We expand on this later.
- **Tractability.** Indoor Manhattan scenes have special geometric structure that permits an elegant “seam” representation. This in turn leads to efficient and exact inference algorithms for a range of sensor models, and an attractive learning algorithm. This are the subject of the ensuing chapters.

By adopting the indoor Manhattan representation we are *not* restricted to working with *ideal* indoor Manhattan environments — environments free of clutter, occlusions, windows, or other deviations from our assumptions. Rather, the point is to take account of such deviations in the probabilistic model relating image features to scene structure. For example, the model we present for depth measurements in Chapter 6 explicitly marginalises over the possibility that a given depth measurement corresponds to a clutter object, an object observed through a window, or an erroneous measurement. Thus we cast the Manhattan scene geometry as a latent variable to be inferred in the presence of noise, clutter, and occlusions, rather than as a complete description of everything we might observe. Empirical results in forthcoming chapters show many examples of our system working in real-world environments that are very far from ideal indoor Manhattan scenes, containing poor lighting and noisy images in addition to clutter, windows, and non-Manhattan-aligned surfaces.

However, this chapter focusses on analysing the representation itself, and so is concerned with ideal indoor Manhattan scenes. The contributions of this chapter are (1) the first comprehensive definition of the indoor Manhattan assumption; (2) two novel representations for indoor Manhattan scenes; and (3) a number of minor results connecting these representations and providing other insights crucial to the ensuing chapters. As mentioned above, this chapter does not contain any experimental results, though it does provide the conceptual infrastructure that the remainder of this thesis builds on. We begin by defining indoor Manhattan scenes, then we discuss our first representation in terms of floorplans, followed a description of the “vertex” and “seam” parametrisations that will be used in following chapters. Section 4.5 contains a

number of minor deductions of relevance to the ensuing chapters, then we close with a brief conclusion.

4.2 Indoor Manhattan Environments

Let a reconstruction G consist of a set of polygons P_i with vertices $\mathbf{v}_j \in \mathbb{R}^3$. An *indoor Manhattan reconstruction* is a reconstruction G with the following properties.

1. *Each polygon is oriented in one of three mutually orthogonal directions.* Formally, for each pair of polygons $P_i, P_j \in G$ with unit-length normals $\mathbf{n}_i, \mathbf{n}_j$,

$$\mathbf{n}_i \cdot \mathbf{n}_j \in \{0, 1\} \quad (4.1)$$

2. *There is a floor plane and a ceiling plane.* Formally, there exists some direction \mathbf{n}_v and a pair of real numbers z_f, z_c such that each polygon $P \in G$ with normal $\mathbf{n} = \mathbf{n}_v$ has vertices \mathbf{v} satisfying

$$\mathbf{v} \cdot \mathbf{n}_v \in \{z_f, z_c\} \quad (4.2)$$

3. *Vertical walls extend from floor to ceiling.* Formally, if $\mathbf{n}_i \neq \mathbf{n}_v$ then we say that $P_i \in G$ is a *vertical* polygon. This condition is satisfied if and only if each vertex \mathbf{v} of each vertical polygon satisfies

$$\mathbf{v} \cdot \mathbf{n}_v \in \{z_f, z_c\} \quad (4.3)$$

4. *The edges of walls are vertical.* Formally, if $\mathbf{v} \in P$ and $\mathbf{v} \cdot \mathbf{n}_v = z_f$, then $\exists \mathbf{u} \in P$ such that

$$\mathbf{u} \cdot \mathbf{n}_v = z_c \quad (4.4)$$

and

$$\mathbf{u} = \mathbf{v} + \lambda \mathbf{n}_v \quad (4.5)$$

for some $\lambda \in \mathbb{R}$.

We assume a pinhole camera mapping homogeneous world coordinates $\mathbf{X} \in \mathbb{R}^4$ to homogeneous image coordinates $\mathbf{x} \in \mathbb{R}^3$ according to

$$\mathbf{x} = K \begin{bmatrix} R & \mathbf{t} \end{bmatrix} \mathbf{X}, \quad (4.6)$$

where K is a 3×3 homography, R is a 3-dimensional rotation matrix, and \mathbf{t} is a 3-dimensional translation.

We define an indoor Manhattan *scene* as a reconstruction together with a camera viewpoint, subject to the following additional constraints.

1. The camera centre \mathbf{c} is located between the floor and ceiling planes,

$$z_f < \mathbf{c} \cdot \mathbf{n}_v < z_c \quad (4.7)$$

2. The reconstruction is closed relative to the camera viewpoint. Formally, for each viewing direction \mathbf{x} there is some polygon $P \in G$ containing a point \mathbf{X} that projects to \mathbf{x} .

In general we will assume that scenes consist only of those polygons visible to the camera. Since reconstructions consists of a finite number of polygons and our camera model is linear, it can be shown that an indoor Manhattan reconstruction truncated to a view frustum remains an indoor Manhattan reconstruction as defined above.

4.3 Floorplans

The polygonal representation for indoor Manhattan environments is inconvenient for our purposes. A more useful representation is a *floorplan*, in which walls are represented as line segments in the XY plane. Formally, we define a floorplan F as a tuple (R_w, Z, T) , where

- R_w is a 3-dimensional rotation;

- $Z = (z_f, z_c)$ is a pair of scalars defining the floor and ceiling planes,

$$P_{\text{floor}} = \{\mathbf{x} \mid \mathbf{x} \cdot [0 \ 0 \ 1] = z_f\} \quad (4.8)$$

$$P_{\text{ceil}} = \{\mathbf{x} \mid \mathbf{x} \cdot [0 \ 0 \ 1] = z_c\}; \quad (4.9)$$

- $T = \{(\mathbf{u}_i, \mathbf{v}_i)\}_{i=0}^M$, is a set of line segments defining walls, $\mathbf{u}_i, \mathbf{v}_i \in \mathbb{R}^2$.

A floorplan F is converted to a reconstruction as follows. First, extrude each line segment vertically out of the floor plane to meet the ceiling plane, resulting in a set of vertical planes. Next, add polygons for the floor and ceiling planes, which may simply be sufficiently large polygons parallel to the XY plane. Finally, apply the coordinate transform R_w . Formally, the reconstruction corresponding to a given floorplan is given by $G = \{(\mathbf{p}_i, \mathbf{q}_i, \mathbf{r}_i, \mathbf{s}_i)\}$ where

$$\mathbf{p}_i = R_w \begin{bmatrix} \mathbf{u}_i \\ z_f \end{bmatrix} \quad (4.10)$$

$$\mathbf{q}_i = R_w \begin{bmatrix} \mathbf{v}_i \\ z_f \end{bmatrix} \quad (4.11)$$

$$\mathbf{r}_i = R_w \begin{bmatrix} \mathbf{v}_i \\ z_c \end{bmatrix} \quad (4.12)$$

$$\mathbf{s}_i = R_w \begin{bmatrix} \mathbf{u}_i \\ z_c \end{bmatrix} \quad (4.13)$$

For the remainder of this chapter we use the term “corner” to refer to a point at which two line segments in the floorplan meet. In 3D, a corner is therefore a vertical seam at which two walls meet. This contrasts with the usual computer vision terminology in which a corner corresponds to a point in 3D.

4.3.1 Images of Floorplans

The remainder of this thesis will be simplified considerably if we can assume that vertical lines in the world appear vertical in image coordinates. To that end we define the following homography:

$$H_{\text{rect}} = \begin{pmatrix} \mathbf{v}_v \times \mathbf{e}_3 \\ \mathbf{v}_v \\ \mathbf{v}_v \times \mathbf{e}_3 \times \mathbf{v}_v \end{pmatrix}. \quad (4.14)$$

where \mathbf{v}_v is the vertical vanishing point and $\mathbf{e}_3 = [0 \ 0 \ 1]$. Let I' be the image formed by applying the homography H_{rect} as a coordinate transform to I ,

$$I'(\mathbf{x}) = I(H_{\text{rect}}\mathbf{x}) \quad (4.15)$$

where we are implicitly working in homogeneous coordinates. We say that I' is *vertically rectified* and we make the following claim. If two 3D points are separated by a displacement parallel to the z -axis then their projections in I' are separated by an offset parallel to the y -axis. To show that this is the case, let \mathbf{v} and \mathbf{u} be points in 3D with $\mathbf{u} = \mathbf{v} + \lambda \mathbf{e}_3$ and consider their projections in I' ,

$$\mathbf{p} = H_{\text{rect}}(R\mathbf{v} + \mathbf{t}) \quad (4.16)$$

$$\mathbf{q} = H_{\text{rect}}(R\mathbf{u} + \mathbf{t}). \quad (4.17)$$

Now

$$\mathbf{q} - \mathbf{p} = H_{\text{rect}}(R\mathbf{u} + \mathbf{t}) - H_{\text{rect}}(R\mathbf{v} + \mathbf{t}) \quad (4.18)$$

$$= H_{\text{rect}}(R(\mathbf{v} + \lambda \mathbf{e}_3) + \mathbf{t} - R\mathbf{v} - \mathbf{t}) \quad (4.19)$$

$$= \lambda H_{\text{rect}} R \mathbf{e}_3. \quad (4.20)$$

$$= \lambda H_{\text{rect}} \mathbf{v}_v. \quad (4.21)$$

Now the x component of (4.21) is

$$\lambda(\mathbf{v}_v \times \mathbf{e}_3)^T \mathbf{v}_v = 0 , \quad (4.22)$$

where we have used the fact that $(\mathbf{v}_v \times \mathbf{e}_3)$ must be orthogonal to \mathbf{v}_v . Hence the image displacement between \mathbf{q} and \mathbf{p} is parallel to the y -axis.

In practice the image I is not a continuous signal but is sampled at discrete pixel locations. As a result, the transformation H_{rect} will result in a non-uniformly sampled image plane. Worse, if the vertical vanishing point \mathbf{v}_v is within the original image boundary, then the transformation H_{rect} will introduce a singularity at this point.

Thankfully, the validity of the algorithms presented in the remainder of this thesis do not in fact depend on performing the rectification (4.15) — we could perform all calculations assuming a non-rectified image, in which case “image columns” would be replaced by “rays cast from the vertical vanishing point” and “paths from left to right through the image” would be replaced by “paths moving clockwise with respect to the vertical vanishing point”. However, it will simplify the remainder of this thesis considerably if we assume the rectification (4.15) so without loss of generality we assume throughout the remainder of this thesis that the vertical vanishing point falls outside the image bounds and that images have been rectified.

4.4 Parametrisation

In this section we turn to the scene parametrisation that we will use during learning and inference in the following chapters. We parametrise each component of the floorplan tuple $F = (R_w, Z, T)$ separately. R_w is parametrised as a member of the Lie group $SO(3)$, which will be discussed further in Chapter 5. The remainder of this section is concerned with the parametrisation of Z (the floor and ceiling position) and T (the position of the walls).

4.4.1 Parametrising the floor and ceiling location

In the preceding sections we implicitly parametrised the floor and ceiling planes with a pair of scalars $z_f, z_c \in \mathbb{R}$. We now switch to a different parametrisation (μ, λ) where $\mu \in \mathbb{R}$ parametrises a planar homology relating the images the floor and ceiling planes, and λ represents a scale factor that cannot be observed from a single view. The advantage of this representation is that μ can be estimated from a single image and is sufficient to perform inference in the single image context. If only one image is available then we may simply ignore λ and the remainder of the algorithms in this thesis go through without modification, while in the multiple view context we may estimate λ directly.

Another way to say this is that the algorithms to be presented in following chapters recover scene structure only up to an unknown scale factor, which can be estimated only in the context of multiple views. Intuitively, one can think of μ as a scale-invariant representation of the relationship between the floor and ceiling planes in the image domain, and λ as the parameter that resolves this scale ambiguity. In the single view setting we simply avoid resolving scale.

We define λ as the absolute z components of either the floor or ceiling plane. To define μ we first need to introduce the *Manhattan correspondence*. For the purpose of this section we will say that two image points p_f and p_c correspond if and only if the difference between their back-projections onto the floor and ceiling plane respectively is parallel to the z axis, as illustrated in Figure 4.2. Formally, for a camera (K, R, t) , p_f and p_c are in correspondence if and only if there exists $X_f, X_c \in \mathbb{R}^3$ with

$$p_f = K(RX_f + t) \quad (4.23)$$

$$p_c = K(RX_c + t) \quad (4.24)$$

$$X_c = X_f + r\mathbf{e}_3 \quad r \in \mathbb{R} \quad (4.25)$$

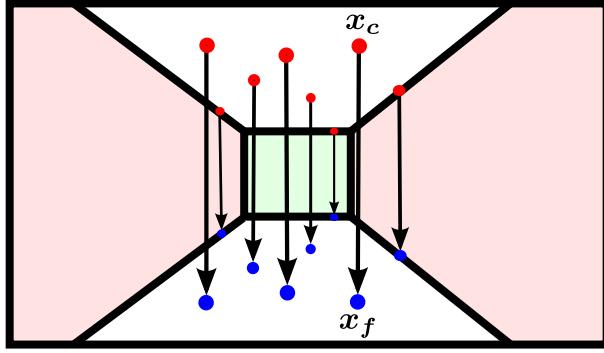


Figure 4.2: The mapping H_a transfers points between the ceiling and floor.

Since the floor and ceiling planes are parallel, there is a planar homology H_a such that

$$\mathbf{p}_f = H_a \mathbf{p}_c \quad (4.26)$$

if and only if \mathbf{p}_f and \mathbf{p}_c are in correspondence [11]. We define μ as the unique real number such that

$$H_a = I + \mu \frac{\mathbf{v}_v \mathbf{h}^T}{\mathbf{v}_v \cdot \mathbf{h}} \quad (4.27)$$

where

$$\mathbf{h} = K R \mathbf{e}_1 \times K R \mathbf{e}_2 \quad (4.28)$$

is the horizon line. Given any corresponding pair $(\mathbf{p}_f, \mathbf{p}_c)$, μ is given by [11]

$$\mu = \langle \mathbf{v}_v, \mathbf{p}_c, \mathbf{p}_f, (\mathbf{p}_c \times \mathbf{p}_f) \times \mathbf{h} \rangle, \quad (4.29)$$

where

$$\langle \mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d} \rangle = \frac{(\mathbf{a} - \mathbf{c}) \cdot (\mathbf{b} - \mathbf{d})}{(\mathbf{b} - \mathbf{c}) \cdot (\mathbf{a} - \mathbf{d})} \quad (4.30)$$

is the characteristic cross ratio of H_a .

Note that μ is invariant to the camera intrinsics. To see this, consider transforming the image by some homography M . In the transformed image the horizon is given by $\mathbf{h}' = |M|M^{-T}\mathbf{h}$, the vertical vanishing point is $\mathbf{v}_v' = M\mathbf{v}_v$, and the Manhattan homology is

$$H'_a = I + \mu' \frac{\mathbf{v}_v' \mathbf{h}'^T}{\mathbf{v}_v' \cdot \mathbf{h}'} . \quad (4.31)$$

We also have that

$$H'_a = MH_aM^{-1} \quad (4.32)$$

$$= M(I + \mu \frac{\mathbf{v}_v \mathbf{h}^T}{\mathbf{v}_v \cdot \mathbf{h}})M^{-1}. \quad (4.33)$$

Substituting $\mathbf{h} = \frac{1}{|M|}M^T\mathbf{h}'$ and $\mathbf{v}_v = M^{-1}\mathbf{v}'_v$ gives

$$H'_a = I + \mu \frac{M(M^{-1}\mathbf{v}'_v)(\frac{1}{|M|}M^T\mathbf{h}')^T M^{-1}}{(M^{-1}\mathbf{v}'_v)^T(\frac{1}{|M|}M^T\mathbf{h}')} \quad (4.34)$$

$$= I + \mu \frac{\mathbf{v}'_v \mathbf{h}'^T}{\mathbf{v}'_v \cdot \mathbf{h}'}. \quad (4.35)$$

Comparing (4.35) and (4.31) shows that $\mu' = \mu$. We therefore assume without loss of generality that K is the identity for the remainder of this section.

To complete our parametrisation of the floor and ceiling planes we need to show how to compute (μ, λ) from (z_f, z_c) . We may simply set λ to either z_f or z_c . Let

$$\mathbf{e}_f = z_f \mathbf{e}_3 \quad (4.36)$$

$$\mathbf{e}_c = z_c \mathbf{e}_3 \quad (4.37)$$

$$\mathbf{p}_f = R\mathbf{e}_f + \mathbf{t} \quad (4.38)$$

$$\mathbf{p}_c = R\mathbf{e}_c + \mathbf{t} \quad (4.39)$$

Then

$$(\mathbf{p}_c \times \mathbf{p}_f) \times \mathbf{h} = ((R\mathbf{e}_c + \mathbf{t}) \times (R\mathbf{e}_f + \mathbf{t})) \times (R\mathbf{e}_1 \times R\mathbf{e}_2) \quad (4.40)$$

$$= ((R\mathbf{e}_c + \mathbf{t}) \times (R\mathbf{e}_f + \mathbf{t})) \times R(\mathbf{e}_1 \times \mathbf{e}_2) \quad (4.41)$$

$$= (R(\mathbf{e}_c - \mathbf{e}_f) \times \mathbf{t}) \times R\mathbf{e}_3. \quad (4.42)$$

We can now write

$$\mu = \langle \mathbf{v}_v, \mathbf{p}_c, \mathbf{p}_f, (\mathbf{p}_c \times \mathbf{p}_f) \times \mathbf{h} \rangle \quad (4.43)$$

$$= \frac{(\mathbf{v}_v - \mathbf{p}_f)^T (\mathbf{p}_c - (\mathbf{p}_c \times \mathbf{p}_f) \times \mathbf{h})}{(\mathbf{p}_c - \mathbf{p}_f)^T (\mathbf{v}_v - (\mathbf{p}_c \times \mathbf{p}_f) \times \mathbf{h})} \quad (4.44)$$

$$= \frac{(R\mathbf{e}_3 - (R\mathbf{e}_f + \mathbf{t}))^T (R\mathbf{e}_c + \mathbf{t} - (R(\mathbf{e}_c - \mathbf{e}_f) \times \mathbf{t}) \times R\mathbf{e}_3)}{(R\mathbf{e}_c + \mathbf{t} - R\mathbf{e}_f - \mathbf{t})^T (R\mathbf{e}_3 - (R(\mathbf{e}_c - \mathbf{e}_f) \times \mathbf{t}) \times R\mathbf{e}_3)} \quad (4.45)$$

$$= \frac{(\mathbf{e}_3 - \mathbf{e}_f - R^T \mathbf{t})^T R^T (R\mathbf{e}_c + \mathbf{t} - (R(\mathbf{e}_c - \mathbf{e}_f) \times \mathbf{t}) \times R\mathbf{e}_3)}{(\mathbf{e}_c - \mathbf{e}_f)^T R^T (R\mathbf{e}_3 - (R(\mathbf{e}_c - \mathbf{e}_f) \times \mathbf{t}) \times R\mathbf{e}_3)} \quad (4.46)$$

$$= \frac{(\mathbf{e}_3 - \mathbf{e}_f - R^T \mathbf{t})^T (\mathbf{e}_c + R^T \mathbf{t} - ((\mathbf{e}_c - \mathbf{e}_f) \times R^T \mathbf{t}) \times \mathbf{e}_3)}{(\mathbf{e}_c - \mathbf{e}_f)^T \mathbf{e}_3 - (\mathbf{e}_c - \mathbf{e}_f)^T (((\mathbf{e}_c - \mathbf{e}_f) R^T \mathbf{t}) \mathbf{e}_3)} \quad (4.47)$$

$$= \frac{(\mathbf{e}_3 - \mathbf{e}_f - R^T \mathbf{t})^T (\mathbf{e}_c + R^T \mathbf{t}) + (z_c - z_f)(\|\mathbf{t}\|^2 - (\mathbf{e}_3^T R^T \mathbf{t})^2)}{z_c - z_f} \quad (4.48)$$

$$= \frac{1}{z_c - z_f} \left((z_c + z_\rho)(1 - z_f - z_\rho) + z_\rho^2 - \|\mathbf{t}\|^2 \right) + \|\mathbf{t}\|^2 - z_\rho^2 \quad (4.49)$$

where in the last line we substituted $z_\rho = \mathbf{e}_3^T R^T \mathbf{t}$.

4.4.2 Parametrising walls

We choose to parametrise T in image coordinates because this will simplify the algorithms to be presented later. We first present the image–domain parametrisation, then show that it uniquely specifies a 3D scene.

Consider the scenes shown in Figure 4.3. Following rectification, vertical edges in the world appear vertical in each image, including the vertical seams between adjacent wall segments, so any line drawn vertically from the top to bottom of a rectified image intersects exactly one wall segment. That this is true in general follows from our assumption that walls meet at vertical seams and extend continuously from floor to ceiling, that the camera is located between the floor and ceiling planes, and that the environment is closed.

It turns out that once the vanishing points \mathbf{v}_i and the Manhattan homology H_a are known, the quadrangle outlining a wall segment in rectified image coordinates is fully specified by just

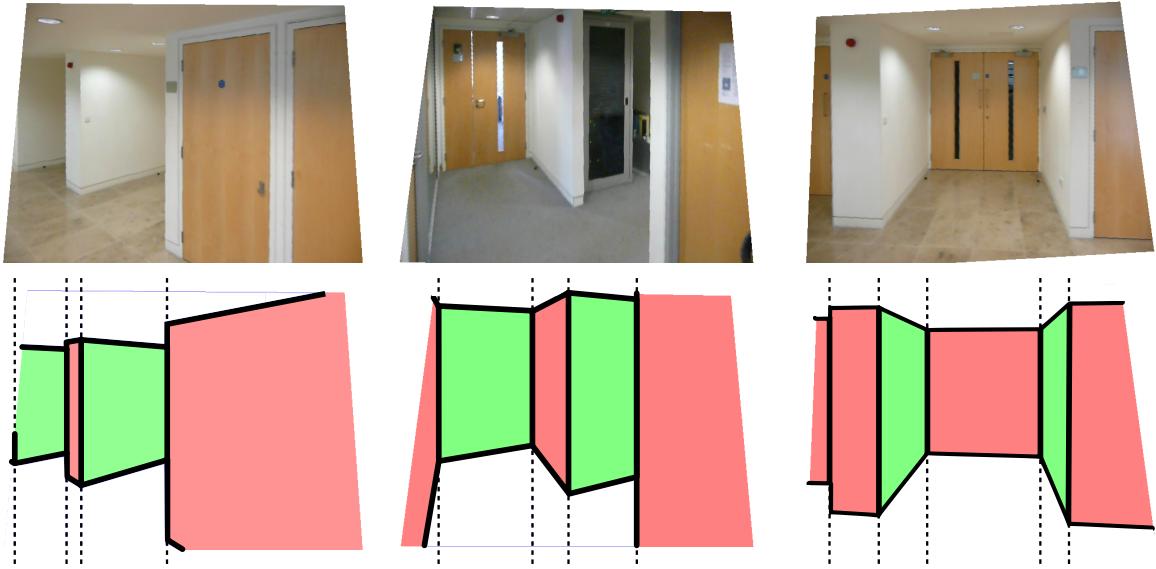


Figure 4.3: Three input images and the indoor Manhattan models we seek. Notice how each image column intersects exactly one wall.

four scalars,

$$W = (x_0, y, o, x_1) \quad (4.50)$$

where x_0 is the location of the left edge, x_1 is the location of the right edge, y is the y -coordinate of the top-left corner, and $o \in \{1, 2\}$ is the index of the vanishing point associated with the wall. The four vertices of the wall are then given by

$$\mathbf{p}_i = \begin{bmatrix} x & y & 1 \end{bmatrix}^t \quad (4.51)$$

$$\mathbf{q}_i = \mathbf{p}_i \times \mathbf{v}_a \times \begin{bmatrix} 1 & 0 & -c_{i+1} \end{bmatrix}^T \quad (4.52)$$

$$\mathbf{r}_i = H_a \mathbf{q}_i \quad (4.53)$$

$$\mathbf{s}_i = H_a \mathbf{p}_i . \quad (4.54)$$

Figure 4.4 illustrates the geometry of the parametrisation above. Since each image column intersects exactly one wall, we may think of the scene as a sequence of walls from left to right in the image. This observation suggests our scene parametrisation, which is simply a sequence of wall segments. We make the additional observation that the right edge of a wall (x_1 in equation (4.50)) is redundant because it is always coincident with the left edge of the wall to its

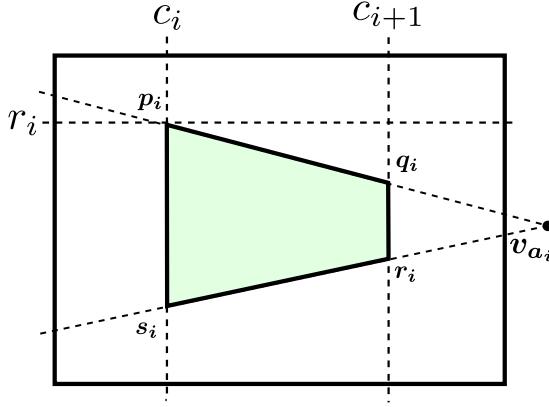


Figure 4.4: The row/column indices c_i, c_{i+1}, r_i , together with the vanishing point index $a_i \in \{l, r\}$ and the homology H_a fully determine the four vertices of a wall.

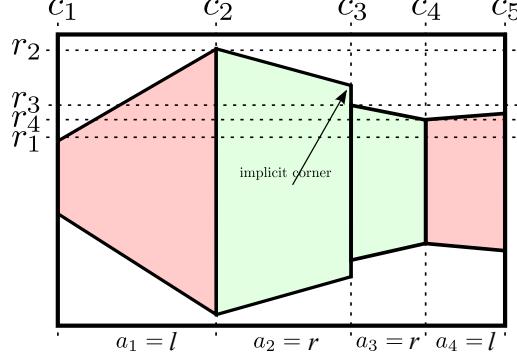


Figure 4.5: An illustration of the model $M = (c_1, r_1, a_1, \dots, c_5, r_4, a_4, c_5)$.

right. A scene therefore is fully specified by

$$M = (x_1, y_1, o_1, x_2, y_2, o_2, \dots, x_{n-1}, y_{n-1}, o_{n-1}, x_n). \quad (4.55)$$

An example scene and its parametrisation is shown in Figure 4.5. Note that although the right edge of a wall always coincides with the left edge of its neighbour, it is not always the case that the top and bottom edges of adjacent walls meet at a point, *i.e.* $r_i \neq s_{i+1}$ in general. For example, notice the second edge in Figure 4.5.

Physical Feasibility

Not all scenes M are physically realisable as metric reconstructions, but those that are not can be discarded using simple tests on the locations of walls and vanishing points as enumerated by

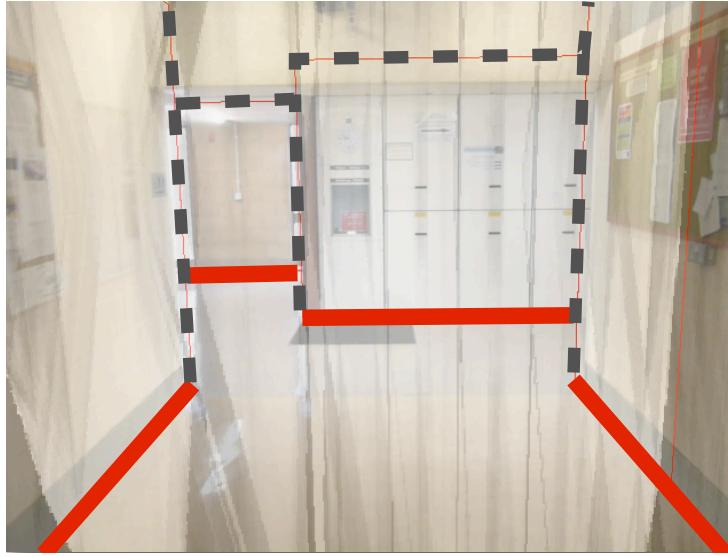


Figure 4.6: An example of an indoor Manhattan scene and its seam representation shown in red. The grey dotted lines can be recovered from the seam using the Manhattan homology H_a .

Lee *et al.* [13]. The reader is referred to their paper for details; the key result for our purposes is that a model is feasible if all of its corners are feasible, and the feasibility of a corner is dependent only on the immediately adjoining walls. We discuss these issues in detail during Chapter 6.

4.4.3 The Seam Representation

For the purpose of the probabilistic model that we will present in chapter Chapter 6 we will now present an additional, slightly different parametrisation that we will refer to as the *seam* representation of indoor Manhattan scenes. We will refer to the parametrisation of equation (4.55) as the *vertex* representation. A scene in vertex representation M uniquely determines a scene in seam S representation, but the converse is not true. For this reason we will always work in the vertex representation during inference, but to evaluate likelihoods, priors, and posteriors for a fixed scene we will find it notationally more convenient to work in the seam representation. Under the seam representation a scene is represented by associating a pair of scalars (s_j, o_j) to each image column j ,

$$S = \{(s_j, o_j)\}_{j=0}^W, \quad (4.56)$$

where s_j is the y -coordinate of the bottom edge of the wall that intersects the j^{th} column, and $o_j \in \{1, 2\}$ is the index of its associated vanishing point. Note that the size of this representation is proportional to the image width, whereas the size of the vertex representation is proportional to the number of distinct wall segments. See Figure 4.6 for an illustration of this parametrisation.

We compute the seam representation S from a scene in vertex representation M as follows. For each $j \in [0, W]$ we identify the index of the (unique) wall segment that intersects column j by finding $x_i, x_{i+1} \in M$ such that $x_i \leq j \leq x_{i+1}$. Next we compute s_i and r_i according to equations (4.54) and (4.53), after which the position of the wall seam at column j is given by

$$s_j = (\mathbf{r}_i \times \mathbf{s}_i) \times \begin{bmatrix} 0 & 1 & -j \end{bmatrix}^T. \quad (4.57)$$

Finally, the orientation o_j at column j is equal to the orientation of the identified wall segment, o_i . The constructive process described above shows that there is exactly one seam S associated with a scene in vertex representation. However, there is no unique mapping in the reverse direction. In other words, the mapping from vertex representation to seam representation is many-to-one.

4.5 Deductions

In this section we make several geometric observations that are crucial to the following chapters.

4.5.1 Classification of Corners

Lee *et al.* [13] showed that corners between adjacent walls can be divided into three categories as follows.

- A *convex corner* occurs where two wall segments meet exactly and form an internal angle

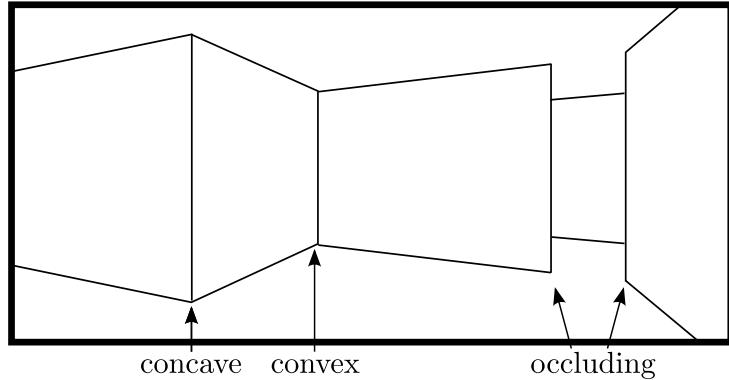


Figure 4.7: Each corner in an indoor Manhattan environment can be categorised as concave, convex, or occluding.

not greater than 180°, i.e.

$$r_i = s_{i+1} \quad (4.58)$$

$$\text{and} \quad (4.59)$$

$$\frac{(s_i - r_i) \cdot (q_i - r_i)}{\|s_i - r_i\| \| (q_i - r_i)\|} \leq \frac{(r_{i+1} - r_i) \cdot (q_i - r_i)}{\|r_{i+1} - r_i\| \| (q_i - r_i)\|}. \quad (4.60)$$

- A *concave corner* occurs where two wall segments meet exactly and form an internal angle greater than 180°, i.e.

$$r_i = s_{i+1} \quad (4.61)$$

$$\text{and} \quad (4.62)$$

$$\frac{(s_i - r_i) \cdot (q_i - r_i)}{\|s_i - r_i\| \| (q_i - r_i)\|} > \frac{(r_{i+1} - r_i) \cdot (q_i - r_i)}{\|r_{i+1} - r_i\| \| (q_i - r_i)\|}. \quad (4.63)$$

- An *occluding corner* occurs where two wall segments meet but their lower edges do not coincide, i.e.

$$r_i \neq s_{i+1} \quad (4.64)$$

Examples of each type of corner are shown in Figure 4.7.

4.5.2 Recovering Metric Scene Structure

We now show that once R_w and Z are known, a scene M in vertex representation uniquely specifies a 3D scene G . We show this by demonstrating how to convert a scene M to a floorplan F , after which we may apply equations (4.51), (4.52), (4.53), and (4.54) to recover a polygonal reconstruction.

Let $\text{backproject}(\mathbf{p}; \mathbf{w}, K, R, t)$ be the back-projection of image point \mathbf{p} onto the plane $\mathbf{x} \cdot \mathbf{w} = 0$ given camera intrinsics K and extrinsics R, t . To compute this we solve for \mathbf{y} in

$$P\mathbf{y} = \mathbf{0} \quad (4.65)$$

$$\|\mathbf{y}\| = 1 \quad (4.66)$$

where

$$P = \begin{bmatrix} KR & Kt - \mathbf{p} \\ 0 & 1 \\ 0 & -z_f \end{bmatrix}. \quad (4.67)$$

We will usually omit the camera parameters and simply write $\text{backproject}(\mathbf{p}; \mathbf{w})$. Furthermore, since we will often be back-projecting onto planes orthogonal to the z axis we adopt the shorthand

$$\text{backproject}(\mathbf{p}; z) = \text{backproject}(\mathbf{p}; \begin{bmatrix} 0 & 0 & 1 & -z \end{bmatrix}). \quad (4.68)$$

To recover a floorplan F from a scene M we simply need to back-project q_i, s_i pairs onto the floor plane. This is formalised in Algorithm 4.1.

Algorithm 4.1 Recovering a metric 3D scene from the vertex representation M .

Require: $M = (x_1, y_1, o_1, \dots, x_n)$ {a scene in vertex representation}

Ensure: $F = \{(\mathbf{u}_i, \mathbf{v}_i)\}$

$F \leftarrow \emptyset$

for $i = 1 \dots n - 1$ **do**

$\mathbf{u}_i = \text{backproject}(\mathbf{s}_i; z_f)$

$\mathbf{u}_i = \text{backproject}(\mathbf{r}_i; z_f)$

$F \leftarrow F \cup (\mathbf{u}_i, \mathbf{v}_i)$

end for

Algorithm 4.2 Recovering orientation and depth for an image location p under a scene hypothesis M .

```

Require:  $M = (x_1, y_1, o_1, \dots, x_n)$  {a scene in vertex representation}
Require:  $p = (x, y) \in \mathbb{R}^2$ 
Require:  $x_1 \leq x < x_n$ 
Ensure:  $i_p \in [1, n], a_p \in \{1, 2, 3\}, d_p > 0$ 
for  $i = 1 \dots n - 1$  do
    if  $x_i \leq x < x_{i+1}$  then
        {Compute the floor-wall and ceiling-wall seam in column  $x$ }
         $s_f \leftarrow e_2 \cdot (s_i \times r_i) \times [1 \ 0 \ -x]^T$ 
         $s_c \leftarrow e_2 \cdot (s_i \times r_i) \times [1 \ 0 \ -x]^T$ 
        if  $y < s_c$  then
             $a_p \leftarrow \text{"vertorient"}$ 
             $X_p \leftarrow \text{backproject}(p; z_c)$ 
        else if  $s_c < y < s_f$  then
             $a_p \leftarrow o_x$ 
             $w_p \leftarrow [1 \ 0 \ 0 \ -\text{backproject}(x_f; z_f) \cdot e_1]^T$ 
             $X_p \leftarrow \text{backproject}(p; w_p)$ 
        else
             $a_p \leftarrow \text{"vertorient"}$ 
             $X_p \leftarrow \text{backproject}(p; z_f)$ 
        end if
         $d_p \leftarrow (RX_p + t) \cdot e_3$ 
        return
    end if
end for

```

4.5.3 Recovering Orientation and Depth

We say that an indoor Manhattan scene M predicts the depth d_p of each pixel p because we can compute the depth of each p given the hypothesis M . Similarly, a scene predicts an orientation $a_p \in \{1, 2, 3\}$ for each image point, which is the orientation of the 3D surface that projects to p .

To compute either the orientation a_p or depth d_p for an image point p we first need to identify a wall that contains p within its (image-domain) outline. If there is such a wall then p has the orientation of that wall and we back-project p onto the wall in 3D to recover depth. If p is not contained by any wall then p must be on the floor or ceiling and therefore has the horizontal orientation. In this case we recover depth by back-projecting onto the floor or ceiling plane, according to whether p is above or below the horizon. This process is formalised in Algorithm 4.2.

Algorithm 4.3 Recovering orientation and depth for an image location \mathbf{p} under the seam representation S .

Require: $S = \{(s_j, o_j)\}$ {a scene in seam representation}

Require: $\mathbf{p} = (x, y) \in \mathbb{Z}^2$ {a query pixel}

Ensure: $a_p \in \{1, 2, 3\}, d_p > 0$

{Compute the floor–wall and ceiling–wall seam in column x }

$$s_f \leftarrow s_x$$

$$\mathbf{x}_f \leftarrow [x \ s_f \ 1]^T$$

$$\mathbf{x}_c \leftarrow H_a \mathbf{x}_f$$

$$s_c \leftarrow \frac{\mathbf{x}_c \cdot \mathbf{e}_2}{\mathbf{x}_c \cdot \mathbf{e}_3}$$

if $y < s_c$ **then**

$$a_p \leftarrow \text{"vertorient"}$$

$$X_p \leftarrow \text{backproject}(\mathbf{p}; z_c)$$

else if $s_c < y < s_f$ **then**

$$a_p \leftarrow a_x$$

$$\mathbf{w}_p \leftarrow [1 \ 0 \ 0 \ -\text{backproject}(\mathbf{x}_f; z_f) \cdot \mathbf{e}_1]^T$$

$$X_p \leftarrow \text{backproject}(\mathbf{p}; \mathbf{w}_p)$$

else

$$a_p \leftarrow \text{"vertorient"}$$

$$X_p \leftarrow \text{backproject}(\mathbf{p}; z_f)$$

end if

$$d_p \leftarrow (R X_p + \mathbf{t}) \cdot \mathbf{e}_3$$

These calculations are considerably simpler in the seam representation, which is why it proves useful in following chapters. Suppose we are given a scene in seam representation S and an integer pixel location $\mathbf{p} \in \mathbb{Z}^2$. We first look up the pair $(s_j, o_j) \in S$ for the column j in which the pixel $\mathbf{p} = (x, y)$ falls, which we denote (s_x, o_x) . Then the floor–wall seam in column x is at $s_f = s_x$ and the ceiling–wall seam can be recovered using the Manhattan homology. From here the orientation and depth at \mathbf{p} are obtained using the procedure shown in Algorithm 4.3.

4.5.4 Column-wise Decomposability

We have so far shown that the vertex representation suffices to recover metric scene structure up to scale, and that the depth and orientation predicted for each pixel can be computed directly from either the vertex or seam representation. One property of particular significance for the algorithms to be presented in the remainder of this thesis is that the procedure of Algorithm 4.3 only requires access to the pair (s_j, o_j) for the image column j that contains the query pixel. That is, the depth and orientation predicted for a pixel \mathbf{p} are functionally independent of the

location of the floor–wall seam in columns other than the one containing p .

4.6 Conclusion

We have presented the indoor Manhattan model, in which the world is represented by floor, ceiling, and wall planes. We have motivated this model in terms of its salience, compactness, and the simple parametrisations it permits. The vertex and seam representations are the primary contributions of this chapter; their specific structure is crucial to all following chapters. In particular, we will make repeated use of the decomposability property described in Section 4.5.4. While this chapter has described the geometry of ideal indoor Manhattan environments, the system that we develop in this thesis is not restricted to working in such sterile contexts; rather, the model presented in this chapter describes the scene structure that we seek to recover *despite* clutter and sensor noise.

5

Identifying Manhattan Orientations

In order to make use of the Manhattan world assumption we must first discover the orientation of the Manhattan world with respect to the camera. This chapter focusses on estimating a 3D rotation that transforms the three cardinal Manhattan orientations onto the x , y , and z axes. In contrast to much previous work, we seek to recover this rotation given a sequence of calibrated images, such as may be provided by a structure–from–motion or SLAM system. Where previous multiple–view approaches have focussed on surface normal estimates derived from a reconstructed point cloud, we build upon ideas from single–view vanishing point detection and cast estimation in terms of photometric information. We develop a probabilistic model relating observed line segments to the 3D rotation we seek, and solve maximum–likelihood inference using an Expectation–Maximisation algorithm. Our likelihoods are derived from principled error models, and estimation is performed as a single optimisation. We show that our approach substantially out–performs a state–of–the–art approach based on surface normals, while providing much greater robustness than single–view vanishing point estimation.¹

¹This work was published in part in:
Flint, Mei, Murray, and Reid, “Growing Semantically Meaningful Models For Visual SLAM”, in Proceedings of the 2010 Conference on Computer Vision and Pattern Recognition[21]

5.1 Introduction

Many man-made environments contain numerous surfaces oriented in three mutually orthogonal directions. In order to leverage the special structure of these Manhattan scenes we must identify these three canonical directions. Since these directions are assumed to be orthogonal to one another, identifying them is equivalent to finding a 3-dimensional rotation R_w that maps them onto the x , y , and z axes.

We assume that an estimate of the camera pose for each frame is available, such as might be provided by a structure-from-motion system. We will not use any point cloud generated from structure-from-motion in this chapter. We assume that the provided camera poses are measured in a fixed but arbitrary coordinate frame. The problem of determining R_w in the context of a single image is equivalent to detecting vanishing points since three vanishing points and a calibrated camera are sufficient to recover R_w . However, in the multiple view context it makes sense to estimate R_w jointly using all available information, rather than to merge single-view estimates post-hoc. One view of the contribution of this chapter is therefore as a generalisation of vanishing point detection to multiple calibrated views.

We estimate R_w using the assumption that a significant number of the observed line segments correspond to one of three Manhattan orientations. We present a graphical model relating line segments to R_w , then we present an Expectation–Maximisation algorithm for inference within this model. Our likelihoods are derived from a well-defined model of image formation, and we optimise jointly with respect to all line segments in all views in a single optimisation. Our approach parallels ideas presented in the context of single-view vanishing point detection, particularly those of Andrew Zisserman [27] and Frank Dellaert [57]. Our contribution is a principled way to incorporate multiple calibrated views into a single optimisation, and an empirical demonstration that doing so significantly out-performs both single-view vanishing point estimation and surface-normal-based estimators.

5.2 Background

Vanishing point detection has a long history in the computer vision literature, beginning with the seminal work of Barnard *et al.* [4]. In this section we survey key contributions within this literature and discuss their relationship to our work.

5.2.1 Single View Approaches

Several approaches to single–image vanishing point detection have been proposed in the literature [4, 41, 58]. Common among all such approaches is the use of line segments as the fundamental observation from which inference is performed. The vanishing points under consideration are assumed to generate several associated line segments, all of which meet at the vanishing point (modulo measurement errors). The estimation of vanishing points is approached in various ways. The classic approach due to Barnard [4] was to project images onto the Gaussian sphere and use the Hough transform to identify vanishing points. Several authors have proposed similar voting schemes under different parametrisations of the accumulator space that improve statistical robustness.

Shufelt [58] introduced explicit error models for observed line segments, but retained the voting-based estimation strategy. Hartley and Zisserman [27] first proposed a generative model relating vanishing point to line segments and cast maximum likelihood estimation as a non-linear estimation problem. We make similar modelling assumptions in the present work. Kanatani [38] minimised the algebraic deviation of vanishing points from observed line segments, which leads to an efficient linear least-squares solution at the cost of a less precise error model. Schindler and Dellaert [57] laid out an integrated Bayesian approach to the estimation of multiple vanishing points.

The approaches discussed thus far estimate vanishing points independently, using K-means clustering or the expectation maximisation algorithm to resolve the assignment of line segments to vanishing points. Coughlan and Yuille [10] observed that under the Manhattan world

assumption the three cardinal vanishing points are mutually orthogonal and hence their locations in the image plane are highly correlated. Rather than estimating three independent vanishing points, the authors cast the estimation problems in terms of Euler angles corresponding to R_w , resulting in a three–DoF estimation problem in place of the 6 for independent vanishing point estimation. Koseckà and Zhang [41] showed that the Manhattan world assumption can be exploited even in the case of an uncalibrated camera. Their approach used a prior on camera intrinsics to condition the estimation process, though the estimation of three vanishing points was not explicitly coupled after this stage.

Denis *et al.* [15] took this a step further by exploiting a prior on the rotation R_w learned from training data. Whereas Coughlan and Yuille used a discretisation of the space of rotations to identify R_w , they showed how to exploit the Manhattan world assumption under the more principled Expectation–Maximisation approach of Schindler and Dellaert [57]. The M–step in their approach is a non–linear optimisation over rotations, leveraging the maximum likelihood reprojection error of Hartley and Zisserman[27].

Recently, Mirzaei and Roumeliotis [48] showed that minimising the algebraic error for three mutually orthogonal vanishing points can be solved globally using a polynomial solver. Their approach assumes the algebraic error; it is not clear whether the reprojection error leads to a likelihood that is amenable to a polynomial solution.

5.2.2 Multiple View Approaches

The literature concerning vanishing point estimation from multiple views is sparse, and generally involves non–photometric approaches to recover R_w . Furukawa and Curless [25] use multiple–view stereo to recover a dense point cloud, then estimate R_w by clustering surface normals on the unit hemisphere. Werner and Zisserman [67] identify vanishing points independently in several uncalibrated views, then resolve associations between views using a combinatorial search.

5.3 Overview of Proposed Approach

Reliable vanishing point detection from single images is fundamentally challenging in environments in which axis-oriented edges are rare, or in which one or two cardinal orientations have very few associated edges. Both scenarios are common in video sequences of indoor environments since the camera often views only a small portion of the scene. On the other hand, obtaining a dense reconstruction is a complex and computationally expensive pre-requisite for a three-parameter estimation problem. Furthermore, the point cloud provided by on-line structure-from-motion is too sparse to obtain accurate surface normal estimates.

We overcome these difficulties by reasoning in terms of line segments and integrating observations from many frames into the estimation. Since R_w is fixed for all frames it makes sense to leverage all available data during estimation. Whereas previous approaches often estimate vanishing points as a precursor to reconstructing camera poses [41, 67], we prefer to rely on the structure-from-motion system to provide camera poses, then recover vanishing points afterwards. Under this problem setup we can relate line segments in all views directly to R_w and therefore incorporate all available evidence into a joint estimation. We show that this is both computationally efficient and avoids the fragile single-view estimation scenario.

5.4 Generative Model

We assume the graphical model shown in Figure 5.1. The generative process operates as follows. First we sample a scene orientation R_w and the rotation component of the camera pose, R_j for each frame. (Camera translation plays no part when reasoning about vanishing points.) Next we sample a discrete variable z_i from a categorical distribution with mixing parameters π . The variable z_i is a binary vector of length 4 indicating which vanishing point each line segment is associated with. Exactly one element is set to 1, the other three are 0. The first three elements correspond to the three vanishing points, the fourth to a spurious observation not associated with any of the three Manhattan directions. We denote the (binary) value of the k^{th}

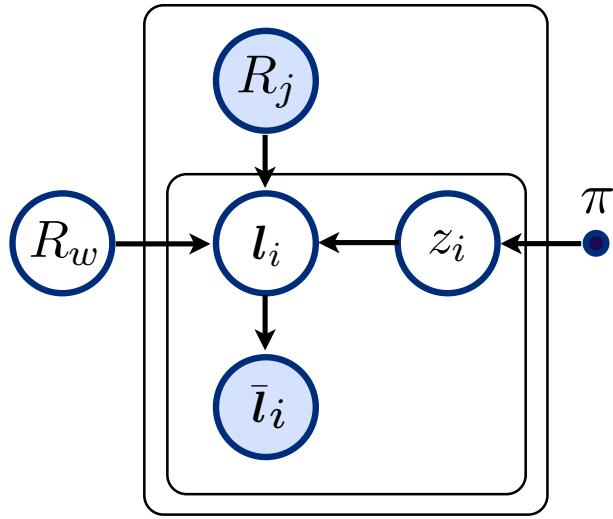


Figure 5.1: A graphical model relating line segments to vanishing points.

element z_{ik} , but we will also write $z_i = k$ as shorthand for the event that z_i is the binary vector with the k^{th} element set to 1. For clarity we write $z_i = \text{sp}$ in place of $z_i = 4$. If $z_i = \text{sp}$ then the line segment \bar{l}_i is sampled uniformly from the image. Otherwise, \bar{l}_i is associated with the vanishing point v_{z_i} and is sampled as follows.

The vanishing point corresponding to z_i is

$$\mathbf{v}_{z_i} = R_j R_w \mathbf{e}_{z_i} . \quad (5.1)$$

We sample a ray through v , then sample a line segment $\bar{l} = (\bar{a}, \bar{b})$ along the ray. We add isotropic Gaussian noise to arrive at the measured line segment $l = (a, b)$:

$$\mathbf{a} \sim \mathcal{N}(\bar{\mathbf{a}}, \Sigma) \quad (5.2)$$

$$\mathbf{b} \sim \mathcal{N}(\bar{\mathbf{b}}, \Sigma) . \quad (5.3)$$

This process is illustrated by the graphical model in Figure 5.1.

The likelihood for line segments is

$$P(l_i | z_i, R_w) = \mathcal{N}(d(l_i, \mathbf{v}_{z_i}); 0, \sigma) \quad (5.4)$$

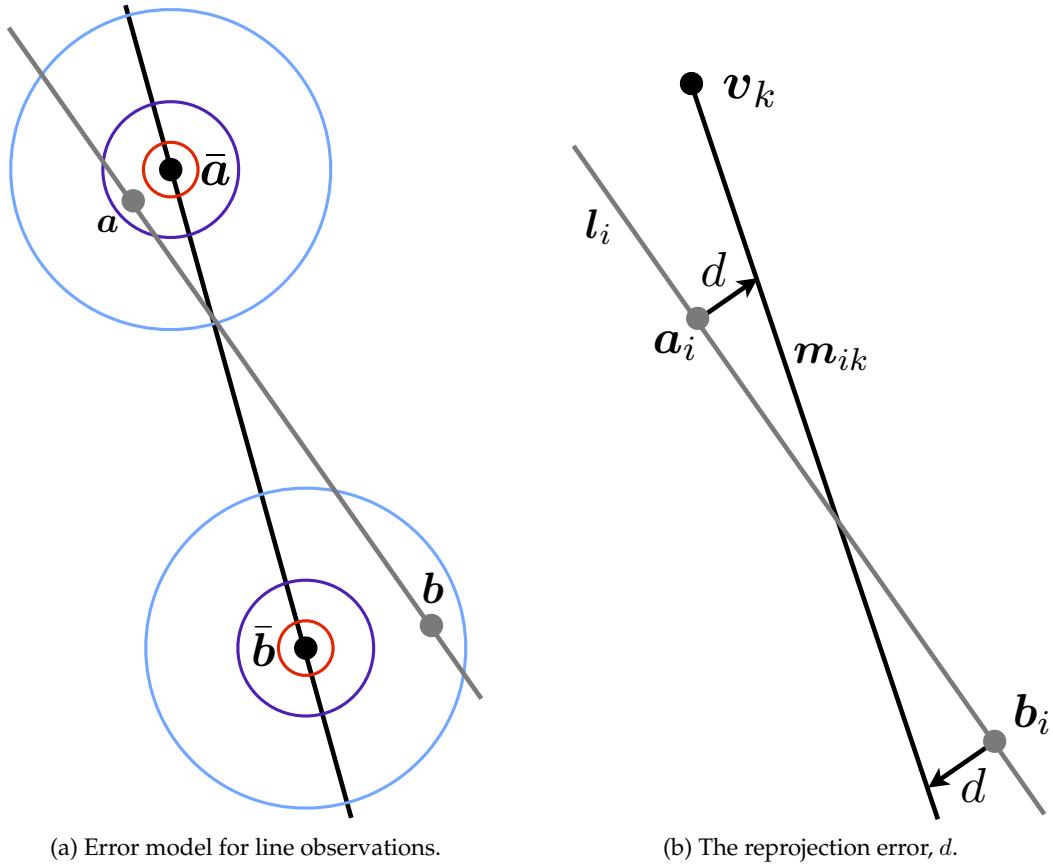


Figure 5.2: Illustration of our error model and various geometric quantities involved in the derivations in the main text.

where $d(\mathbf{l}, \mathbf{v})$ is the reprojection error illustrated in Figure 5.2. d is computed as follows. First, draw a line between the vanishing point \mathbf{v} and the mid-point of the line segment \mathbf{l} ,

$$\mathbf{m}_{ik} = \mathbf{v}_k \times \frac{\mathbf{a}_i + \mathbf{b}_i}{2} \quad (5.5)$$

Now, d is the distance between the line \mathbf{m} and either of the two end points of \mathbf{l} (the two quantities are equal, as evident by inspecting Figure 5.2). So

$$d(\mathbf{l}_i, \mathbf{v}_k) = \frac{\mathbf{a}_i \cdot \mathbf{m}_{ik}}{(\mathbf{a}_i \cdot \mathbf{e}_3)\eta_{ik}} \quad (5.6)$$

$$\eta_{ik} = \sqrt{(\mathbf{m}_{ik} \cdot \mathbf{e}_1)^2 + (\mathbf{m}_{ik} \cdot \mathbf{e}_2)^2}. \quad (5.7)$$

5.5 Estimating the Manhattan Coordinate Frame

We now turn to the inference algorithm that we use to find the maximum-likelihood estimate of R_w under the probabilistic assumptions described in the previous section. We begin by running the Canny edge detector [8], followed by an edge linking algorithm [41] to identify a set of straight line segments $\mathcal{L}_j = \{\mathbf{p} : \mathbf{l}_j \cdot \mathbf{p} = 0\}$.

Under the model presented in the previous section, the likelihood of R_w is

$$P(L | R_w, \{R_i\}) = \int P(L, Z | R_w, \{R_i\}) dZ. \quad (5.8)$$

For brevity we will omit the camera poses $\{R_i\}$ in the remainder of this section; it may be assumed that all probabilities to follow are conditioned on $\{R_i\}$.

The indicators Z are latent variables over which we need to marginalise in order to perform inference on R_w . Integrating explicitly is intractable so we turn to the expectation–maximisation (EM) algorithm, which alternately refines estimates of the posterior on Z (the “E-step”) and R_w (the “M-step”).

Due to the iterative nature of the EM algorithm we adopt superscript notation for the scene rotation and indicators to make clear the dependence between successive time steps. R_w^t is the estimate of the scene rotation at time t and q^t is the estimate for the posterior on Z at time t . Note that the former is an estimate of a variable but the latter is an estimate of a distribution. For notational convenience we will treat q as a continuous function, though of course in practice we really just store sufficient statistics for q . Also note that we never actually store or estimate the values of the indicators Z , so we have no need for superscripts on this variable. Although Z does appear in the derivations to follow, in every equation we are either marginalising it out, or defining a function over it.

5.5.1 Complete Data Likelihood

In sections to follow we will need to deal with the complete data likelihood,

$$P(L, Z | R_w) = \prod_i P(\mathbf{l}_i, z_i | R_w) \quad (5.9)$$

The likelihood terms $P(\mathbf{l}_i, z_i | R_w)$ depend on z_i in a complicated way, since \mathbf{l}_i is drawn from a different distribution depending on z_i :

$$P(\mathbf{l}_i, z_i | R_w) = P(z_i | R_w) P(\mathbf{l}_i, | z_i, R_w) \quad (5.10)$$

$$= \begin{cases} \pi_1 \mathcal{N}(d(\mathbf{l}_i, R_i R_w \mathbf{e}_1); 0, \sigma), & \text{if } z_i = 1 \\ \pi_2 \mathcal{N}(d(\mathbf{l}_i, R_i R_w \mathbf{e}_2); 0, \sigma), & \text{if } z_i = 2 \\ \pi_3 \mathcal{N}(d(\mathbf{l}_i, R_i R_w \mathbf{e}_3); 0, \sigma), & \text{if } z_i = 3 \\ \pi_{\text{sp}} \frac{1}{Z_{\text{sp}}}, & \text{if } z_i = \text{sp} . \end{cases} \quad (5.11)$$

However, due to our definition of z_i as a binary vector we can express this likelihood in the following analytic form,

$$P(\mathbf{l}_i, z_i | R_w) = \prod_{k=1}^4 \left(P(\mathbf{l}_i | z_i = k, R_w) P(z_i = k | R_w) \right)^{z_{ik}} . \quad (5.12)$$

5.5.2 The E-step

During this phase we compute sufficient statistics for the posterior on the indicators given a fixed estimate of R_w . This posterior is

$$q^t(Z) = P(Z | L, R_w^t) \quad (5.13)$$

$$= \frac{\prod_i P(\mathbf{l}_i, z_i | R_w^t)}{\int \prod_i P(\mathbf{l}_i, z_i | R_w^t) dZ} \quad (5.14)$$

Consider the term on the top line,

$$P(\mathbf{l}_i, z_i | R_w^t) = P(\mathbf{l}_i | z_i, R_w^t)P(z_i | R_w^t). \quad (5.15)$$

As a function of z_i this is simply a categorical distribution: z_i can take on 3 possible values, and we can compute each explicitly since R_w^t and \mathbf{l}_i are fixed. Now since equation (5.14) is a normalised concatenation of categorical distributions, it is itself a categorical distribution, meaning that the sufficient statistics for q^t are the expectations

$$\mathbb{E}_{q^t}[z_{ik}] = \int z_{ik} q^t(Z) dZ \quad (5.16)$$

$$= \sum_{z_{ik} \in \{0,1\}} z_{ik} P(z_{ik} | \mathbf{l}_i, R_w^t) \quad (5.17)$$

$$= P(z_i = k | \mathbf{l}_i, R_w^t) \quad (5.18)$$

$$= \frac{P(\mathbf{l}_i, z_i = k | R_w^t)}{\sum_{j=0}^4 P(\mathbf{l}_i, z_i = j | R_w^t)} \quad (5.19)$$

where in the third line we have expanded the sum over the two possible values for z_{ik} and cancelled the one in which z_{ik} is 0. Substituting the specific forms for the the complete data likelihood (5.12),

$$\mathbb{E}_{q^t}[z_{ik}] = \frac{\prod_{k=1}^4 \left(P(\mathbf{l}_i | z_i = k, R_w^t) P(z_i = k | R_w^t) \right)^{z_{ik}}}{\sum_{z_i=1}^4 \prod_{k=1}^4 \left(P(\mathbf{l}_i | z_i = k, R_w^t) P(z_i = k | R_w^t) \right)^{z_{ik}}} \quad (5.20)$$

$$= \frac{\left(\pi_{\text{sp}} P(\mathbf{l}_i | z_i = \text{sp}, R_w^t) \right)^{z_{ik}} \prod_{k=1}^3 \left(\pi_k P(\mathbf{l}_i | z_i = k, R_w^t) \right)^{z_{ik}}}{\sum_{z_i=1}^4 \left[\left(\pi_{\text{sp}} P(\mathbf{l}_i | z_i = \text{sp}, R_w^t) \right)^{z_{ik}} \prod_{k=1}^3 \left(\pi_k P(\mathbf{l}_i | z_i = k, R_w^t) \right)^{z_{ik}} \right]}. \quad (5.21)$$

We now define

$$\rho_i = \pi_{\text{sp}} P(\mathbf{l}_i | z_i = \text{sp}, R_w^t) \quad (5.22)$$

$$s_i = \begin{cases} 1, & \text{if } z_i = \text{sp} \\ 0, & \text{otherwise} \end{cases} \quad (5.23)$$

and substituting into (5.21) we have

$$\mathbb{E}_{q^t}[z_{ik}] = \frac{\rho_i^{s_i} (\pi_{z_i} \mathcal{N}(d(\mathbf{l}, R_i R_w^t \mathbf{e}_k); 0, \sigma))^{\frac{1}{1-s_i}}}{\sum_{z_i=1}^4 \rho_i^{s_i} (\pi_{z_i} \mathcal{N}(d(\mathbf{l}, R_i R_w^t \mathbf{e}_{z_i}); 0, \sigma))^{\frac{1}{1-s_i}}} \quad (5.24)$$

$$= \frac{\rho_i^{s_i} (\pi_{z_i} \mathcal{N}(d(\mathbf{l}, R_i R_w^t \mathbf{e}_k); 0, \sigma))^{\frac{1}{1-s_i}}}{\rho_i + \sum_{z_i=1}^3 (\pi_{z_i} \mathcal{N}(d(\mathbf{l}, R_i R_w^t \mathbf{e}_{z_i}); 0, \sigma))}. \quad (5.25)$$

The E-step therefore consists of computing the expectations for each indicator according to (5.25), which are sufficient statistics for the posterior on Z . The expression above can be evaluated for each indicator in constant time, leading to a complexity for the E-step linear in the number of observed line segments.

5.5.3 M-step

During this phase we compute R_w^{t+1} as the maximiser of the expectation of the complete data log-likelihood under the (fixed) distribution q^t . The logarithm of the complete data log-likelihood (5.12) is

$$\log P(L, Z | R_w) = \sum_i \sum_{k=1}^4 z_{ik} (\log P(\mathbf{l}_i | z_i = k, R_w) + \log P(z_i = k | R_w)) \quad (5.26)$$

$$= \sum_i \sum_{k=1}^4 z_{ik} (\log \mathcal{N}(d(\mathbf{l}_i, R_i R_w \mathbf{e}_k); 0, \sigma) + \log \pi_k) \quad (5.27)$$

$$= \sum_i \sum_{k=1}^4 z_{ik} \left(\frac{-d(\mathbf{l}_i, R_i R_w \mathbf{e}_k)^2}{2\sigma^2} + \log \pi_k \right) + c \quad (5.28)$$

where c is a constant arising from the logarithm of the normal distribution, which we now drop since it plays no part in the maximisation to follow. The expectation of the above with respect

to the distribution q^t is

$$\mathbb{E}_{q^t} [\log P(L, Z | R_w)] = \sum_i \sum_{k=1}^4 \mathbb{E}_{q^t} [z_{ik}] \left(\frac{-d(\mathbf{l}_i, R_i R_w \mathbf{e}_k)^2}{2\sigma^2} + \log \pi_k \right) \quad (5.29)$$

$$= \sum_i \sum_{k=1}^4 r_{ik}^t \left(\frac{-d(\mathbf{l}_i, R_i R_w \mathbf{e}_k)^2}{2\sigma^2} + \log \pi_k \right) \quad (5.30)$$

where in the last line we have defined and substituted

$$r_{ik}^t = \mathbb{E}_{q^t} [z_{ik}] . \quad (5.31)$$

The maximisation we wish to perform is

$$R_w^{t+1} = \operatorname{argmax}_{R_w} \mathbb{E}_{q^t} [\log P(L, Z | R_w)] . \quad (5.32)$$

It is worth examining the expression being maximised and noting which variables it is and is not a function of. During each M-step, the distribution q^t , as represented by the sufficient statistics computed in the E-step, is held fixed. The expectation is over all possible values of the indicators Z , each weighted according to q^t , which was computed from the previous estimate R_w^t but does not depend on the quantity R_w being maximised in this step. Hence the expression being maximised in (5.32) is a function of R_w alone. Substituting (5.30),

$$R_w^{t+1} = \operatorname{argmax}_{R_w} f(R_w) \quad (5.33)$$

$$f(R_w) = \sum_i \sum_{k=1}^4 r_{ik}^t \left(\frac{-d(\mathbf{l}_i, R_i R_w \mathbf{e}_k)^2}{2\sigma^2} + \log \pi_k \right) . \quad (5.34)$$

Importantly, note that r_{ik}^t is a constant with respect to R_w in the above. We perform the maximisation (5.32) by gradient descent. Differentiating (5.34) with respect to R_w ,

$$\frac{\partial f}{\partial R_w} = \sum_i \sum_{k=1}^4 r_{ik}^t \left(\frac{-d_{ik}}{\sigma^2} \frac{\partial d_{ik}}{\partial R_w} \right) \quad (5.35)$$

$$\frac{\partial d_{ik}}{\partial R_w} = \frac{1}{\mathbf{a}_i \cdot \mathbf{e}_3} \left(\frac{(\mathbf{m}_{ik})_1 \mathbf{e}_1 + (\mathbf{m}_{ik})_2 \mathbf{e}_2}{\eta_{ik}^3} (\mathbf{m}_{ik} \cdot \mathbf{a}_i) - \frac{\mathbf{a}_i}{\eta_{ik}} \right) \left[\frac{\mathbf{a}_i + \mathbf{b}_i}{2} \right] \times \frac{\partial \mathbf{v}_k}{\partial R_w} \quad (5.36)$$

where

$$d_{ik} = d(\mathbf{l}_i, R_i R_w \mathbf{e}_k) \quad (5.37)$$

$$\eta_{ik} = \sqrt{(\mathbf{m}_{ik} \cdot \mathbf{e}_1)^2 + (\mathbf{m}_{ik} \cdot \mathbf{e}_2)^2} \quad (5.38)$$

and

$$[\mathbf{x}]_\times = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix} \quad (5.39)$$

is the matrix form for the cross product.

For the purpose of optimisation we write R_w as the product of a fixed part R_0 and a variable part R_δ , the latter of which is represented in the Lie algebra for the special orthogonal group $SO(3)$, so

$$R_w = R_0 R_\delta \quad (5.40)$$

$$R_\delta = \exp\left(\sum m_i G_i\right) \quad (5.41)$$

where the G_i are the generator matrices for $SO(3)$ and the m_i provide a minimal representation for the 3D rotation matrix group. The advantage of using this representation is that after each gradient step we are guaranteed that R_w remains a pure rotation, unlike under other representations such as optimising the elements of the 3×3 rotation matrix directly. Differentiating \mathbf{v} with respect to \mathbf{m} yields

$$\frac{\partial \mathbf{v}_k}{\partial \mathbf{m}} = R_i \frac{\partial R_w}{\partial \mathbf{m}} \mathbf{e}_k. \quad (5.42)$$

The division of R_w into fixed and variable parts allows us to evaluate the gradient at $\mathbf{m} = 0$ without loss of generality. In this case,

$$\left. \frac{\partial \mathbf{v}_k}{\partial \mathbf{m}} \right|_{\mathbf{m}=0} = \begin{bmatrix} R_i R_0 G_1 \mathbf{e}_k & R_i R_0 G_2 \mathbf{e}_k & R_i R_0 G_3 \mathbf{e}_k \end{bmatrix}. \quad (5.43)$$

This completes the derivation of the gradient of the error function f with respect to the parametri-

sation \mathbf{m} . We proceed as normal with gradient steps of the form

$$\mathbf{m}^{t+1} = \mathbf{m}^t - \gamma \frac{\partial f}{\partial \mathbf{m}} . \quad (5.44)$$

Due to the low dimensionality of the search space and relatively low curvature of the error function we found that a simple instantiation of the gradient descent algorithm converged quickly and robustly. Our algorithm initialises the step size γ to a fixed constant γ_0 at the beginning of each M-step, then divides γ by 2 each time that an update leads to an increase in the error function. Convergence is detected when $\gamma < \epsilon_1$ and the value of the error function decreases by less than ϵ_2 in any one update.

5.5.4 Summary

To obtain R_w we iterate between updating q (the E-step) and optimising R_w (the M-step). Each M step consists of a gradient descent in the Lie group $SO(3)$. In practice we find that our system converged in around 25 iterations of the EM algorithm, and that approximately 10 steps were required for each gradient descent.

5.6 Results

We evaluated our approach using a dataset of 18 video sequences of 8 unique places, which we collected in indoor environments such as homes and office spaces. We ran a off-the-shelf bundle adjuster on each sequence to yield calibrate views, then transformed each sequence by a random 3D rotation to eliminate any bias in the initial conditions for our optimisation that might have been introduced by structure–from–motion. We manually provided the ground truth scene orientation R_w for each sequence. We sampled frames at regular intervals of approximately one second, then divided all sequences into successive 7-frame blocks. Each such block is a single “evaluation instance” in our dataset.

We compared four approaches including our own. First, the surface–normal–based approach of [25] described at the beginning of this chapter, which we refer to by symbol “NRM”. For each evaluation instance, this algorithm received as input all reconstructed 3D points that were visible in any of the 7 views in that instance. We were generous to this approach here since the reconstruction of some of these points may have relied upon data from other frames. Second, we tested a system identical to ours but in which the reconstruction error was replaced by the algebraic error — we refer to this system as “ALG” below. Finally, we evaluated our own system using both a single input view (“SIN”), and all 7 views (“MUL”). The former approach is hence similar to that of Schindler and Dellaert [57].

The metric we used to compare the estimated and ground truth rotations was

$$\|R_1^T R_2 - I\|_{\mathcal{F}} . \quad (5.45)$$

where $\|\cdot\|_{\mathcal{F}}$ denotes the Frobenius norm. We chose this metric because it was shown by Huynh [35] to be iso-equivalent to many 3D rotation metrics in common usage within the computer vision literature, including quaternion norms and inner products, and metrics based on Euler angles and the Lie algebra $so(3)$.

Table 5.1 summarises the performance of each algorithm. Unsurprisingly, the surface normal approach fails under the sparse structure–from–motion point clouds — it was proposed in the context of dense reconstructions. Strikingly, the multiple–view algebraic minimiser performs consistently worse than single–image estimation. Our investigations suggest that this is because the algebraic error is an even poorer approximation in the multiple view context than in the single–view context, since among many frames there is a greater chance of observing vanishing points close to infinity, which is where the algebraic error breaks down.

We ran a separate experiment to explore the relationship between the number of frames used for estimation and the quality of the estimated rotation; these results are shown in Figure 5.3. As the number of frames used for estimation increases, the quality of the estimator described in this chapter improves as expected. However, it is interesting to note that the algebraic min-

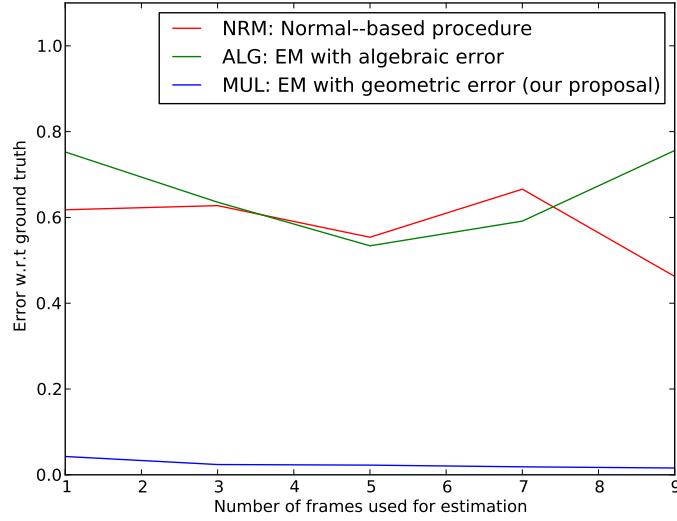
Sequence	Error for NRM	Error for ALG	Error for SIN	Error for MUL
Kitchen	0.53	0.59	0.037	0.019
Foyer 1	0.72	0.77	0.054	0.39
Foyer 2	0.81	0.75	0.070	0.021
Bursary	0.50	0.53	0.067	0.030
Ground	0.90	0.68	0.55	0.66
Atrium	0.84	1.00	0.31	0.16
MCR	0.56	0.56	0.06	0.04
Corridor	0.58	0.78	0.20	0.21

Table 5.1: Distance between estimated and ground truth rotation for four algorithms. Each cell shows the average error over all frames within a given sequence. From left to right the algorithms are the point–cloud procedure of [25] (NRM), Expectation–Maximisation using the algebraic error (ALG), Expectation–Maximisation using the geometric error applied to single (SIN) and multiple (MUL) images. All algorithms other than SIN were given 7 input frames in this experiment.

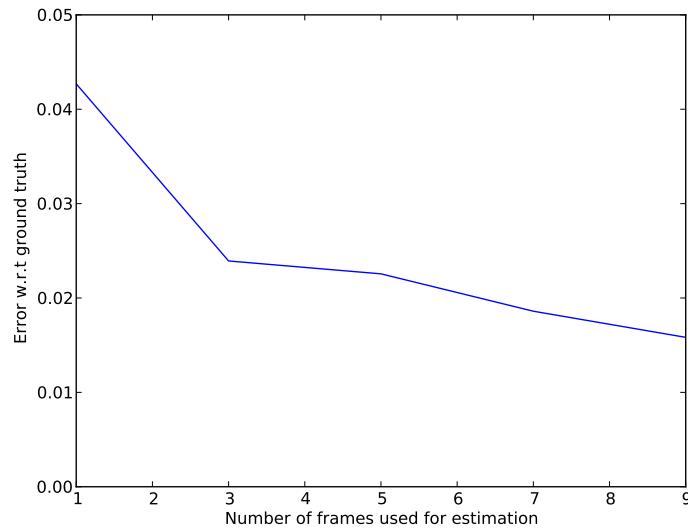
imiser actually degrades for more than 5 input frames. This supports our hypothesis above that the algebraic error is an even poorer choice in the multiple–view context than for single images.

Our third experiment explored the convergence properties of the gradient descent component of our optimisation. Starting from an initial rotation a specified distance from the ground truth, we plotted the progress of our algorithm after each gradient step (Figure 5.4). In this experiment the indicators Z were assumed known. Progress was measured both in terms of the log–likelihood, which the optimiser itself has access to (lower figure), and distance to ground truth, which the optimiser obviously does not have access to (upper figure). This experiment shows two things. First, that the basin of attraction is very wide — convergence is robust for distances up to 1.0, which is a large distance in the metric (5.45). Second, that the log-likelihood accurately tracks the ground truth error in practice, since the runs that did not converge to the ground truth also terminated at much lower log–likelihoods.

Finally, examples of the output from the four systems described above are shown in Figures 5.5, 5.6, and 5.7.



(a) Estimation error versus number of input frames.



(b) A magnification of the third series above.

Figure 5.3: Plots showing the performance of three algorithms as the number of input frames is increased. We compute each data point by executing the relevant algorithm on sets of n consecutive frames drawn from our dataset, where n is the value shown on the x -axis. The y -axis then shows the average distance between the estimated and ground truth rotation. The lower figure shows the series corresponding to our algorithm on an magnified set of axes.

5.7 Identifying The Vertical Direction

Of the three dominant directions defined by R_w , two correspond to horizontal directions and the third to the vertical direction. The latter is semantically distinct since it defines the orientation of the ground and ceiling planes, as well as the direction in which gravity operates. It is easy to identify the vertical axis since humans necessarily move over the ground plane when capturing video sequences, and have limited scope for moving the camera in the up–down direction. We therefore set the vertical axis to that over which camera positions range the least. Having identified R_w there are only three possible choices, and we found this heuristic to work correctly in all of our evaluation sequences.

5.8 Identifying The Floor And Ceiling Planes

An indoor Manhattan scene has exactly one floor and one ceiling plane, both with normal v_v . It will be useful in the following chapters to have available the mapping H_a between the image locations of ceiling points the floor points that are vertically below them (see Figure 5.8). H_a is a planar homology with axis $\mathbf{h} = \mathbf{v}_l \times \mathbf{v}_r$ and vertex \mathbf{v}_v [11] and can be recovered given the image location of any pair of corresponding floor/ceiling points $(\mathbf{p}_f, \mathbf{p}_c)$ as

$$H_a = I + \mu \frac{\mathbf{v}_v \mathbf{h}^T}{\mathbf{v}_v \cdot \mathbf{h}} , \quad (5.46)$$

where

$$\mu = \langle \mathbf{v}_v, \mathbf{p}_c, \mathbf{p}_f, \mathbf{p}_c \times \mathbf{p}_f \times \mathbf{h} \rangle \quad (5.47)$$

is the characteristic cross ratio of H_a . Although we do not have *a priori* any such pair $(\mathbf{p}_f, \mathbf{p}_c)$, we can recover H_a using the following sampling algorithm. First we identify edges in the image using one of the many available edge detection algorithms. Next we sample one edge pixel \hat{x}_c from the region above the horizon, then we sample a second point \hat{x}_f collinear with the first and \mathbf{v}_v from the region below the horizon. We then compute a hypothesis for \hat{H}_a as described above, and then assign a score by counting the number of edge pixels that \hat{H}_a maps onto other

edge pixels. After repeating this for a fixed number of iterations we return the hypothesis with greatest score.

Many images contain either no view of the floor or no view of the ceiling. In such cases H_a is unimportant since there are no corresponding points in the image. If the best H_a output from the sampling process has a score below a threshold k_t then we set μ to a large value that will transfer all pixels outside the image bounds. H_a will then have no impact on the estimated model.

In the context of multiple views we resolve scale by identifying λ with either the maximum or minimum z component of any point in the point cloud reconstructed by structure–from–motion.

5.9 Extension: Relaxing The Manhattan World Assumption

The strong Manhattan assumption states that any pair of surfaces of interest are either parallel or orthogonal to one another. One common deviation from this is scenes with walls that are orthogonal to the floor and ceiling but not to one another. We define the weak Manhattan assumption as “the environment consists of a horizontal ground plane and corresponding ceiling plane, and a set of vertical wall segments extending continuously between them.” Weakly Manhattan environments contain much of the regularity of strongly Manhattan environments, and here we discuss how to extending the approach taken in chapter to such scenes. We have not implemented this approach; we discuss these ideas as inspiration for future work.

We can deal with the weak Manhattan assumption as follows. First, we run the EM algorithm described above to obtain R_w . Next, for each line \mathbf{l}_j marked as spurious by the EM algorithm we find its intersection with the horizon,

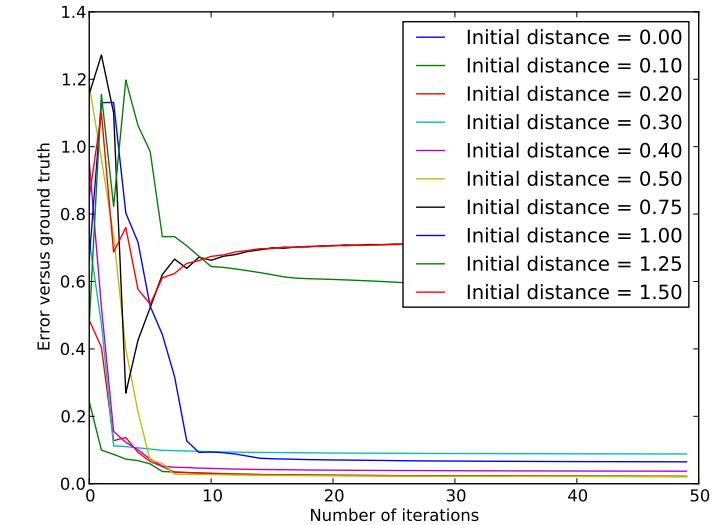
$$\mathbf{u}_j = R_w^{-T} R_i^{-T} \mathbf{l}_j \times \mathbf{e}_3 , \quad (5.48)$$

which would be its vanishing point if it were horizontal in the world. Vertical surfaces of a

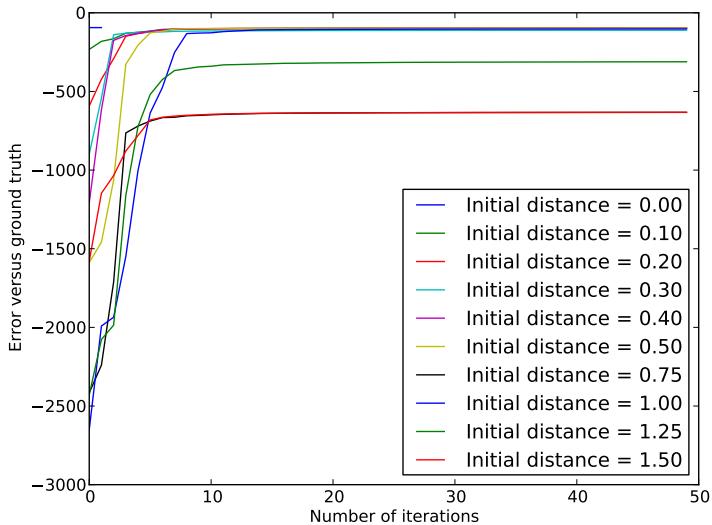
given orientation will generate identical \mathbf{u}_j (modulo measurement error), so we may identify additional vertical orientations by clustering the intersections $\{\mathbf{u}_j\}$. This should be possible even with few such intersections, since the fact that all intersections are on the horizon reduces this to a one-dimensional estimation problem.

5.10 Conclusion

We have proposed a principled approach to discovering the dominant Manhattan directions given multiple calibrated views. Our likelihoods are derived from a well-defined model of image generation, and we solve inference as a single optimisation. Our work is strongly influenced by the literature on vanishing point detection and our approach parallels ideas previously proposed in the single-view context. Our contribution is a principled way to incorporate multiple calibrated views into a single optimisation, and an empirical demonstration that doing so significantly out-performs both single-view vanishing point estimation and surface-normal-based estimators.



(a) Convergence of error (with respect to ground truth)



(b) Convergence of log-likelihood

Figure 5.4: Illustration of the convergence properties of our gradient descent algorithm. We measure the distance from the estimated rotation to the ground truth after each gradient step. Each series above shows this evolution when our algorithm is initialised a particular distance from the ground truth. We see that for distances less than 1 convergence is robust. This corresponds to a large offset under the metric (5.45), indicating that the optima has a wide basin of attraction. Note that for this experiment the labels Z are assumed known.

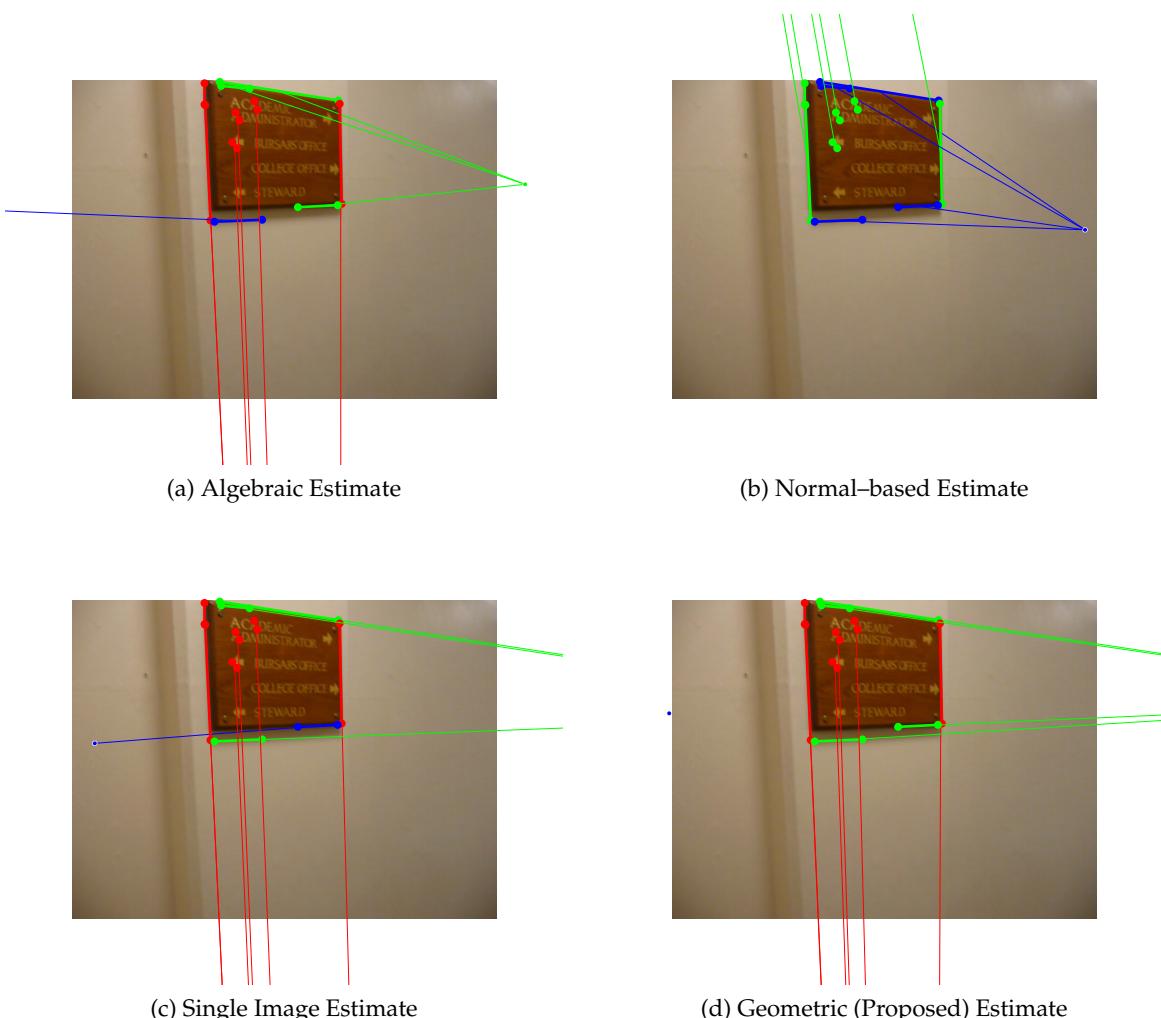


Figure 5.5: Comparison of rotation estimation algorithms for an example drawn from the “Bur-sary” sequence. Though only one frame is shown here, each algorithm other than (c) was provided with 7 input frames.

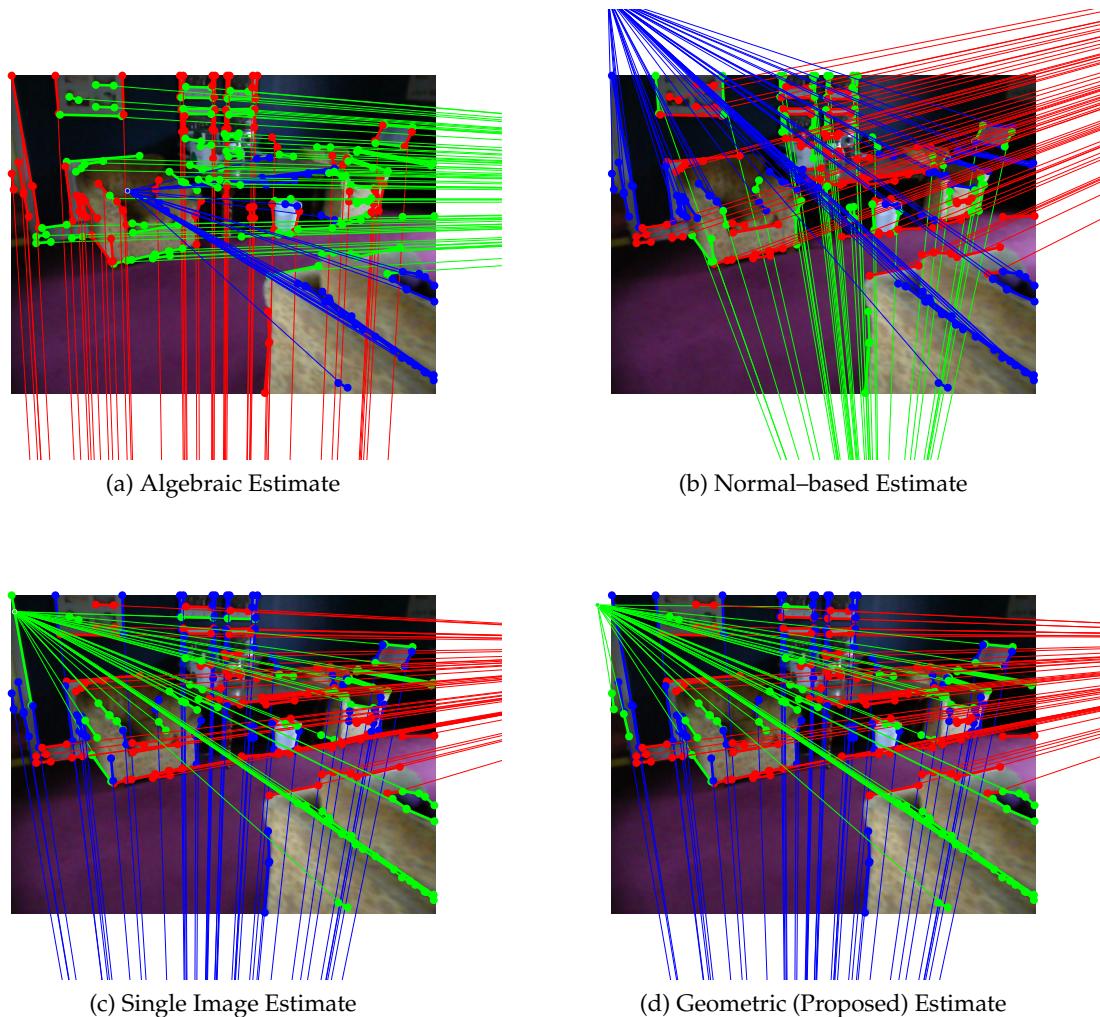


Figure 5.6: Comparison of rotation estimation algorithms for an example drawn from the “MCR” sequence. Though only one frame is shown here, each algorithm other than (c) was provided with 7 input frames.

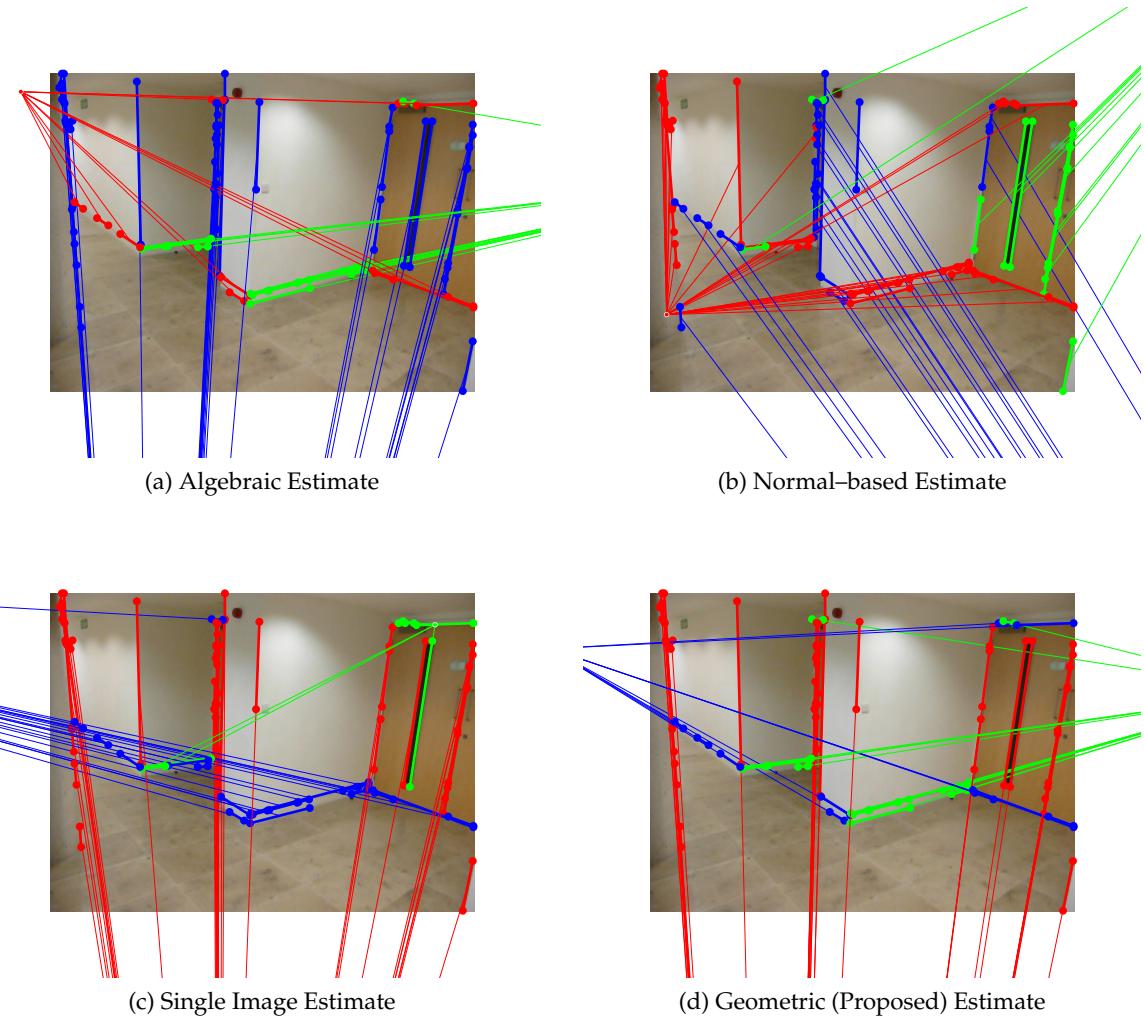


Figure 5.7: Comparison of rotation estimation algorithms for an example drawn from the “Ground1” sequence. Though only one frame is shown here, each algorithm other than (c) was provided with 7 input frames.

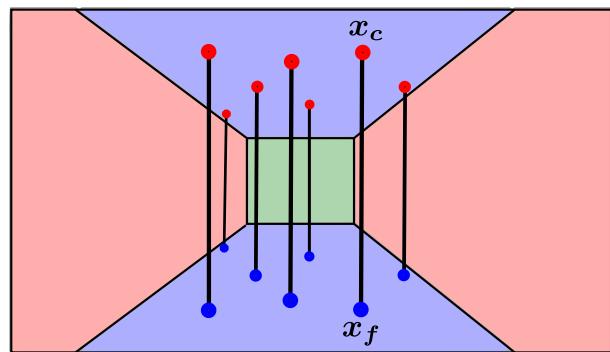


Figure 5.8: The mapping H_a transfers points between the ceiling and floor in the image domain. H_a is a planar homology.

6

Inference From Single and Multiple Views

This chapter addresses the recovery of indoor Manhattan models in the context of single and multiple views of a scene. We describe a Bayesian approach to reasoning about indoor Manhattan models in the face of ambiguous image evidence. To achieve this we present a graphical model that relates photometric cues, stereo photo-consistency, and depth cues to the scene model discussed in previous chapters. We show how to solve MAP inference using dynamic programming, allowing exact, global inference in ~ 100 ms without using specialised hardware. Our approach is applicable to both the single- and multiple-view settings by selecting various combinations of sensor models. Experiments show our system out-performing the state-of-the-art in the domain of indoor Manhattan reconstruction.¹

¹This work was published in part in:

Flint, Mei, Murray, and Reid, “A Dynamic Programming Approach To Reconstruction Building Interiors”, in Proceedings of the 2010 European Conference on Computer Vision[20]

Flint, Murray, and Reid, “Manhattan Scene Understanding Using Monocular, Stereo, and 3D Features”, in Proceedings of the 2011 International Conference on Computer Vision[23]

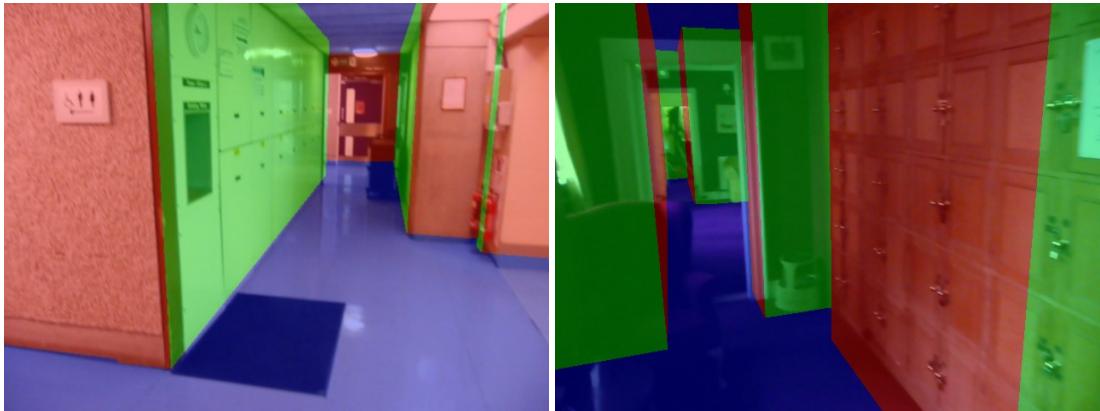


Figure 6.1: Scene structure recovered by the system described in this chapter.

6.1 Introduction

Over the past decade, computer vision researchers working with monocular images have pursued substantially different research agendas to those working with multiple views. The focus for monocular images has increasingly been to infer high-level facts about the world, such as the locations of and interactions between objects, semantic scene categories, and the spatial layout of the environment. In contrast, much of the work concerning multiple views has focused on reconstructing metric scene structure and camera poses using techniques such as structure–from–motion, stereo, and multiple–view stereo.

In this chapter we leverage multiple view geometry for image understanding purposes. We assume a moving camera with a structure–from–motion system estimating its trajectory, and show how to infer semantically meaningful models of the environment. We focus on the *indoor Manhattan representation*[13, 20] that we described in Chapter 4.

Our approach is to define a probabilistic model that relates the unknown scene layout to three types of observations: photometric image features, stereo photo–consistency, and 3D point clouds. These three quantities are commonly available in a moving camera setup, but our system can be used unchanged with any combination of the three, including a single–view setting. The second part of this chapter then focusses on finding the most likely explanation for a set of observations given the assumptions made by our model. A major focus of this chapter is the development of an efficient and exact dynamic programming that solves both

maximum–aposteriori (MAP) and maximum–likelihood (ML) inference in our model.

To present these two components (model and inference algorithm) clearly, we begin by describing a class of optimisation problem that we call the payoff formulation. This optimisation problem provides the connection between the model we present first and the inference algorithm we present second. In particular, we will show during the development of the probabilistic model that MAP inference can be written as an optimisation problem in payoff form. After we have presented our model, we will thereafter be interested only in solving general payoff-form problems, which allows us to decouple the development of our dynamic programming algorithm from the specifics of our probabilistic model — we will simply show that we can solve all optimisation problems in the payoff formulation. A further advantage is that other sensor models that can be similarly reduced to payoff form will also be amenable to our dynamic programming algorithm. Although the presentation of an abstract class of optimisation problem may seem an abstruse way to begin this chapter, we believe that it leads to the clearest presentation overall.

6.2 The Payoff Formulation

In this section we describe a class of optimisation problems that we will refer to as the *payoff formulation*. Throughout the remainder of this chapter it will become clear that a range of inference problems in the context of indoor Manhattan scenes can be expressed in this form, and that this particular formulation permits a general and efficient dynamic programming solution.

Let \mathcal{M} be the set of indoor Manhattan scenes in vertex representation. Let $M \in \mathcal{M}$ be a scene in vertex representation and let S be the corresponding scene in seam representation.

A payoff function $\pi : \mathcal{D} \times \mathcal{A} \rightarrow \mathbb{R}$ maps integer pixel coordinates $p \in \mathcal{D}$ together with orientations $a \in \mathcal{A}$ to real numbers. The payoff for a scene M is defined in terms of the seam representation,

$$\Pi(M) = \Pi(S) = \sum_{j=1}^W \pi(j, s_j, o_j) \quad (6.1)$$

where $S = \{(s_j, o_j)\}$. This sum consists of one term for each image column. It is computed by adding together the value of the payoff function at each pixel along the floor/wall seam in M . The form of the payoff function is problem-specific; all we require is that it is a function of j , s_j , and o_j . Later we will describe specific payoff functions π for the likelihoods in our model. In cases where the payoff function is independent of o_j we will write

$$\pi(\mathbf{p}) = \pi(x, y, \cdot) \quad (6.2)$$

Note that the value of $\pi(\mathbf{p})$ is *not* restricted in any way to dependence on the image evidence at pixel \mathbf{p} , nor even to a local region about \mathbf{p} ; indeed, the payoff functions described in the following sections incorporate image evidence from widely separated image regions.

A penalty function $\Gamma : \mathcal{M} \rightarrow \mathbb{R}$ maps scenes in the vertex representation to real numbers and takes the form,

$$\Gamma(M) = \sum_{i=0}^{K-1} \gamma(i; M) \quad (6.3)$$

where K is the number of walls contained in M and γ may be interpreted as a regulariser related to the meeting between the i^{th} wall in M and its successor. Once again, we have not defined any particular γ ; any that takes the the form (6.3) is suitable for the optimisation problem to follow. Later we will describe specific forms for γ corresponding to particular choices of prior in model.

Given any such Π and Γ , the associated optimisation problem is as follows. Let

$$f(M) = \Pi(M) - \Gamma(M) \quad (6.4)$$

$$= \sum_{j=1}^W \pi(j, s_j, o_j) - \sum_{i=0}^{K-1} \gamma(i; M). \quad (6.5)$$

then we seek

$$\hat{M} = \underset{M \in \mathcal{M}}{\operatorname{argmax}} f(M). \quad (6.6)$$

Note that the objective f is defined partially in the scene representation and partially in the ver-

tex representation. This poses no difficulty to evaluating f in the vertex representation since we can readily obtain the seam representation via equation (4.57). However, as discussed in Section 4.4.3, the mapping from the vertex representation to seam representation is not invertible, so the objective above cannot be evaluated in the seam representation. For this reason we will work in the vertex representation for the presentation of the optimisation algorithm, but for convenience we will work in seam representation to define Π .

6.3 Probabilistic Model

In this section we develop a probabilistic model to guide the recovery of indoor Manhattan scenes from observations. We assume that three types of observations are available: photometric image features, calibrated stereo pairs, and 3D point clouds. These quantities might be acquired, for example, as the output of a SLAM or structure–from–motion system. We now describe probabilistic relationships between each type of observation and the indoor Manhattan scene structure that we wish to infer.

6.3.1 Scene Prior

We turn first to the prior on scenes, $P(M | \lambda)$, which is governed by a set of hyper-parameters λ . As discussed in Section 4.5.1, corners between successive walls can be classified as concave, convex, and occluding. Let n_1, n_2 , and n_3 be the number of corners in M of each category respectively. Our prior on scenes is a geometric distribution in $\mathbf{n} = (n_1, n_2, n_3)$,

$$P(M | \lambda) = \frac{1}{Z}, \quad (6.7)$$

where Z is a normalizing term that we will not need to compute in order to perform maximisation. Our choice of (6.7) is motivated by the desire to penalise scenes for additional complexity, where we measure complexity by the number of distinct walls. We discuss other priors, and some of the problems they raise in Section 6.7.1.

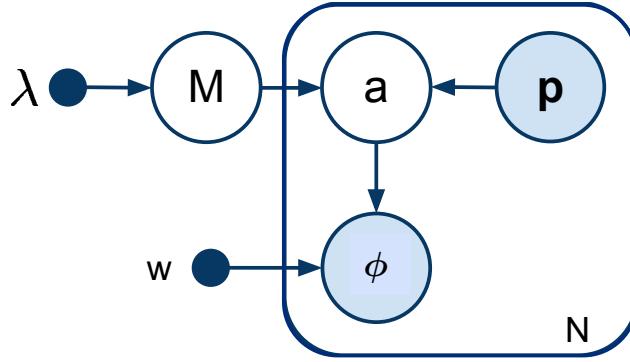


Figure 6.2: The graphical model relating scenes M to monocular image features ϕ . $p = (x, y)$ is a pixel location and a is the orientation predicted (deterministically) by M at p .

Taking logarithms yields

$$\log P(M \mid \boldsymbol{\lambda}) = -\log Z + n_1 \log \lambda_1 + n_2 \log \lambda_2 + n_3 \log \lambda_3 . \quad (6.8)$$

Comparison with (6.3) suggests the following penalty function:

$$\gamma(i; M) = \begin{cases} \log \lambda_1, & \text{if the } i^{\text{th}} \text{ corner in } M \text{ is concave} \\ \log \lambda_2, & \text{if the } i^{\text{th}} \text{ corner in } M \text{ is convex} \\ \log \lambda_3, & \text{if the } i^{\text{th}} \text{ corner in } M \text{ is occluding} \end{cases} \quad (6.9)$$

The cases above can be decided by the algorithm described in Section 4.5.1. Substituting (6.9) into (6.3) yields

$$\Gamma(M) = \log P(M \mid \boldsymbol{\lambda}) + \log Z . \quad (6.10)$$

We can safely ignore the constant term since our ultimate goal is the optimisation expressed in (6.6).

6.3.2 Photometric Sensor Model

We now introduce a model relating indoor Manhattan scenes to observed image features. We begin with the single-view scenario in which we observe a feature $\phi \in \mathbb{R}^n$ at each pixel p .

We assume the graphical model shown in Figure 6.2. The generative process begins by sam-

pling a scene M , then samples pixels p and computes their orientation $a \in \{1, 2, 3\}$ (*c.f.* Algorithm 4.2), which is deterministic given M and is included in the graphical model for notational convenience only. Finally a feature ϕ is sampled from a distribution that is conditionally independent of M given a .

We assume a exponential-family likelihood for pixel features,

$$P(\phi | a) \propto \exp(\boldsymbol{\omega}_a \cdot \phi). \quad (6.11)$$

Denoting the set of all observed pixel features Φ , pixels P , and orientations A , the joint distribution is

$$P(\Phi, P, A, M, \lambda, \omega) = P(M | \lambda) \prod P(p_i) P(a_i | M, p_i) P(\phi_i | a_i, \omega). \quad (6.12)$$

We do not model the distribution $P(p_i)$ and for the remainder of this section it may be assumed that all probabilities are conditioned on this quantity.

We now derive MAP inference. The likelihood for M is

$$P(\Phi | M) \propto \int \prod_i P(\phi_i | a_i) P(a_i | M) dA. \quad (6.13)$$

In the integration over the latent variables a_i the only non-zero term is the one for which all a_i are equal to that predicted by Algorithm 4.2. Therefore, denoting by a_i^* the orientation output by Algorithm 4.2 for pixel p_i under M we have

$$P(\Phi | M) \propto \prod_i P(\phi_i | a_i^*). \quad (6.14)$$

Taking logarithms gives

$$\log P(\Phi | M) = \sum_i \log P(\phi_i | a_i^*) + c \quad (6.15)$$

where c corresponds to the constant of proportionality in (6.14), which we henceforth drop since it makes no difference to the optimisation to come.

At this point we use the crucial observation of Section 4.5.4 that the orientation a_i^* is functionally dependent only on the pair (s_j, o_j) for the column j containing p_i . Let $\bar{a}(x, y; s_j, o_j)$ be the orientation output by Algorithm 4.2 for pixel (x, y) under the hypothesis (s_j, o_j) . We define

$$\pi_{\text{mono}}(x, y, o) = \sum_{r=0}^H \log P(\phi_{xr} | \bar{a}(x, r; y, o)) \quad (6.16)$$

where the double-subscript in ϕ_{xr} is a result of separately indexing rows and columns in (6.16).

Now consider the scene payoff,

$$\Pi(M) = \sum_{j=0}^W \pi_{\text{mono}}(j, s_j, o_j) \quad (6.17)$$

$$= \sum_{j=0}^W \sum_{r=0}^H \log P(\phi_{jr} | \bar{a}(j, r; s_j, o_j)) \quad (6.18)$$

$$= \log P(\Phi | M) + O(1). \quad (6.19)$$

This is simply the log-likelihood (6.15) up to a constant, so maximising (6.19) is equivalent to maximising (6.15). In this sense we have placed our model in the payoff formulation outlined in the previous section since by substituting the particular payoff function π_{mono} into (6.6) we obtain an optimisation problem in payoff formulation that is equivalent to maximum-likelihood inference under the graphical model of Figure 6.2. (Later we will show that this result extends easily to MAP inference also.)

6.3.3 Multiple-View Sensor Model

We now formulate the payoff function π_{stereo} for the case that multiple views of the scene are available. We assume that there one image is identified as the base view I_0 ; the remaining K images are denoted I_1, \dots, I_K . All images have size $W \times H$. We assume that all cameras are calibrated with the action of the i^{th} camera on a 3D point X being

$$K_i(R_i X + \mathbf{t}_i). \quad (6.20)$$

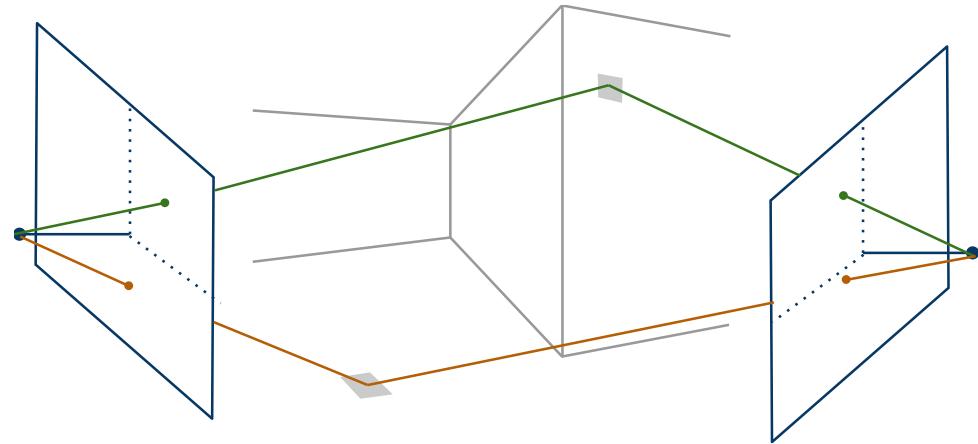


Figure 6.3: Pixel correspondences across multiple views are computed by back-projection onto the model M followed by re-projection into auxiliary views.

We normalise images intensities to zero mean and unit variance.

Intuitively, we treat inference in this settings as follows. We consider models M in terms of their projection into I_0 . We explained in Section 7.3 that models parametrized in image coordinates specify unique 3D models. Any model hypothesized in I_0 can therefore be re-projected into the auxiliary views, giving pixel-wise correspondences between frames (*c.f.* Figure 6.3). From this we compute a photo-consistency measure $\text{PC}(\cdot)$, which provides the likelihood $P(I_0, \dots, I_M | M)$.

Optimising over photo-consistency has been standard in the stereo literature for several decades [56]; our contribution is to show that (i) in the particular case of indoor Manhattan models, photo-consistency can be expressed as a payoff matrix; (ii) that we can therefore perform efficient and exact global optimisation; and (iii) that this fits naturally within a Bayesian framework alongside monocular and 3D features.

In Section 4.5.2 we showed how to convert a scene in vertex representation to a 3D reconstruction. Since all cameras are calibrated we can use this reconstruction to re-project any pixel p in the base view into all auxiliary views, as illustrated in Figure 6.3. Let $q_k(p; M)$ be the re-projection of p into view k via the scene hypothesis M . Let each pixel p_i in the base image be associated with a feature vector $\phi_i \in \mathbb{R}^n$ and let each pixel q_{ki} in the k^{th} auxiliary view be similarly associated with a feature $\theta_{ki} \in \mathbb{R}^n$. Let Φ be the set of all features in the base view

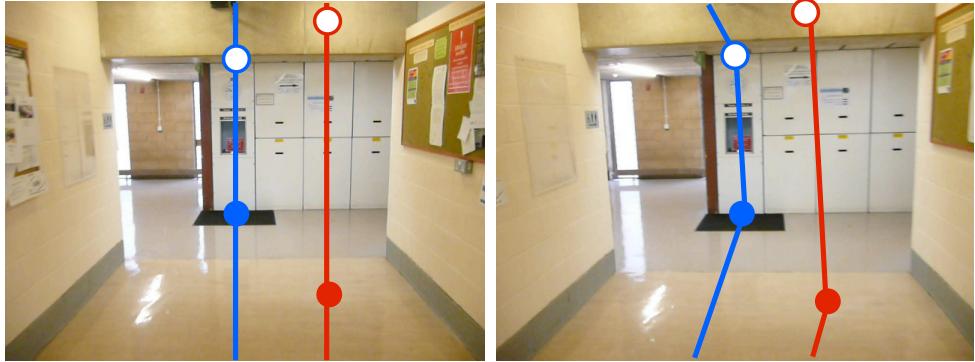


Figure 6.4: Two hypotheses for the location of the seam and the reprojections they imply for each image column in an auxiliary image. The blue dot happens to be a correct hypothesis (unbeknownst to the system), while the red dot is incorrect. As a result, the blue line corresponds to the same set of 3D locations in both images but the red line does not.

and let Θ_k be the set of all features in the k^{th} auxiliary view. Finally, let

$$\bar{\theta}_k(\mathbf{p}_i; M) = \theta(\mathbf{q}_k(\mathbf{p}_i; M)) \quad (6.21)$$

be the feature associated with the reprojeciton of \mathbf{p}_i into the k^{th} auxiliary view under the scene hypothesis M . Then the likelihood for M under our model is

$$P(\Phi, \Theta_1, \dots, \Theta_K \mid M) = \prod_i \prod_{k=0}^K P(\phi_i, \bar{\theta}_k(\mathbf{p}_i; M) \mid \Sigma). \quad (6.22)$$

and following the standard approach [56], the feature likelihood is a zero-mean Gaussian:

$$P(\phi_i, \bar{\theta}_k(\mathbf{p}_i; M) \mid \Sigma) = \mathcal{N}(\phi_i - \bar{\theta}_k(\mathbf{p}_i; M); \Sigma) \quad (6.23)$$

Taking logarithms we recognise a simple sum over pixel-wise photo-consistency terms,

$$\log P(\Phi, \Theta_1, \dots, \Theta_K \mid M) = \sum_i \sum_{k=0}^K \|\phi_i - \bar{\theta}_k(\mathbf{p}_i; M)\|_\Sigma + c \quad (6.24)$$

We now write equation (6.24) in payoff form. To this end we leverage the observation made in Section 4.5.4 that the depth of a pixel \mathbf{p} under a scene hypothesis M is functionally dependent only on the pair (s_j, o_j) for the column j containing \mathbf{p} . Let $\bar{d}(\mathbf{p}; s_j)$ be the depth of \mathbf{p} under the

hypothesis s_j , as output by Algorithm 4.3.

Furthermore, re-projecting a pixel into an auxiliary view requires only the depth at that pixel. Therefore we can identify all correspondences for any pixel p from the value s_j alone. Specifically, the re-projection of p into the k^{th} view under the hypothesis s_j is

$$\mathbf{q}_k(p; s_j) = K_k(R_k X_i + \mathbf{t}_k) \quad (6.25)$$

where

$$X_i = R_0^{-1}(\bar{d}(p; s_j)K_0^{-1}\mathbf{p} - \mathbf{t}_0). \quad (6.26)$$

We can now re-write the correspondence function $\bar{\theta}_{ki}$ in terms of s_j alone,

$$\bar{\theta}_k(p; s_j) = \theta(\mathbf{q}_k(p; s_j)). \quad (6.27)$$

Finally, the payoff function is

$$\pi_{\text{stereo}}(x, y) = \sum_{r=0}^H \sum_{k=1}^K \|\phi_{xr} - \bar{\theta}_k([x, r]^T; y)\|_\Sigma \quad (6.28)$$

where as in (6.16) we have switched to a separate indexing scheme for rows and columns, so ϕ_{xy} is the feature for the pixel at (x, y) and $\bar{\theta}_k(x, r; y)$ the the feature for the reprojection of (x, r) into the k^{th} view under the hypothesis $s_j = y$. Substituting (6.28) into (6.1),

$$\Pi(M) = \sum_{j=0}^W \pi_{\text{stereo}}(j, s_j) \quad (6.29)$$

$$= \sum_{j=0}^W \sum_{r=0}^H \sum_{k=1}^K \|\phi_{jr} - \bar{\theta}_k([j, r]^T; s_j)\|_\Sigma \quad (6.30)$$

$$= \log P(\Phi, \Theta_1, \dots, \Theta_K \mid M) - c \quad (6.31)$$

The completes the reduction of the stereo sensor model to payoff form, since we have shown that maximising the likelihood (6.22) is equivalent to the payoff optimisation problem (6.6) for a particular instantiation of the payoff function π .

A Different View

Our approach could also be cast as solving the general stereo problem in terms of disparity maps, where in place of priors based on pixel-wise smoothness constraints, our prior is (6.7) for those disparity maps that correspond to valid indoor Manhattan reconstructions, and zero for others. Inference under this model would be intractable if cast directly in terms of disparity maps because determining whether a given disparity map corresponds to some indoor Manhattan reconstruction is difficult. Nevertheless, our approach shows that by re-parametrising in the vertex representation the problem becomes tractable.

Note that the column-wise decomposition (6.28) neither commits us to optimising over columns independently, nor to ignoring interactions between columns. By inspecting (6.22) one sees immediately that our model assumes no independence between image columns (only conditional independence given M), and indeed correlations do come into effect when we optimise over the full payoff matrix later in this chapter. Our results will show that widely separated image regions often interact strongly. The derivations in this section follow deductively from the indoor Manhattan assumption; the only approximation is that concerning occlusions, which we discuss below.

Occlusions

We have ignored self-occlusions in (6.22). For short baselines, such as frames sampled over a few seconds from a moving camera, this is unproblematic since indoor environments tend to be mostly convex from any single point of view. Even in highly non-convex environments our system achieves excellent results by integrating 3D and monocular features, and enforcing strong global consistency, as will be shown in our experimental section. Further discussion of this issue is in the final section of this chapter.

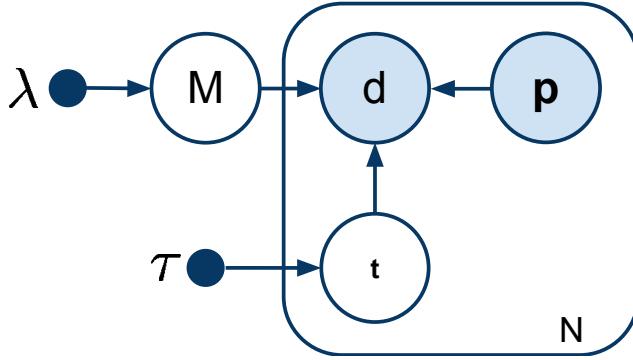


Figure 6.5: The graphical model relating indoor Manhattan models to 3D points. The hidden variable t indicates whether the point is inside, outside, or coincident with the model.

6.3.4 Point Cloud Sensor Model

In this section we explore the context in which a 3D point cloud is available during inference. The point clouds generated by structure–from–motion systems are typically too sparse for direct reconstruction, but can provide useful cues alongside monocular and stereo data.

Our graphical model for 3D data is depicted in Figure 6.5. The model M is sampled according to the prior (6.7), then depth measurements d_i are generated for pixels p_i . Many such measurements will correspond to clutter or measurement errors, rather than to the walls represented by M . Our model captures this uncertainty explicitly through the latent variable t_i , which has the following interpretation. If $t_i = \text{ON}$ then d_i corresponds to some surface represented explicitly in M . Otherwise, either $t_i = \text{IN}$, meaning some clutter object within the room was measured, or $t_i = \text{OUT}$, in which case an object outside the room was measured, such as through a window. The likelihoods we use are

$$P(d | \mathbf{p}, t = \text{IN}, M) = \begin{cases} \alpha, & \text{if } 0 < d < \bar{d}(\mathbf{p}; M) \\ 0, & \text{otherwise} \end{cases} \quad (6.32)$$

$$P(d | \mathbf{p}, t = \text{OUT}, M) = \begin{cases} \beta, & \text{if } \bar{d}(\mathbf{p}; M) < d < d_{\max} \\ 0, & \text{otherwise} \end{cases} \quad (6.33)$$

$$P(d | \mathbf{p}, t = \text{ON}, M) = \mathcal{N}(d ; \bar{d}(\mathbf{p}; M), \sigma). \quad (6.34)$$

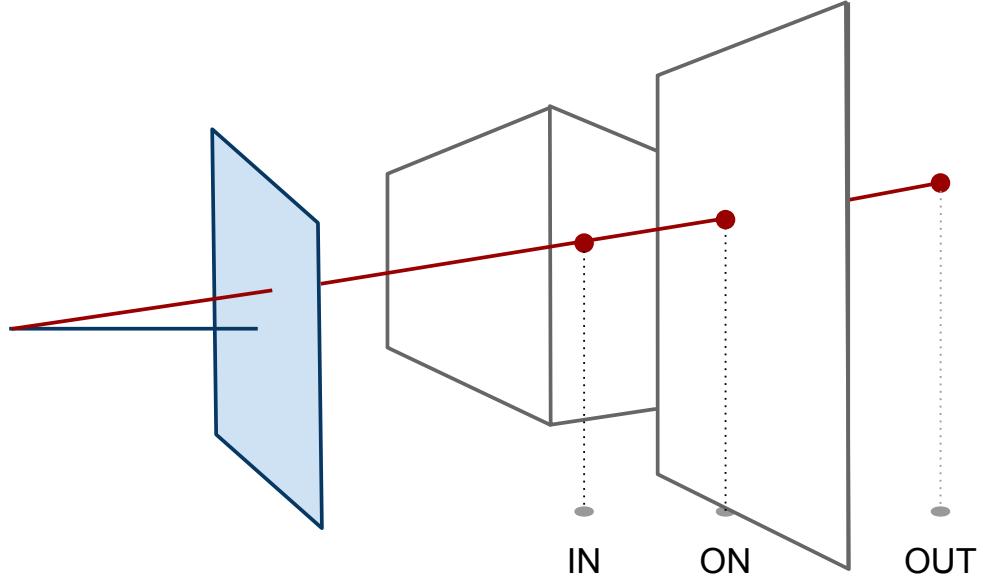


Figure 6.6: Depth measurements d_i might be generated by a surface in our model (represented by $t_i = \text{ON}$) or by an object inside or outside the environment (in which case $t_i = \text{IN}, \text{OUT}$ respectively).

where α and β are determined by the requirement that the probabilities sum to 1 and $\bar{d}(\mathbf{p}; M)$ denotes the depth predicted by M at \mathbf{p} . We compute likelihoods on d by marginalising over t ,

$$P(d | \mathbf{p}, M, \boldsymbol{\tau}) = \sum_t P(d | \mathbf{p}, M, t) P(t | \boldsymbol{\tau}). \quad (6.35)$$

where $P(t | \boldsymbol{\tau})$ is a categorical distribution with parameters τ_{IN} , τ_{OUT} , and τ_{ON} . Equation (6.35) can be readily evaluated for any d and \mathbf{p} since the sum is over just the three possible values for t . Denoting the set of all depth measurements D , the full likelihood for M is

$$P(D | \mathbf{P}, M) = \prod_i P(d_i | \mathbf{p}_i, M) \quad (6.36)$$

$$\log P(D | \mathbf{P}, M) = \sum_i \log P(d_i | \mathbf{p}_i, M) \quad (6.37)$$

We now utilise the same observation as we did in the previous section, namely that the depth at \mathbf{p} is functionally dependent only on the seam pair in the column containing \mathbf{p} . Retaining the notation under which $\bar{d}(\mathbf{p}; s_j)$ is the depth at \mathbf{p} computed by Algorithm 4.3 for M , we define

the payoff function

$$\pi_{3D}(x, y) = \sum_{i \in D_x} \log P(d_i | \mathbf{p}_i, \bar{d}(\mathbf{p}_i; y)) \quad (6.38)$$

where D_x contains indices for all depth measurements in column x . We verify that (6.38) does in fact correspond to the log-likelihood (6.36) by substituting the above into (6.1), giving

$$\Pi(M) = \sum_{j=0}^W \pi_{3D}(j, s_j) \quad (6.39)$$

$$= \sum_{j=0}^W \sum_{i \in D_j} \log P(d_i | \mathbf{p}_i, \bar{d}(\mathbf{p}_i; s_j)) \quad (6.40)$$

$$= \log P(D | \mathbf{P}, M). \quad (6.41)$$

6.3.5 Joint Model

We combine photometric, stereo, and 3D data into a joint model by assuming conditional independence given M ,

$$P(X_{\text{mono}}, X_{\text{stereo}}, X_{3D} | M) = P(X_{\text{mono}} | M)P(X_{\text{stereo}} | M)P(X_{3D} | M). \quad (6.42)$$

Taking logarithms leads to summation over payoffs,

$$\log P(X_{\text{mono}}, X_{\text{stereo}}, X_{3D} | M) = \Pi_{\text{joint}}(M) \quad (6.43)$$

where

$$\pi_{\text{joint}}(\mathbf{p}) = \pi_{\text{mono}}(\mathbf{p}) + \pi_{\text{stereo}}(\mathbf{p}) + \pi_{3D}(\mathbf{p}). \quad (6.44)$$

Finally, the log posterior is

$$\begin{aligned} \log P(M | X_{\text{mono}}, X_{\text{stereo}}, X_{3D}) &\propto \log P(X_{\text{mono}} | M) + \log P(X_{\text{stereo}} | M) \\ &\quad + \log P(X_{3D} | M) + \log P(M) \\ &\propto \Pi_{\text{joint}}(M) - \Gamma(M). \end{aligned} \quad (6.45)$$

The relation above is a proportionality rather than an equality because we have omitted the evidence $P(X)$. For the purpose of maximisation this term is irrelevant since it is independent of the quantity that we are maximising. We can now cast maximum-likelihood and maximum-a posteriori inference in the form (6.4),

$$\hat{M}_{\text{ML}} = \underset{M \in \mathcal{M}}{\operatorname{argmax}} \Pi_{\text{joint}}(M) \quad (6.46)$$

$$\hat{M}_{\text{MAP}} = \underset{M \in \mathcal{M}}{\operatorname{argmax}} \Pi_{\text{joint}}(M) - \Gamma(M). \quad (6.47)$$

This completes the task of writing inference for our model in payoff form. In particular, the above derivations show that for any set of observations (photometric, stereo, depth, or any combination thereof), there is a particular payoff function Π and penalty function Γ such that solving (6.6) is equivalent to solving MAP (or ML) inference under our model. In practice we only need to evaluate Π at integer pixel coordinates, so we represent it as a 2D array. The remainder of this chapter focusses on solving optimisation problems of the form (6.6).

6.4 MAP Inference Using Dynamic Programming

In this section we solve the optimisation problem (6.6). That is, given some π and γ , we wish to identify the maximiser M^* of

$$f(M) = \sum_{j=1}^W \pi(j, s_j, o_j) - \sum_{i=0}^{K-1} \gamma(i; M), \quad (6.48)$$

where

$$M = (x_1, y_1, o_1, x_2, y_2, o_2, \dots, x_{n-1}, y_{n-1}, o_{n-1}, x_n) \quad (6.49)$$

is an indoor Manhattan scene in vertex representation.

We assume that the scene rotation R_w as well as the Manhattan homology H_a have been recovered as discussed in previous chapters. We further assume that all images are rectified as in (4.15). The algorithms presented in this section are valid without the rectification step, but

assuming rectification considerably simplifies their presentation. Our solution uses dynamic programming to efficiently solve the maximisation (6.6). We develop the algorithm conceptually before formalising it.

In the optimisation (6.6), we wish to constrain solutions to valid indoor Manhattan scenes. In terms of individual pixel labels, such a constraint introduces complicated dependencies between large groups of pixels, since assigning a particular label to any one pixel restricts which labels can be assigned to other pixels in the same column. We therefore cast the optimisation directly in terms of the vertex representation introduced in Chapter 4.

We have already seen that every indoor Manhattan scene can be represented as a left-to-right sequence of wall segments, and every image column intersects exactly one wall segment. A direct result is that the placement of each wall is conditionally independent of the other walls given its left and right neighbours. For example, Figure 6.7 shows a partial scene as well as several wall segments that could be appended to it. Some of the candidates are feasible (green dashes) and some are not (red dashes); however, note that once the wall segment from c_3 to c_4 is fixed, the choices for walls following c_4 are independent of choices made for wall segments prior to c_3 . Later we will formalise and prove this statement; for now we proceed conceptually.

We are led to a decomposition of the problem into a series of sub-problems of the form “find the optimal *partial* model spanning columns $0 \dots x$ ” for various x . To solve this we enumerate over all possible walls that terminate at x , then for each we recursively solve sub-problems corresponding to the left extent of that wall. With each recursion, x moves to the left, and this process terminates at the boundary of the image. As in all dynamic programming, the solution to each sub-problem is cached to avoid redundant computation, and a simple counting argument shows that the algorithm has polynomial complexity.

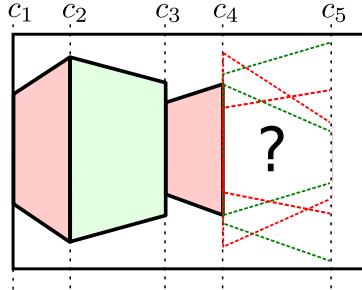


Figure 6.7: A partial scene terminating at column c_4 with several feasible (green dashed) and infeasible (red dashed) wall segments.

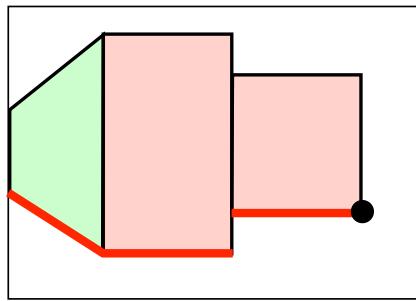


Figure 6.8: A partial scene and the corresponding partial seam. The black dot shows the terminating state of the scene.

6.4.1 Preliminaries

Partial Scene

We begin by introducing the notion of partial scenes, which span the portion of the image from its left edge to some image column x , $0 \leq x \leq W$. Partial scenes are related to the state space in the dynamic programming algorithm below. An example of a partial scene is shown in Figure 6.7 and Figure 6.8.

Definition 1. *The length of a scene M in vertex representation (4.55) is the number of distinct walls in M and is denoted $|M|$.*

Definition 2. *A partial scene M is a scene with $x_{|M|} < W$. The corresponding seam representation is*

$$S = \{(s_j, o_j)\}_{j=0}^{|M|} \quad (6.50)$$

where the values s_j, o_j are defined as for ordinary seams by equation (4.57).

We now define two operations for manipulating scenes. Walls are represented by the 4-tuple (x_a, y_a, o, x_b) as defined by equation (4.50).

Definition 3. *The termination point of a wall W is the triple (x_b, y_b, o) where (x_b, y_b) is the location of the lower-right corner of W as defined by equation (4.52). We say that a scene M terminates at (x_b, y_b, o) if its right-most wall terminates at (x_b, y_b, o) . We will also say that a scene terminates at column x_b if it terminates at (x_b, y_b, o) for some y_b, o . This is illustrated in Figure 6.8.*

Definition 4. *The truncation of the partial scene M is scene obtained by removing the last wall from M ,*

$$M_{-1} = (x_1, y_1, o_1, \dots, x_{|M|-2}, y_{|M|-2}, o_{|M|-2}, x_{|M|-1}). \quad (6.51)$$

Definition 5. *The concatenation of the scene M with the wall $W = (x_a, y, o, x_b)$ is defined if and only if M terminates at column x_a and equals*

$$M \uparrow\downarrow W = (x_1, y_1, o_1, \dots, x_{|M|-1}, y_{|M|-1}, o_{|M|-1}, x_a, y, o, x_b) \quad (6.52)$$

Value Function

We now introduce a value function for walls and scenes. Our definitions broadly mirror the definition for full scenes in equation (6.1), the only difference being that we evaluate the summations over a subset of the image. This is a subtle but important distinction since there is no simple probabilistic interpretation for partial scenes. This poses no difficulty to the probabilistic consistency of our algorithm since in this setting we are simply solving the optimisation problem (6.6). If π has been configured to represent the log-likelihood of some probabilistic model then the solution we find will correspond to MAP inference, but we are not concerned with the specifics of the probabilistic model in this section.

Definition 6. *The value obtained by a partial scene M that terminates at column x_b under the payoff function Π is*

$$f(M) = \sum_{j=0}^{x_b} \pi(j, s_j, o_j) - \sum_{i=0}^{|M|} \gamma(i; M) \quad (6.53)$$

Definition 7. *The payoff for a wall $W = (x_a, y, o, x_b)$ under the payoff function Π is*

$$\Pi(W) = \sum_{j=x_a}^{x_b-1} \pi(j, s_j, o_j). \quad (6.54)$$

We note the following additivity property of the value function.

Lemma 1. *Let M be a scene and W be a wall and suppose $M ++ W$ is well-defined. Then the value obtained by $M ++ W$ is*

$$f(M ++ W) = f(M) + \Pi(W) - \gamma(M, W) \quad (6.55)$$

where $\gamma(M, W)$ is the penalty for the corner added to the scene as a result of appending W .

Proof. Follows directly from Definitions 6 and 7. \square

Feasibility

Not all scenes permitted by the vertex representation are physically realisable. Lee *et al.* [13] showed that physical realisability can be deduced from simple tests on the ordering of walls and the relative positions of vanishing points. Although we do not wish to repeat their work here, several decomposability properties of the physical realisability tests are central to the validity of the dynamic programming algorithm to be presented, so we re-state them here

Definition 8. *The i^{th} corner of the scene M is feasible if and only if it passes the tests described in [13], which is a function of the values,*

$$y_b, o_i, x_{i+1}, y_{i+1}, o_{i+1} \quad (6.56)$$

where the i^{th} wall in M terminates at (x_{i+1}, y_b, o_i) .

Lemma 2. *M is feasible if and only if all corners in M are feasible.*

Proof. See [13] \square

Lemma 3. *Let M be a feasible scene. Then its truncation M_{-1} is feasible.*

Proof. The truncation M_{-1} consists of a subset of the corners in M , all of which are feasible by assumption. \square

Lemma 4. *Let $W = (x_b, y_b, o_b, x_c)$ be a wall and let M and Q be feasible scenes terminating at (x_b, y_a, o_a) . Then $M + W$ is feasible if and only if $Q + W$ is feasible.*

Proof. Suppose $M + W$ is feasible. Then we need to show that each corner in $Q + W$ is feasible. By assumption the first $|Q|$ corners are feasible. According to Definition 8, the feasibility of the last corner is a function of the values

$$y_a, o_a, x_b, y_b, o_b, \quad (6.57)$$

But these values are identical to the last corner in $M + W$, which is feasible by assumption, so $Q + W$ is feasible. The other direction of implication is obtained by a symmetric argument. \square

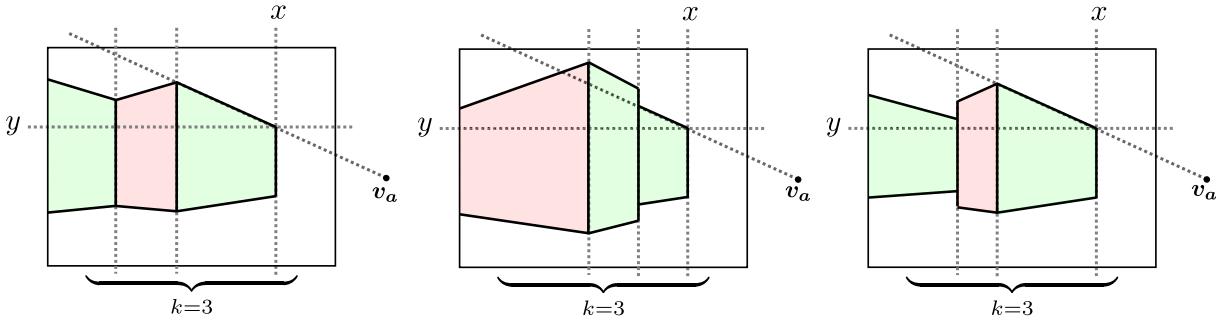
6.4.2 Optimal Substructure

We now present the sub-problems and corresponding recurrence relations comprising our first algorithmic solution to (6.6). Proving that our algorithm solves the optimisation problem of interest hinges on an optimal substructure property of indoor Manhattan scenes. Roughly, this property states that if a particular scene is optimal for a particular sub-problem, then its “pre-decessors” (truncations) are each optimal for their corresponding sub-problems. We formalise this below.

Definition 9. *A scene M satisfies the sub-problem $f_{in}(x, y, a)$ if M is feasible and M terminates at (x, y, a) . A scene M solves the sub-problem $f_{in}(x, y, a)$ if it satisfies $f_{in}(x, y, a)$ and there is no other scene satisfying $f_{in}(x, y, a)$ that obtains a greater value.*

Figure 6.9 shows three examples of different scenes satisfying a particular sub-problem.

Lemma 5. *There is a solution to $f_{in}(x, y, o)$ for all $1 \leq x \leq W, 1 \leq y \leq H, o \in \{1, 2\}$.*

Figure 6.9: Three models satisfying the sub–problem $f_{in}(x, y, a)$.

Proof. Simply construct a scene M with a sequence of alternating concave/convex corners in the interval $[1, x]$. Non–occluding corners are always feasible [13], so M satisfies $f_{in}(x, y, o)$. There are finitely many scenes and at least one satisfies $f_{in}(x, y, o)$; therefore $f_{in}(x, y, o)$ has a solution. \square

We now turn to the optimal substructure theorem, which is the central theorem of this chapter. The quantities in the proof below are depicted in Figure 6.10.

Theorem 1. *Let M be a scene terminating at (x, y, o) . Let M' be the 1–truncation of M and let (x', y', o') be the terminating state of M' . If M solves $f_{in}(x, y, o)$ then M' solves $f_{in}(x', y', o')$.*

Proof. First note that M' satisfies $f_{in}(x', y', o')$ since it terminates at (x', y', o') and is feasible by Lemma 3.

We show that M' solves $f_{in}(x', y', o')$ by appeal to *reductio*. If M' does not solve $f_{in}(x', y', o')$ then there exists a scene Q' satisfying $f_{in}(x', y', o')$ such that

$$f(Q') > f(M') . \quad (6.58)$$

Let W be the right–most wall in M and let $Q = Q' + W$. We have that Q satisfies $f_{in}(x, y, o)$ because:

1. Q terminates at (x, y, o) since its right–most wall is W , which terminates at (x, y, o) by assumption.
2. Q is feasible by Lemma 4.

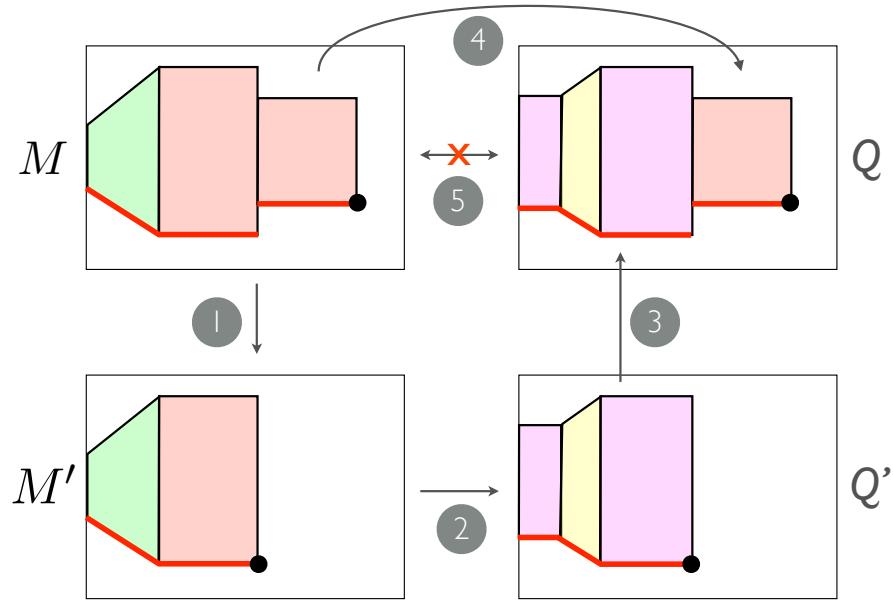


Figure 6.10: A graphical sketch of the proof of Theorem 1. The proof proceeds counter-clockwise from top left. First we let M be an optimal scene for some sub–problem, then (1) define M' to be its truncation. We then (2) postulate some Q' with value greater than M' , and then from this we (3,4) construct Q . The contradiction follows by (5) comparing M and Q .

Next we use Lemma 1 to expand the values obtained by M and Q ,

$$\begin{aligned} f(M) &= f(M') + \Pi(W) - \gamma(M', W) \\ f(Q) &= f(Q') + \Pi(W) - \gamma(Q'; W) \end{aligned} \tag{6.59}$$

But by an analogous argument to the proof of Lemma 4, the corner produced by appending W to M' is the same as that produced by appending W to Q' since M' and Q' terminate at the same state. Therefore

$$\gamma(M', W) = \gamma(Q', W) . \tag{6.60}$$

Combining (6.58), (6.59), and (6.60), we have

$$f(Q) > f(M) , \tag{6.61}$$

but this contradicts the assumption of M as a solution to $f_{in}(x, y, o)$. \square

6.4.3 First Algorithm

Theorem 1 is the main result in the construction of the dynamic programming solution to (6.6). Our first recurrence relation is provided by corollary 1 below, but first we need to define the feasible set,

Definition 10. A state s is a tuple (x, y, o) . The state space \mathcal{S} is the set of all states,

$$\mathcal{S} = [1, W] \times [1, H] \times \{1, 2\}. \quad (6.62)$$

The feasible set $\mathcal{F}(s) \subset \mathcal{S}$ for s is the set of states s' such that the scene

$$M = M' ++ W \quad (6.63)$$

is feasible, where M' is any feasible model terminating at s' and $W = (x', y', a, x)$.

Intuitively one can think of the feasible set as the set of states “reachable in one step” from s .

Our first recurrence relation is a corollary to theorem 1 as follows.

Corollary 1. Let $s = (x, y, o) \in \mathcal{S}$ be a state with $x > 1$. Then

$$f_{in}(s) = \max_{s' \in \mathcal{F}(s)} \left(f_{in}(s') + \Pi(W) - \gamma(s', W) \right), \quad (6.64)$$

where

$$W = (x', y', o, x). \quad (6.65)$$

Further,

$$f_{in}(1, y, o) = 0 \quad \forall y, o. \quad (6.66)$$

Proof. Let M be a solution to $f_{in}(s)$. We first show that the left side of (6.64) is greater than or equal to the right side. We then show that the inequality is an equality.

The inequality is established as follows. Let M' be a solution to $f_{in}(s')$ for $s' \in \mathcal{F}(s)$. Let W be

as defined in (6.65). We have that

1. $M' \uplus W$ is feasible by the definition of the feasible set; and
2. $M' \uplus W$ terminates at s by the definition of W .

So $M' \uplus W$ satisfies $f_{in}(s)$. By the assumption of M as the solution to $f_{in}(s)$ we therefore have

$$f(M) \geq f(M' \uplus W) \quad (6.67)$$

$$\geq f(M') + \Pi(W) - \gamma(s', W) \quad (6.68)$$

$$f_{in}(s) \geq f_{in}(s') + \Pi(W) - \gamma(s', W), \quad (6.69)$$

thus giving the desired inequality.

The equality is obtained by noting that theorem 1 guarantees that there is at least one state $s^* \in \mathcal{F}(s)$ such that a solution to $f_{in}(s^*)$ concatenated with W is a solution to $f_{in}(s)$. The boundary condition (6.66) follows from the fact that a scene that terminates at column 1 does not span any part of the image. \square

Finally, the value of the solution to (6.6) is

$$f(M^*) = \max_{y,o} f_{in}(W, y, o). \quad (6.70)$$

We are now ready to describe our first solution to optimisation problems in payoff form. We compute $f(M^*)$ by recursively evaluating f_{in} according to (6.64) until we reach the boundary condition (6.66). To avoid redundant computation we cache the result of each evaluation together with the state s' corresponding to the maximising term in (6.64). Finally, the desired model M^* is reconstructed by back-tracking through the evaluation graph once all evaluations are complete. This is formalised in Algorithm 6.1.

Algorithm 6.1 Solution to (6.6) [version 1]

Require: Π is a payoff function
Require: γ is a penalty function
Ensure: M^* is the solution to (6.6)

```

cache =  $\emptyset$ 
source =  $\emptyset$ 
 $f^* = -\infty$ 
for  $y = 1$  to  $H$  do
  for  $o \in \{1, 2\}$  do
     $f_{cur} \leftarrow \text{Solve}(W, y, o)$ 
    if  $f_{cur} > f^*$  then
       $f^* \leftarrow f_{cur}$ 
       $s^* \leftarrow \{W, y, o\}$ 
    end if
  end for
end for
 $M^* \leftarrow (W)$ 
while  $s^* \neq \emptyset$  do
   $M^* \leftarrow (s_x^*, s_y^*, s_o^*) \uplus M^*$ 
   $s^* \leftarrow \text{source}[s^*]$ 
end while
```

Subprocedure Solve:

Require: $s = (x, y, o) \in \mathcal{S}$
Ensure: $\text{cache}[s] = f_{in}(s)$

```

if  $s \notin \text{cache}$  then
  if  $x = 0$  then
     $\text{cache}[s] \leftarrow 0$ 
     $\text{source}[s] \leftarrow \emptyset$ 
  else
    for all  $s' \in \mathcal{F}(s)$  do
       $W \leftarrow (s'_x, s_y, s_o, s_y)$ 
       $f_{cur} \leftarrow \text{Solve}(s') + \Pi(W) - \gamma(s', W)$ 
      if  $f_{cur} > \text{cache}[s]$  then
         $\text{cache}[s] \leftarrow f_{cur}$ 
         $\text{source}[s] \leftarrow s'$ 
      end if
    end for
  end if
end if
return  $\text{cache}[s]$ 
```

Algorithmic Complexity

Due to the caching scheme, (6.64) is evaluated at most once for each unique state. There are $2WH$ states and the complexity of each evaluation is $O(W^2H)$, since the minimisation in (6.64) is over $O(WH)$ terms and computing each marginal payoff $\Pi(W)$ requires $O(W)$ additions. The overall complexity of the algorithm above is therefore $O(W^3H^2K) = O(L^5K)$ where $L = \max(W, H)$.

6.4.4 First Refinement: Auxiliary Sub-problems

The basic algorithm described thus far involves minimising over all pixels to the left of s for each state s . In the previous section we enforced feasibility by explicitly testing each s' and omitting any that lead to an infeasible model. In this section we show that by introducing auxiliary sub-problems we can deal with feasibility constraints while avoiding the minimisation over $O(WH)$ items at each evaluation. We introduce three new sub-problems as follows.

Definition 11. Let M be a scene terminating at (x_b, y_b, o_b) and let $s = (x, y, o)$ be a state. Then

- M satisfies $f_{up}(s)$ iff M is feasible and $x_b = x$ and $o_b = o$ and $y_b \leq y$;
- M satisfies $f_{down}(s)$ iff M is feasible and $x_b = x$ and $o_b = o$ and $y_b \geq y$;
- M satisfies $f_{out}(s)$ iff M is feasible and $x_b = x$ and $M \uparrow (x, y, o, x+1)$ is feasible.

Intuitively, the scenes satisfying f_{up} and f_{down} are those that terminate directly above (x, y) and directly below (x, y) , respectively. The scenes that satisfy f_{out} are those to which a wall with lower-left corner (x, y) and orientation o could feasibly be added. The purpose of introducing auxiliary sub-problems is to permit a series of more efficient recurrence relations, which we introduce below.

Corollary 2. Let $s = (x, y, o) \in \mathcal{S}$ be a state. Then

$$f_{up}(x, y, o) = \begin{cases} \max(f_{in}(x, y, a), f_{up}(x, y - 1, o)), & \text{if } y > 1 \\ f_{in}(x, y, a), & \text{if } y = 1 \end{cases} \quad (6.71)$$

$$f_{down}(x, y, o) = \begin{cases} \max(f_{in}(x, y, o), f_{down}(x, y + 1, o)), & \text{if } y < H \\ f_{in}(x, y, o), & \text{if } y = H \end{cases} \quad (6.72)$$

Proof. We prove (6.71) only; the proof of (6.72) is identical.

Let M be a solution to $f_{up}(x, y, o)$ and let its terminating state be (x_b, y_b, o_b) . Consider first the case that $y = 1$. By the definition of f_{up} it must be that M terminates at (x, y, a) , in which case the conditions on M now exactly match those for f_{in} given in definition 9.

Now suppose $y > 1$. Then either $y_b = y$ or $y_b \leq y - 1$, with the former corresponding again to the sub-problem $f_{in}(x, y, a)$ and the latter to the sub-problem $f_{up}(x, y - 1, a)$. Therefore, we may simply maximise over those two cases. \square

Corollary 3. Let $s = (x, y, o) \in \mathcal{S}$ be a state. Then

$$\begin{aligned} f_{out}(x, y, o) = \max_{o' \in \{l, r\}} & \max \left(f_{up}(x, y - 1, o') - \gamma(x, o', o, -1), \right. \\ & f_{in}(x, y, o') - \gamma(x, o', o, 0), \\ & \left. f_{down}(x, y + 1, o') - \gamma(x, o', o, +1), \right) \end{aligned} \quad (6.73)$$

Proof. Omitted. The main result is Theorem 1; this proof closely mirrors that of Corollary 2 above. \square

Finally we re-write (6.64) in terms of the new sub-problems as follows.

Corollary 4. Let $s = (x, y, o) \in \mathcal{S}$ be a state with $x > 1$. Let $Y_s(x')$ be the intersection of column x' with the line through v_o and (x, y) (see Figure 6.13). Then

$$f_{in}(x, y, o) = \max_{x' < x} \left(f_{out}(x', Y_s(x'), o) + \Pi(W) \right), \quad (6.74)$$

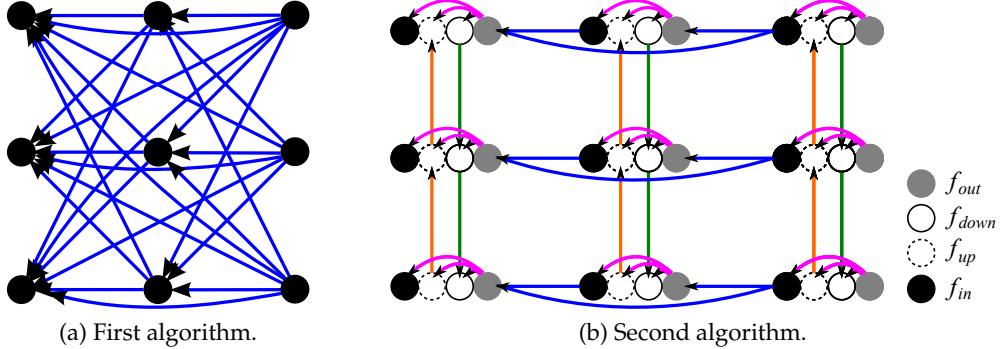


Figure 6.11: The evaluation graph for a 3×3 image before (left) and after (right) introducing auxiliary sub-problems.

where $s' = (x, Y_s(x'), o)$ and $W = (x', Y_s(x'), o, x)$ is a wall.

Proof. Omitted. The main result is Theorem 1; this proof closely mirrors that of Corollary 2 above. \square

The dependencies between the sub-problems are illustrated as an evaluation graph in Figure 6.11.

Algorithmic Complexity

By introducing auxiliary sub-problems we have increased the total number of sub-problems by a factor of 4. However, the sub-problems f_{up} , f_{down} , and f_{out} have $O(1)$ complexity and the maximisation (6.74) is now over $O(W)$ terms (whereas (6.64) consists of $O(WH)$ terms). The complexity of the algorithm in this section is therefore $O(L^3)$.

6.4.5 Second Refinement: From $O(L^3)$ to $O(L^2)$

Evaluating (6.74) remains an $O(W)$ operation due to the minimisation over x' . In this section we reduce this to $O(1)$.

Consider recurrence relation (6.74). The maximisation is over columns $x-1, x-2, \dots, 1$. Suppose we pick a some z and divide the columns into two sets, $Z_+ = \{x-1, x-2, \dots, z\}$ and

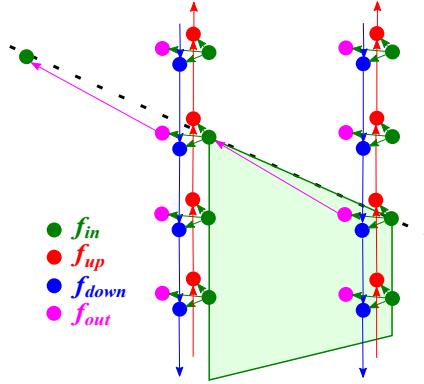


Figure 6.12: A graph in which each node represents a sub–problem and each edge is a dependence relation. Two columns are expanded; other column are omitted for brevity. The green quad is a wall corresponding to a particular pair of nodes in the graph.

$Z_- = \{z - 1, z - 2, \dots, 1\}$. Then expanding the maximisation of (6.74) we have

$$f_{in}(x, y, o) = \max \left\{ \begin{aligned} & \max_{s' \in Z_+} (f_{out}(s', o) + \Pi(W) - \gamma(s', W)), \\ & \max_{s' \in Z_-} (f_{out}(s', o) + \Pi(W) - \gamma(s', W)) \end{aligned} \right\}. \quad (6.75)$$

Now compare the maximisation over Z_- in (6.75) with the sub–problem $f_{in}(z, \tilde{y}, o)$, where \tilde{y} equals $Y_s(z)$ rounded to the nearest integer ($Y_s(z)$ was defined in Corollary 4). Both maximizations are over columns $z - 1 \dots 1$; the only difference is that in (6.75) the y –coordinates at each column are $Y_s(x)$ whereas in $f_{in}(z, \tilde{y}, o)$ they are $Y_t(x)$ where $t = (z, \tilde{y}, o)$. If we accept the latter as an approximation to the former then we could set $z = x - 1$ and replace the $O(W)$ maximisation over Z_- with a single call to $f_{in}(z, \tilde{y}, o)$, giving an $O(1)$ expression for f_{in} . Unfortunately this is a very poor approximation because the quantities $Y_s(x)$ and $Y_t(x)$ may differ substantially, since a small vertical offset at one point may correspond to a large offset at another image column. Furthermore, repeatedly making this approximation in recursive evaluations of f_{in} compounds the errors. Instead, we proceed as follows.

Consider the sub–problem $f_{in}(x, y, o)$ as formulated in (6.74). Evaluating f_{in} is like walking along each column $x - 1, x - 2, \dots, 1$ and considering two possibilities at each step: insert a corner or continue walking. The former corresponds to evaluating $f_{out}(x', y', o)$; the latter to $f_{in}(x', y', o)$. But y' is computed by intersecting two lines, so in general is not an integer. While it is sufficient to round y' to the nearest integer $\tilde{y} = \lfloor y' + 0.5 \rfloor$ when evaluating f_{out} , doing the

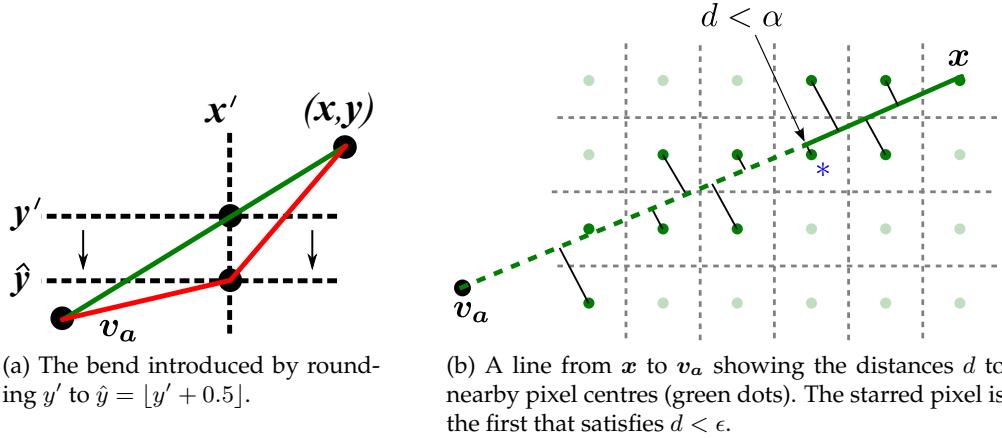


Figure 6.13: Two quantities related to the “line–jump” approximation described in Section 6.4.5. same for f_{in} would produce a bend in the wall as shown in Figure 6.13. In (6.74) we avoided this by evaluating f_{out} for all $x' < x$, but this is unnecessarily wasteful.

We now introduce a parameter $\epsilon > 0$ and allow $t = (z, \tilde{y}, o)$ to replace $s = (x, y, o)$ whenever

$$|y - \tilde{y}| < \epsilon. \quad (6.76)$$

To evaluate $f_{in}(x, y, o)$ we find the largest integer $z < x$ satisfying the above. We do this by enumerating each z from $x - 1$ to 0. We cannot do better with a more efficient search strategy since we must evaluate f_{out} for each column between the z that we do eventually identify and $x - 1$. Finally, the recurrence relation for f_{in} is

$$f_{in}(x, y, o) = \max \left\{ \begin{aligned} & \max_{x' \in [z, x-1]} \left(f_{out}(x', Y_s(x'), o) + \Pi(W) - \gamma(s', W) \right), \\ & f_{in}(z, Y_s(z), o) + \Pi(W) - \gamma(s', W) \end{aligned} \right\}. \quad (6.77)$$

where

$$z = \max \left\{ x' \in \mathbb{Z} : x' < x \wedge |Y_s(x') - \lfloor Y_s(x') + 0.5 \rfloor| \leq \epsilon \right\} \quad (6.78)$$

A bound on number of terms in (6.77) is provided by the following lemma.

Lemma 6. *For all $\epsilon > 0$ there exists $R > 0$ such that for any $(x, y) \in \mathbb{Z}^2$ and any $(v_x, v_y) \in \mathbb{R}^2$ there*

is some $(p, q) \in \mathbb{Z}^2$ with $x - R \leq p < x$ such that

$$\frac{|(v_y - y)(p - x) - (v_x - x)(q - y)|}{\sqrt{(v_x - x)^2 + (v_y - y)^2}} \leq \epsilon \quad (6.79)$$

Further, the above holds for

$$R = \epsilon\sqrt{2} + \frac{1}{\epsilon} \quad (6.80)$$

Proof. First we substitute $r = p - x, s = q - y, a = \frac{v_y - y}{\sqrt{(v_y - y)^2 + (v_y - y)^2}}$, and $b = \frac{v_x - x}{\sqrt{(v_x - x)^2 + (v_y - y)^2}}$,

$$|ar - bs| < \epsilon \quad (6.81)$$

Next, noting that $[a, b]$ is a unit vector and without loss of generality letting $a \geq b$ we have

$$\left| \frac{b}{a}s - r \right| \leq \frac{\epsilon}{a} \leq \epsilon\sqrt{2}. \quad (6.82)$$

Dirichlet's theorem [42] guarantees the existence of a pair (r, s) with $1 \leq s \leq \frac{1}{\epsilon}$. Rearranging the above we get

$$|r| \leq \epsilon\sqrt{2} + \left| \frac{b}{a}s \right| \quad (6.83)$$

$$\leq \epsilon\sqrt{2} + \frac{1}{\epsilon}. \quad (6.84)$$

Without loss of generality we assume $r < 0$ (since we may multiply both r and s by -1), yielding

$$-\epsilon\sqrt{2} - \frac{1}{\epsilon} \leq r < 0 \quad (6.85)$$

$$-\epsilon\sqrt{2} - \frac{1}{\epsilon} \leq p - x < 0 \quad (6.86)$$

$$x - R \leq p < x. \quad (6.87)$$

where $R = \epsilon\sqrt{2} + \frac{1}{\epsilon}$. □

Algorithmic Complexity

For fixed ϵ , the number of terms in the maximisation in (6.77) is bounded by a constant independent of the problem size. All sub-problems are thus $O(1)$ operations and the overall complexity of the algorithm is given by the total number of unique sub-problems, which is $O(L^2)$.

6.5 Results

Our data-set consists of 18 manually annotated video sequences of 8 unique indoor locations averaging 59 seconds in duration. We sample frames at one second intervals and divide frames into consecutive blocks of 3 (one base frame and two auxiliary frames). Our evaluation set consists of 150 such triplets generated from 8 different sequences.

To acquire ground truth data we reconstructed camera trajectories using structure-from-motion software, then manually specify the ground truth floor-plan. Recall that we seek to recover the *boundaries* of the environment, whether or not they are visible at every pixel. When our algorithm ignores clutter within a room, we consider that a *success*.

The monocular features ϕ_i consist of 3 RGB channels, 3 HSV channels, 24 Gabor filters (4 scales, 6 orientations), and 3 binary line sweep features [13]. For stereo we use patches of size 5×5 .

For these experiments, we fixed the various hyper-parameters using a simple bootstrapping algorithm evaluated on a separate training set. This approach is discussed in a published paper [23]; we omit it here because it is superseded by the more principled and effective learning algorithm described in the following chapter, and because in later experiments we found that it did not substantially improve upon manually chosen values. Qualitatively, the discriminative power of our model comes from the strong geometric feasibility constraints, the global inference algorithm, and the Bayesian sensor models.

We compute two error metrics: the labelling accuracy, which is the proportion of all pixels that

were labeled with the correct surface orientation, and the mean relative depth error, computed per-pixel with respect to ground truth. While the latter better captures similarity to the ground truth, one of the systems we compared with did not have a direct 3D interpretation, so we compared on labelling accuracy.

To the best of our knowledge, there is no previously published work other than our own on precisely this problem (indoor–Manhattan reasoning from multiple views), so we compare with two alternative systems, though neither comparison is ideal. Our first comparison is with the approach of Brostow *et al.* [6], who performed semantic segmentation by training a per-pixel classifier on structure–from–motion cues. Our implementation of their system uses exactly the features they describe, with classes corresponding to the three Manhattan orientations. While they trained a randomised forest, we trained a multi-class SVM because a reliable SVM library was more readily available to us. Given the margin between our results it is unlikely that a different classifier would significantly change the outcome. The second comparison is with the monocular approach of Lee *et al.* [13]. One would of course expect a multiple view approach to outperform a monocular approach, but as one of the very few previous approaches to have explicitly leveraged the indoor Manhattan assumption we feel this comparison is important to demonstrate the benefit of a Bayesian framework and integration of stereo and 3D cues.

The performance of each system is shown in Figure 6.1. Our system significantly out–performs both others. Even when restricted to monocular features, our system outperforms [6], which has access to 3D cues. This reflects the utility of global consistency and the indoor Manhattan representation in our approach. The initialisation procedure of [13] fails for 31% of our training images, so at the bottom of Figure 6.1 we show results for their system after excluding these images. Labelling accuracy increases to within 3% of our monocular–only results, though on the depth error metric a margin of 10% remains. This illustrates the effect of our training procedure, which optimises for the depth error. Figure 6.1 also shows that joint estimation is superior to using any one sensor modality alone. Anecdotally we find that using 3D cues alone often fails within large textureless regions in which the structure–from–motion system failed to track any points, whereas stereo or monocular cues alone often perform better in such regions

Algorithm	Mean depth error (%)	Labelling accuracy (%)
Our approach (full)	14.5	75.5
Stereo only	17.4	69.5
3D only	15.2	71.1
Monocular only	24.8	69.2
Brostow <i>et al.</i> [6]		40.6
Lee <i>et al.</i> [13]	79.8	45.5
excluding failures ²	34.1	66.2

Table 6.1: Performance on our data-set. Labelling accuracy is the percentage of correctly labeled pixels over the data-set, and depth error is the mean relative depth error.

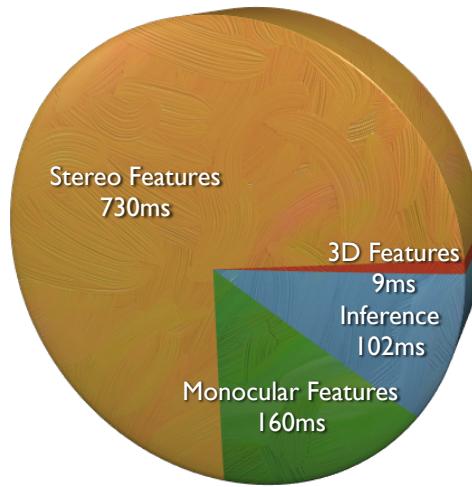


Figure 6.14: Processing time for feature computation and inference in our system, averaged over all evaluation instances. The mean total processing time was 997ms.

but can lack precision at corners and boundaries.

Figure 6.14 shows timing results for our system. For each triplet of frames, our system requires on average less than one second to compute features for all three frames and less than 100 milliseconds to perform optimisation.

²This row excludes cases for which [13] was unable to find overlapping lines during initialisation.

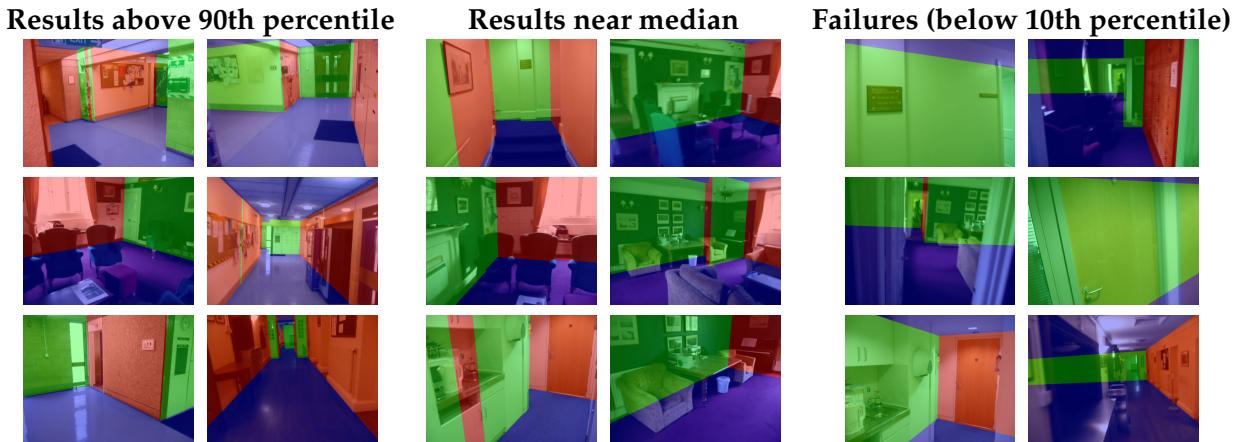


Figure 6.15: Scenes output from our system. The left column shows results above the 90th percentile of performance (relative depth error), the middle column shows results near median performance, and the right column shows failure cases.

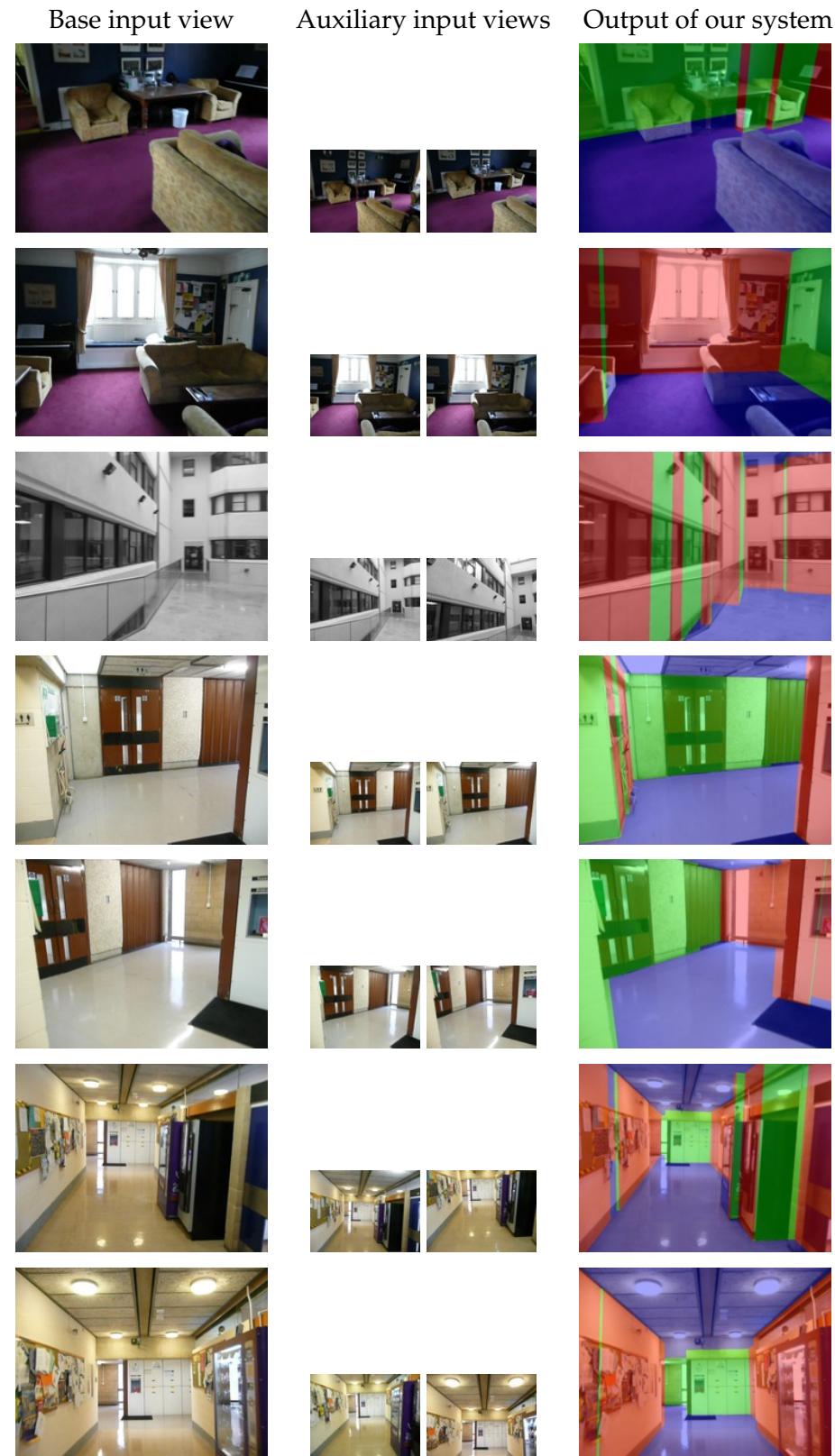
Table 6.2: Here we show further examples of scenes inferred by our system. In the table below, the left panel shows the base input view, the middle panel shows the two auxiliary views used for photo-consistency calculations, and the right panel shows the MAP scene M inferred by our system.



Continued on next page



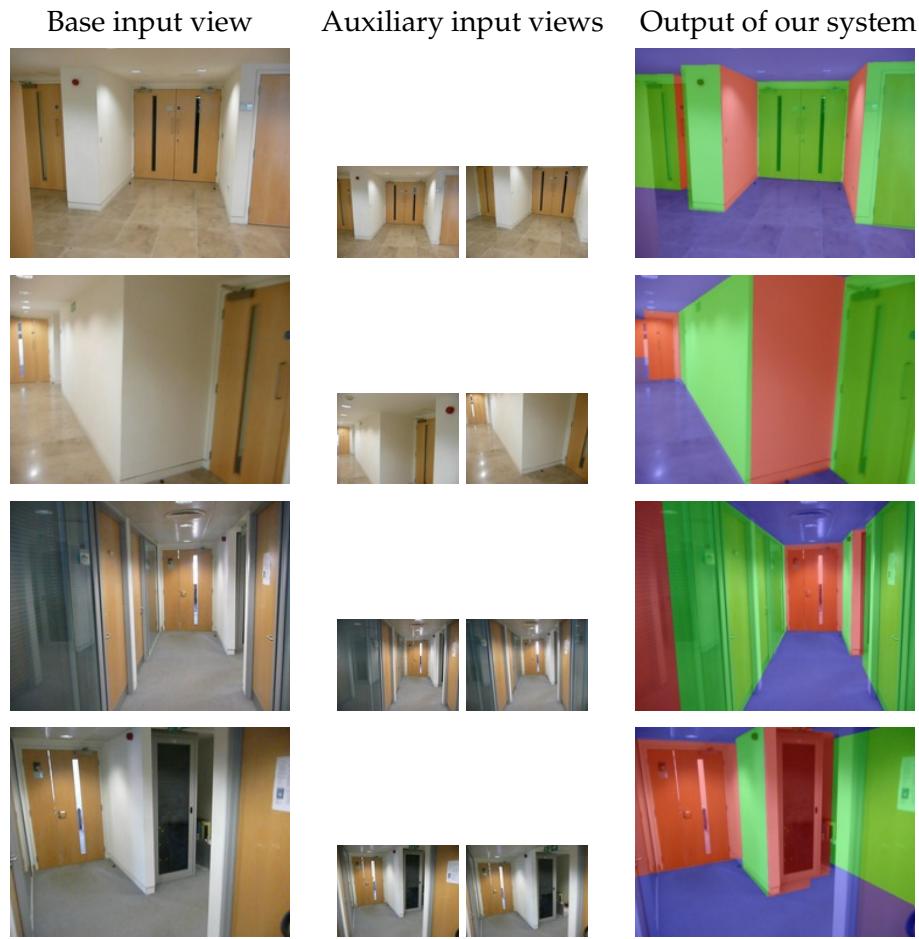
Continued on next page



Continued on next page

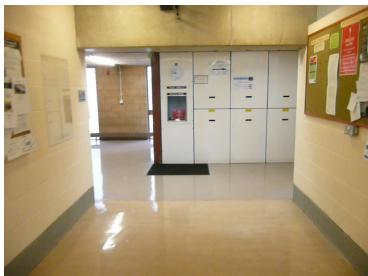


Continued on next page



6.5.1 Payoff Matrices

In this section we provide some visualisations of the payoff matrices discussed above in order to give extra insight into the structure of each sensor modality.



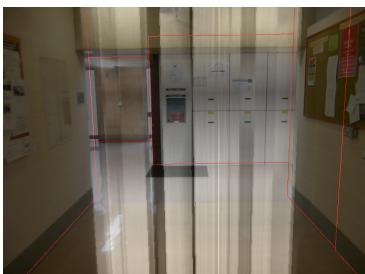
The raw image provided as input to our system.



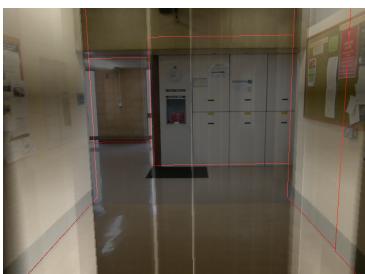
The ground truth segmentation for this image. Horizontal surfaces are shaded blue. Vertical surfaces are shaded red and green. We will refer to these as the “red” and “green” orientations.



The MAP indoor manhattan model w output by our system for this input.



Payoffs π_{mono} derived from monocular image features, for the “green” orientation. Pixels of higher intensity correspond to larger values in the payoff matrix. The MAP model is shown in wireframe using red lines. Intuitively, the optimisation over models can be thought of as finding the minimal cost path through the payoff matrix, where higher intensity pixels correspond to lower costs. This is only a rough picture, however; the real optimisation situation is more complex since models are penalised for each additional corner.



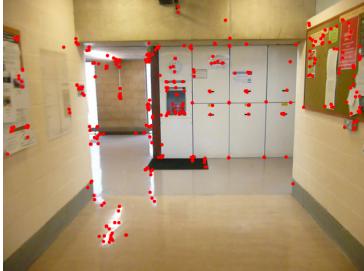
As above, for the “red” surface orientation.



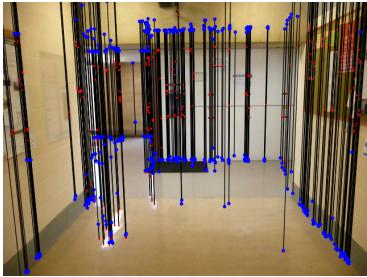
Auxiliary images used for stereo photo-consistency. In our experiments we used two image auxiliary images for each base image, which were sampled one second before and one second after the base image in the video sequence.



Payoffs π_{stereo} corresponding to the auxiliary images above. Each pixel represents the photo-consistency score for a wall segment with floor/wall (or ceiling/wall) intersection that point at that pixel. Notice the repeated “pizza slice” patterns in which one tip of the triangle is located at the floor/wall intersection.



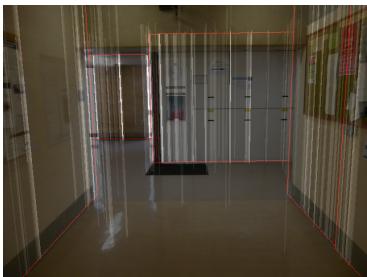
Here we show the structure-from-motion point cloud. The points are shown projected into the image, but the system has access to their 3D locations. Notice how the points are not uniformly distributed in the image.



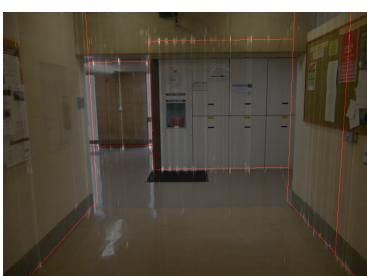
Here we show the structure-from-motion point cloud projected onto the floor and ceiling planes, which were recovered as a separate step as described in the main paper. The red dots show the original 3D point cloud and the blue dots show the projections onto the floor and ceiling.



This shows the component of the payoffs π_{3D} intended to provide a bias towards models that explain the observed 3D points. This is the component corresponding to $t = \text{ON}$. Each bright spot corresponds to the projection of a 3D point onto the floor or ceiling plane.



This shows the component of the payoffs π_{3D} intended to penalise walls that occlude observed 3D points. This corresponds to the case that $t \in \{\text{IN}, \text{OUT}\}$. Notice that for each 3D point, payoffs are assigned to walls that pass between the floor and ceiling projection of that point. Such walls are precisely those which do *not* occlude the point.



Joint payoff matrix π_{joint} .

6.6 Other Approaches

In this section we briefly discuss the conceptual differences between our algorithm and two related approaches.

6.6.1 Branch and Bound

Lee *et al.* [13] proposed a branch-and-bound solution to a similar inference problem to that considered in this chapter. Their approach identifies straight lines in the image and then searches over all possible combinations, generating from each combination a scene hypothesis. The hypotheses are evaluated using a cost function similar to (6.53). Whereas the algorithmic complexity of their algorithm is exponential in the number of lines used to generate scene hypotheses, our approach is *independent* of scene complexity; yet our hypothesis class is a strict superset of theirs.

Their approach also differs from ours in that they only allow wall boundaries to occur where lines are observed in the image. Our system can easily be extended to enforce such a constraint by including the f_{in} term in (6.71) and (6.72) only where lines are detected. However, we have found this to be unnecessary because the objective (6.4) already incorporates this information. Furthermore, our system avoids dependence on edge detection, whereas Lee *et al.* are unable to find the correct model if one or more structurally important edges are missed by the edge detector.

6.6.2 Graph Cuts

Many pixel-labelling problems can be solved using graph cuts. Kolmogorov and Zabih [40] showed that only regular functions (a subset of sub-modular functions) can be minimised via graph cuts. Interpreted as an energy, (6.4) is not regular because implicit in the optimisation is the hard constraint that labellings must form an indoor Manhattan model, which induces complicated dependencies between the pixels in each column. For example, if we were to optimise directly within a labelling representation then if some pixel p were assigned label a then $q = H_a p$ must be assigned the same label, even though the two may be arbitrarily far from one another in the image. Further, if a were either of the vertical orientations then no pixel in the same column can be assigned the opposing vertical orientation, leading to cliques of size equal to the height of the image. These constraints only capture a fraction of the full feasibility

requirements.

Even if an appropriate relaxation of this constraint yielded a regular cost function, applying graph cuts would entail using a technique such as α -expansion [40], which is both approximate and non-deterministic. In contrast, our approach is exact, deterministic, and highly efficient.

6.7 Discussion

In this section we discuss a possible generalisation of our model to relax the constraint on priors.

6.7.1 Non-memoryless Scene Priors

Recall that our prior on scenes (6.7) is

$$P(M \mid \boldsymbol{\lambda}) = \frac{1}{Z} \lambda_1^{n_1} \lambda_2^{n_2} \lambda_3^{n_3} \quad (6.88)$$

This prior is memoryless because it corresponds to the outcome of a series of independent trials. Intuitively, memorylessness means that the marginal probability of each additional wall is independent of the number of walls already added to a model. Formally, memorylessness is defined by the necessary and sufficient condition

$$P(n_i = k + m \mid n_i \geq k, \boldsymbol{\lambda}) = P(n_i = m \mid \boldsymbol{\lambda}) . \quad (6.89)$$

This property is important for the algorithm presented above because the logarithm of a memoryless prior is linear in the hyper-parameters, which we used to write the prior as a sum over independent penalties for each corner category in (6.9). On this basis we defined sub-problems independently of the number of corners and were able to incorporate penalties as additive terms in (6.73). With a non-memoryless prior this is impossible because the marginal probability of an additional corner is no longer independent of the number of corners already

added.

How well does (6.88) reflect our actual prior expectations for scenes? Certainly the provision that complex scenes are apriori less likely than simple scenes matches our intuition, but the assertion that a scene composed of just one wall is more likely than a scene with two or three walls seems less justified. Here we show how one could incorporate any prior that can be written as a function of n_1, n_2, n_3 (memoryless or not), at the cost of introducing extra dimensions to the state space and a corresponding increase in the computational complexity of the algorithm. We have not implemented this algorithm.

Let $\mathbf{n} = [n_1, n_2, n_3]$ be a vector containing three integers. First we redefine the state space to incorporate \mathbf{n} ,

$$\mathcal{S} = \{(x, y, o, \mathbf{n})\} \quad (6.90)$$

$$\mathcal{S} = [1, W] \times [1, H] \times \{1, 2\} \times \mathbb{Z}^3 \quad (6.91)$$

Next we refine the sub-problem definitions as follows.

Definition 12. A scene M satisfies $f_{in}(x, y, o, n_1, n_2, n_3)$ iff it terminates at (x, y, o) and contains exactly n_1 concave corners, n_2 convex corners, and n_3 occluding corners.

The other three sub-problems are redefined similarly.

Let ω be a function identifying the category of the corner resulting from concatenating a wall $W = (x_0, y_0, o_0, x_1)$ to a scene M terminating at (x_b, y_b, o_b) . By Definition 8, ω is functionally dependent only on the values

$$x_b, o_b, o_0, \text{Sign}(y_0 - y_b) \quad (6.92)$$

so we write

$$\omega(x, o_1, o_2, s) = \begin{cases} [1 0 0], & \text{for concave corners} \\ [0 1 0], & \text{for convex corners} \\ [0 0 1], & \text{for occluding corners} \end{cases} \quad (6.93)$$

The only recurrence relation we need to update is (6.73). The relation for f_{out} becomes

$$\begin{aligned} f_{out}(x, y, o, \mathbf{n}) = \max_{o' \in \{l, r\}} \max & \left(f_{up}(x, y - 1, o', \mathbf{n} - \omega(x, o', o, -1)), \right. \\ & f_{in}(x, y, o', \mathbf{n} - \omega(x, o', o, 0)), \\ & \left. f_{down}(x, y + 1, o', \mathbf{n} - \omega(x, o', o, +1)) \right) \end{aligned} \quad (6.94)$$

Comparison with (6.73) shows that we have replaced the penalty term with a transition from \mathbf{n} to $\mathbf{n}' = \mathbf{n} - \omega(\cdot)$. This means our sub-problems no longer incorporate penalties at all, instead we have expanded our state space to incorporate that information in a different form. In order to solve for the optimal scene we will need to enumerate many terminating states, replacing (6.70) by

$$f(M^*) = \max_{y, o, \mathbf{n}} f_{in}(W, y, o, \mathbf{n}) - P(\mathbf{n} \mid \boldsymbol{\lambda}). \quad (6.95)$$

At the cost of a substantial expansion of the state space we are now able to optimise with respect to a much larger class of priors.

6.7.2 Conclusion

We have presented a Bayesian framework for scene understanding in the context of a moving camera. Our approach draws on the indoor Manhattan assumption introduced for reasoning from single views and we have shown that techniques from monocular and stereo vision can be integrated with geometric observations in a coherent Bayesian framework. We have connected inference in our model to a class of optimisation problems that we labelled the payoff formulation, and we have presented an efficient and exact solution in the form of a dynamic programming algorithm. Our approach is able to model complex scenes, which would be intractable for previous methods that involved combinatorial searches in the space of models. Experiments show our system out-performing two similar systems for both single- and multiple-view inference.

7

Learning to Construct Indoor Manhattan Models

We develop a structured prediction approach to reconstructing 3D polygonal models from single and multiple images of a scene. Building on recent advances in single view reconstruction we adopt the indoor Manhattan hypothesis class — one of the most complicated output spaces (in terms of internal constraints) yet considered within a structured prediction framework. Our approach can learn in both the single- and multiple-view contexts. We show that the chosen hypothesis class permits optimising a variety of high-level loss functions, such as the relative depth error. Our results out-perform the state-of-the-art, including an improvement of more than 50% on one metric.¹

7.1 Introduction

Previous chapters cast scene understanding as a fixed inference problem, with each input instance considered separately. Such systems have no capacity to improve their accuracy over time based on feedback from past predictions, since there is no information carried over from

¹At the time of publishing, part of the work presented in this chapter was under review for the 2012 European Conference on Computer Vision.

one inference task to the next. This chapter considers the task of learning to recover indoor Manhattan models based on labelled training examples. We are given a series of input images together with ground truth reconstructions, and the goal is to discover a rule to recover the correct scene structure from future images.

We adopt the standard approach in machine learning of distinguishing a learning phase — a one-off computation performed ahead of time — from the inference phase — in which the results of the learning phase are used to label incoming training examples. This chapter focusses on the learning phase; we will show that under the learning scheme we propose, inference reduces to exactly the algorithm presented in the previous chapter. As a result, the approach described in this chapter is strictly complimentary to the inference algorithm described in Chapter 6.

In this chapter we emphasise a learning-based approach to recovering scene structure. Traditionally, neither the single- nor multiple-view reconstruction literature places learning at the heart of the reconstruction problem. Within single-view reconstruction, for example, few authors cast learning as a single optimisation with respect to a clearly defined loss function [33, 55, 13, 23]. On the other hand, multiple-view reconstruction techniques rarely learn from training data at all. In contrast, this chapter casts reconstruction fundamentally as a learning problem, with the goal being to learn a prediction function f mapping observed image features to indoor Manhattan reconstructions.

Using indoor Manhattan reconstructions as the hypothesis class brings a variety of attractive features such as a simple parametrisation, efficient and exact inference, and a balance between expressiveness and robustness, as has been discussed in preceding chapters. In this chapter we show that this hypothesis class also leads to a convenient decomposition of loss functions mirroring the likelihood decomposition described in Chapter 6, which permits efficient optimisation of several popular image-level losses.

For learning we employ the tools of structured prediction — in particular, the structured support vector machine [62]. The use of these tools is part of a long trend within computer

vision towards statistically rigorous, well-understood convex optimisation techniques. Recent successfully applications of the structured SVM include detection [5], segmentation [60], and scene classification [68]. In the domain of reconstruction, structured prediction ideas have been applied to several simple models classes such as stereo disparities [43] and cuboids [30].

The application of structured prediction to the indoor Manhattan class of models constitutes one of the most complex output spaces yet considered within this framework. The indoor Manhattan model enforces hard geometric constraints that lack simple expressions in terms of individual pixel labels. These constraints are time-varying, being tied to quantities such as camera rotation and the location of vanishing points. We have learnt several valuable lessons of general relevance from this complex prediction task, to which we dedicate the final section of this chapter.

The contributions of this chapter are thus (i) a unified learning framework for single- and multiple-view reconstruction, utilising the indoor Manhattan model and the tools of structured prediction; (ii) the reduction of two image-level loss functions to a form amenable to efficient optimisation; (iii) an efficient separation procedure for identify the “most-violated constraint” during learning, (iv) an empirical demonstration of structured prediction in perhaps the most complex output space yet considered within this framework; and (v) a series of practical observations concerning the application of structured prediction techniques to complicated output spaces.

7.2 Background

The approach to learning adopted in this chapter has roots in the structural risk minimisation framework developed by Vapnik and Chervonenkis [64]. On this view, learning is framed in terms of a predictor f , which maps from an input space \mathcal{X} to an output space \mathcal{Y} . Learning consists of minimising the expected loss

$$\mathbb{E}[\Delta(f(\mathbf{x}), \mathbf{y})] = \int \Delta(f(\mathbf{x}), \mathbf{y}) \, dP(\mathbf{x}, \mathbf{y}) \quad (7.1)$$

with respect to the predictor f , where Δ is a loss function. Given training samples $(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{X} \times \mathcal{Y}$, the integral above is approximated by

$$\mathcal{R}(f) + \sum_i \Delta(f(\mathbf{x}_i), \mathbf{y}_i) \quad (7.2)$$

where $\mathcal{R}(f)$ is a regulariser on f . Structural risk minimisation does not specify any particular form for the input space, output space, predictor class, or loss function. Many machine learning algorithms can be viewed as instantiations of structural risk minimisation for particular predictor classes and loss functions. For example, the classic support vector machine (SVM) [9] is defined for Hilbert input spaces, binary output spaces, the hypothesis class of linear predictors, and the Hinge loss function. In this case equation (7.2) becomes

$$\|\mathbf{w}\|^2 + \sum h(\mathbf{y}_i, \mathbf{w} \cdot \mathbf{x}_i)) \quad (7.3)$$

where h is the hinge loss, and learning consists of minimisation with respect to \mathbf{w} . Rewriting the hinge loss as a soft constraint leads to the well-known quadratic programming formulation of the SVM [9].

Extending binary classification to more general output spaces is a major research programme within machine learning; an excellent introduction to recent advances is given by Bakir *et al.* [2]. One recently successful approach is the structured support vector machine proposed by Tsochantaridis *et al.* [62], which reduces learning in non-binary (“structured”) output spaces to a ranking problem. The structured SVM makes use of a joint feature space $\Psi(\mathbf{x}, \mathbf{y})$ mapping input/output pairs to real vectors. The hypothesis class consists of predictors of the form

$$f(\mathbf{x}; \mathbf{w}) = \operatorname{argmax}_{\mathbf{y}} \langle \mathbf{w}, \Psi(\mathbf{x}, \mathbf{y}) \rangle \quad (7.4)$$

where \mathbf{w} is a parameter vector and $\langle \cdot, \cdot \rangle$ denotes an inner product. As in the classic SVM, the hinge loss is used to define the objective function,

$$\|\mathbf{w}\|^2 + \sum h(\mathbf{y}_i, f(\mathbf{x}_i)). \quad (7.5)$$

In order to optimise with respect to \mathbf{w} , Toschantaridis *et al.* showed how to write (7.5) as a constrained optimisation problem paralleling the classic SVM:

$$\begin{aligned} \min_{\mathbf{w}, \xi} \quad & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i \\ \text{s.t. } \forall k, \mathbf{y} \neq \mathbf{y}_i : \quad & \langle \mathbf{w}, \Psi(\mathbf{x}_i, \mathbf{y}_i) \rangle - \langle \mathbf{w}, \Psi(\mathbf{x}_i, \mathbf{y}) \rangle \geq \Delta(\mathbf{y}, \mathbf{y}_i) - \xi_i . \end{aligned} \quad (7.6)$$

Unfortunately the number of constraints in (7.6) is in general very large since there is a constraint for every element of the output space. Tsochantaridis *et al.* [62] showed how to overcome this by exploiting the sparsity structure of support vector machines. Their algorithm iteratively adds constraints beginning with zero constraints, and they showed that only a polynomial number of constraints need ever be added in order to achieve an ϵ -accurate solution. The remainder of this chapter shows how to use the structured SVM to learn to reconstruct indoor Manhattan models.

7.3 Model

In this section we describe three components of the model that we are trying to learn (and, at test time, infer): a hypothesis class, a feature space, and a loss function. In our setup these are, respectively, the class of indoor Manhattan models, a log-linear Bayesian likelihood, and either the relative depth error or a labelling error (we describe both).

7.3.1 Hypothesis Class

This chapter is concerned with the output space consisting of indoor Manhattan reconstructions as described in Chapter 4. For consistency with preceding chapters we use the following notation: our input space consists of observed image features denoted Φ , and the output space consists of indoor Manhattan reconstructions in the seam representation denoted S . A training sample is then a set of pairs $\{(\Phi_i, S_i)\}$.

7.3.2 Feature Space

In order to learn to reconstruct indoor Manhattan models we must define a family of prediction functions f mapping image features to indoor Manhattan models. Building on Chapter 6, we would like each predictor to implement MAP inference for some value of the hyperparameters, *i.e.* we are interested in predictors of the form

$$f(\Phi) = \operatorname{argmax}_S P(S | \Phi, \mathbf{w}) \quad (7.7)$$

for various $\mathbf{w} \in \mathbb{R}^n$. In other words, we would like to optimise over the set

$$\left\{ f \mid f : \Phi \mapsto \operatorname{argmax}_S P(S | \Phi, \mathbf{w}), \mathbf{w} \in \mathbb{R}^n \right\} \quad (7.8)$$

consisting of MAP inference predictors for all possible hyper-parameter values, which we have grouped into the parameter \mathbf{w} . In order to employ the structured SVM we need to write f in the form (7.4), or equivalently, we need to write the logarithm of the posterior on scene structure in the form

$$\log P(S | \Phi, \mathbf{w}) = \langle \mathbf{w}, \Psi(\Phi, S) \rangle. \quad (7.9)$$

That is, we need to show that the posterior described in the previous chapter is log-linear. The remainder of this section is devoted to defining a joint feature map Ψ permitting such a representation. We now examine each term in the posterior (6.42), our goal being to show that each is log-linear in the relevant hyper-parameters.

Prior

Recall the log-prior (6.8), which is repeated below.

$$\log P(S | \boldsymbol{\lambda}) = -\log Z + n_1 \log \lambda_1 + n_2 \log \lambda_2 + n_3 \log \lambda_3. \quad (7.10)$$

Defining $\mathbf{n} = \begin{bmatrix} n_1 & n_2 & n_3 \end{bmatrix}$ we have

$$\log P(S | \boldsymbol{\lambda}) = \langle \boldsymbol{\lambda}, \mathbf{n} \rangle + O(1). \quad (7.11)$$

Photometric Likelihood

Combining (6.15) and (6.11), the log likelihood for photometric features that was described in Chapter 6 is

$$\log P(\Phi | S) = \sum_i \boldsymbol{\omega}_{a_i^*} \cdot \boldsymbol{\phi}_i + O(1). \quad (7.12)$$

where the sum is over pixels, and, as defined in Chapter 6, a_i^* is the orientation predicted by S for pixel i , $\boldsymbol{\omega}_a$ is the model for photometric features given orientation a , and $\boldsymbol{\phi}_i$ is the feature observed at pixel i . We define

$$\boldsymbol{\omega} = \begin{bmatrix} \boldsymbol{\omega}_1 \\ \boldsymbol{\omega}_2 \\ \boldsymbol{\omega}_3 \end{bmatrix} \quad \Psi_{\text{mono}}(\Phi, S) = \begin{bmatrix} \sum_{i:a_i^*=1} \boldsymbol{\phi}_i \\ \sum_{i:a_i^*=2} \boldsymbol{\phi}_i \\ \sum_{i:a_i^*=3} \boldsymbol{\phi}_i \end{bmatrix}. \quad (7.13)$$

Finally we can write

$$\log P(\Phi | S) = \langle \boldsymbol{\omega}, \Psi_{\text{mono}}(\Phi, S) \rangle + O(1). \quad (7.14)$$

Photo-consistency Likelihood

The log likelihood for photo-consistency features given by equation (6.24) is

$$\log P(\Phi, \Theta_1, \dots, \Theta_K | S) = \sum_i \sum_{k=0}^K \|\boldsymbol{\phi}_i - \bar{\boldsymbol{\theta}}_k(\mathbf{p}_i; S)\|_\Sigma. \quad (7.15)$$

Here we assume that

$$\Sigma^{-1} = \text{diag}(\sigma_1, \dots, \sigma_2). \quad (7.16)$$

It follows that by defining

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_1 \\ \vdots \\ \sigma_m \end{bmatrix} \quad \Psi_{\text{stereo}}(\Phi, S) = \begin{bmatrix} \sum_i \sum_{k=1}^K ((\phi_i)_1 - (\bar{\theta}_k(\mathbf{p}_i; S))_1)^2 \\ \vdots \\ \sum_i \sum_{k=1}^K ((\phi_i)_m - (\bar{\theta}_k(\mathbf{p}_i; S))_m)^2 \end{bmatrix} \quad (7.17)$$

we can write the log likelihood (7.15) in the form

$$\log P(\Phi, \Theta_1, \dots, \Theta_K) = \langle \boldsymbol{\sigma}, \Psi_{\text{stereo}}(\Phi, S) \rangle + O(1). \quad (7.18)$$

Point Cloud Likelihood

The log-likelihood for point cloud features (6.35) is non-linear so for the purpose of learning we approximate it by a first order Taylor expansion about $\boldsymbol{\tau}_0$ as follows.

$$\log P(d | S, \boldsymbol{\tau}) \approx \frac{\sum_t P(d | S, t)P(t | \boldsymbol{\tau})}{\sum_t P(d | S, t)P(t | \boldsymbol{\tau}_0)} + O(1) \quad (7.19)$$

$$\log P(D | S, \boldsymbol{\tau}) \approx \sum_i \sum_t \frac{P(d_i | S, t)P(t | \boldsymbol{\tau})}{\sum_t P(d_i | S, t)P(t | \boldsymbol{\tau}_0)} + O(1). \quad (7.20)$$

It follows that by defining

$$\Psi_{3D}(D, S) = \frac{1}{\sum_t P(d_i | S, t)P(t | \boldsymbol{\tau}_0)} \begin{bmatrix} \sum_i P(d_i | S, t = \text{IN})P(t = \text{IN} | \boldsymbol{\tau}) \\ \sum_i P(d_i | S, t = \text{OUT})P(t = \text{OUT} | \boldsymbol{\tau}) \\ \sum_i P(d_i | S, t = \text{ON})P(t = \text{ON} | \boldsymbol{\tau}) \end{bmatrix}. \quad (7.21)$$

we can write

$$\log P(D | S, \boldsymbol{\tau}) = \langle \boldsymbol{\tau}, \Psi_{3D}(D, S) \rangle + O(1). \quad (7.22)$$

Posterior

We have now shown that the likelihoods we defined in Chapter 6 for each sensor modality are log-linear in the respective hyper-parameters. Combining equations (7.11), (7.14), (7.18), and

(7.22) we have the following linear form for the log–posterior,

$$\log P(S | X, \mathbf{w}) = \langle \mathbf{w}, \Psi(X, S) \rangle \quad (7.23)$$

where X denotes all observations from all sensor modalities and we have defined

$$\mathbf{w} = \begin{bmatrix} \lambda \\ \omega \\ \sigma \\ \tau \end{bmatrix} \quad \Psi(X, S) = \begin{bmatrix} n \\ \Psi_{\text{mono}} \\ \Psi_{\text{stereo}} \\ \Psi_{\text{3D}} \end{bmatrix}. \quad (7.24)$$

Discussion

We have shown in the derivations above that the posterior from Chapter 6 is log–linear (or can be approximated as such), so substituting (7.23) into (7.7) we are now interested in predictors of the form

$$f(\Phi) = \underset{S}{\operatorname{argmax}} \langle \mathbf{w}, \Psi(X, S) \rangle. \quad (7.25)$$

One view of this result is that it is a convenient property of our chosen likelihoods that allows us to efficiently optimise the hyper–parameters in the graphical models of Chapter 6 using learning algorithm that we describe below. However, an alternative view is that the joint feature map is simply an arbitrary feature space with no special interpretation attached to any element. Each element of the feature space simply expands the range of predictors that can be expressed in the form (7.25), and we may add whichever features we believe will be helpful for reconstruction. The fact that our features are derived from a graphical model is suggestive of their relevance but not absolutely necessary.

Both views lead to the same learning objective, which is to optimise \mathbf{w} with respect to an expected loss over a training set.

Feature	Dimensionality	Multi-view?	Single-view?	Reference
Stereo photo-consistency	4	yes	no	
Point cloud	2	yes	no	
Line sweeps	1	yes	yes	Lee <i>et al.</i> [13]
RGB+HSV	6	no	yes	
Gabor responses ²	12	no	yes	

Table 7.1: The composition of our single- and multiple-view feature space. We omit colour and Gabor features from the multiple-view feature space for training efficiency.

Features

The precise make-up of the feature space depends on the available sensor modalities. For our experiments we define separate feature spaces for the single- and multiple-view contexts; these are summarised in Figure 7.1.

7.3.3 Loss Functions

In this section we define a loss function $\Delta(S, \hat{S})$ measuring the cost of predicting some reconstruction \hat{S} when in fact the true reconstruction is S . In the context of learning one often faces a trade-off between choosing a loss that leads to tractable optimisation, and choosing a loss that measures the quantity that one “really” cares about. For example, Hoiem *et al.* [33] learn a per-segment orientation classifier, then pass this as input to a separate 3D reconstruction system [32]. However, what one “really” cares about is some loss defined on the output of the entire system rather than the output of individual components. This is not a criticism of those authors’ choice, but an illustration of the trade-off faced when choosing a loss function. In this chapter we show how to learn efficiently with respect to loss functions defined on the final reconstruction.

The relative depth error has been the gold standard within the reconstruction community for more than a decade [27], and measures the average deviation between reconstructed and

²4 orientations, 3 scales

ground truth depths. In our notation,

$$\Delta_{\text{depth}}(S, \hat{S}) = \frac{1}{N} \sum_{\mathbf{p}} \frac{|d(\mathbf{p}; \hat{S}) - d(\mathbf{p}; S)|}{d(\mathbf{p}; S)}, \quad (7.26)$$

where N is the number of pixels. Another reasonable choice is the labelling error, used widely within the semantic segmentation literature,

$$\Delta_{\text{labelling}}(\hat{S}, S) = \frac{1}{N} \sum_{\mathbf{p}} [a(\mathbf{p}; \hat{S}) \neq a(\mathbf{p}; S)], \quad (7.27)$$

where $[p]$ is 1 if p is true and 0 otherwise. An attractive characteristic of the indoor Manhattan class is that *both of these losses can be optimised exactly*. The algorithmic details are left to Section 7.4; the key result we establish here is that Δ_{depth} and $\Delta_{\text{labelling}}$ can be written in a form resembling the payoff formulation (6.1) for the feature likelihoods.

First we invoke the independence between columns established in Section 4.5.4:

$$\Delta_{\text{depth}}(\hat{S}, S) = \frac{1}{N} \sum_{x=1}^W \sum_{y=1}^H \frac{|\tilde{d}(x, y; \hat{s}_x) - d(x, y; S)|}{d(x, y; S)}. \quad (7.28)$$

Defining a real matrix δ_S ,

$$\delta_S(x, j) = \frac{1}{N} \sum_{y=1}^H \frac{|\tilde{d}(x, y; j) - d(x, y; S)|}{d(x, y; S)}, \quad (7.29)$$

we see that we can write Δ_{depth} in the form

$$\Delta_{\text{depth}}(\hat{S}, S) = \sum_{x=1}^W \delta_S(x, \hat{s}_x). \quad (7.30)$$

There is a similar form for $\Delta_{\text{labelling}}$, which we omit here because it follows almost identically to the derivations above.

Choosing a Loss Function

Neither of the above losses (relative depth or labelling error) is unequivocally the “correct” loss; the choice will depend on the application at hand. One might expect a strong correlation between the losses, and indeed one can show analytically that

$$\Delta_{\text{depth}}(\hat{S}, S) = 0 \iff \Delta_{\text{labelling}}(\hat{S}, S) = 0. \quad (7.31)$$

However, in our experiments we found only a weak correlation between these losses away from the origin. For example, the scatter plot shown in figure Figure 7.1 shows a significant number of outliers that score very well on Δ_{depth} but poorly on $\Delta_{\text{labelling}}$, and vice versa. This underscores the importance of choosing a loss function suitable for the problem at hand, and highlights the advantages of a learning scheme that can incorporate widely-used loss functions such as the two discussed above.

7.4 Learning Algorithm

We turn now to the problem of learning within the model described above. Our learning task is to identify a prediction function f mapping observed features Φ to reconstructions S . We seek the loss minimiser

$$f^* = \operatorname{argmin}_f \mathbb{E} [\Delta(f(\Phi), S)], \quad (7.32)$$

which we approximate in the framework of structural risk minimisation as

$$f^* = \operatorname{argmin}_f \mathcal{R}(f) + \sum_k \Delta(f(\Phi_k), S_k), \quad (7.33)$$

where k indexes a training set and \mathcal{R} is a regulariser. To perform this optimisation we turn to the tools of structured prediction [2], and in particular the structured SVM [62]. Following [62]

we cast the learning problem as a constrained optimisation problem,

$$\begin{aligned} \min_{\mathbf{w}, \xi} \quad & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{k=1}^n \xi_k \\ \text{s.t. } \forall k, S \neq S_k : \quad & \langle \mathbf{w}, \Psi(\Phi_k, S_k) \rangle - \langle \mathbf{w}, \Psi(\Phi_k, S) \rangle \geq \Delta(S, S_k) - \xi_k . \end{aligned} \tag{7.34}$$

where C is a user-specified parameter that trades off between prediction accuracy and model complexity. Tsochantaridis *et al.* [62] described an algorithm for solving this minimisation that is now used extensively within machine learning and computer vision. To apply this algorithm here we must solve two inference problems:

1. *Prediction*. This is the maximisation described in (7.25).
2. *Separation*. The algorithm described in [62] requires a user-supplied procedure to find the “most-violated constraint” at each iteration. That is,

$$\operatorname{argmax}_S \langle \mathbf{w}, \Psi(\Phi_k, S) \rangle + \Delta(S, S_k) . \tag{7.35}$$

Our solutions to both of the above build on the dynamic programming algorithm presented in Chapter 6, which solves problems of the form

$$\operatorname{argmax}_S \sum_x \pi(x, s_x) - \sum_j \gamma(j; S) . \tag{7.36}$$

7.4.1 Inference (Prediction)

We showed in Section 7.3 that (7.36) can be written in the form (7.25), so the prediction problem is a straightforward application of the inference algorithm of Chapter 6. This is as expected since, as we have already remarked, (7.25) is equivalent to MAP inference on indoor Manhattan reconstructions, which was precisely the subject of Chapter 6.

7.4.2 Loss–Augmented Inference (Separation)

The separation problem can also be solved using the dynamic programming algorithm from Chapter 6, as the following proposition shows.

Proposition 1. *Let (Φ_k, S_k) be a training instance with payoff matrices $\{\pi_i\}$ as defined in Chapter 6.*

Let

$$\pi_{\text{aug}} = \delta_{S_k} + \sum_i \pi_i . \quad (7.37)$$

Then the solution to (7.36) with $\pi = \pi_{\text{aug}}$ is identical to the solution to (7.35).

Proof. Direct equivalence of the expressions to be maximised. First substitute (7.23) and (7.30) into (7.35):

$$\log P(S | \Phi) + \sum_{x=1}^W \delta_{S_k}(x, s_x) \quad (7.38)$$

Further substituting (6.45) and defining γ as in Chapter 6 gives

$$\sum_i \sum_{x=1}^W \pi_i(x, s_x) - \sum_j \gamma(j; S) + \sum_{x=1}^W \delta_{S_k}(x, s_x) . \quad (7.39)$$

Finally we see that substituting (7.37) gives

$$\sum_{x=1}^W \pi_{\text{aug}}(x, s_x) - \sum_j \gamma(j; S) . \quad (7.40)$$

which is precisely (7.36) under the desired substitution $\pi = \pi_{\text{aug}}$. \square

7.5 Multiple View Results

We evaluated our approach on the same data-set used in Chapter 6, which consists of 18 sequences of six environments averaging 59 seconds in duration. We sampled key-frames at regular intervals. Each “instance” in our training and hold-out sets consists of one base frame together with four auxiliary frames.

We compared with the bootstrapping approach described in [23] and outlined in the preceding chapter. Our metrics differ from those of the previous chapter in two ways. Firstly, the previous chapter computed relative depth error using the maximum of the ground truth and estimated depths in the denominator, whereas we always use the ground truth in the denominator. These metrics are separated by at most a monotonic transform but we must use the latter here in order to represent the loss function in the form (7.30). Secondly, when we compute labelling error we differentiate vertical and horizontal surfaces only, whereas the previous chapter also differentiate the two vertical orientations. The latter approach makes a side-by-side comparison difficult because the two vertical orientations are symmetric and their labels can always be interchanged.

We described two loss functions above: the relative depth error and the labelling error. We also described two feature spaces: one containing exclusively single view features and one containing and multiple view features. In total we trained four predictors, corresponding to all combinations of feature spaces and loss functions. For clarity we will use the following notation throughout this section.

$f_{\text{depth}}^{\text{sview}}$	Trained with respect to Δ_{depth} in the single view feature space
$f_{\text{labelling}}^{\text{sview}}$	Trained with respect to $\Delta_{\text{labelling}}$ in the single view feature space
$f_{\text{depth}}^{\text{mview}}$	Trained with respect to Δ_{depth} in the multiple view feature space
$f_{\text{labelling}}^{\text{mview}}$	Trained with respect to $\Delta_{\text{labelling}}$ in the multiple view feature space

The performance of all predictors are summarised in Table 7.2. Our approach significantly out-performs the approach of the previous chapter. Anecdotally we noticed that much of the improvement resulted from a reduction in catastrophic failures. This makes sense because we would expect the learning algorithm to concentrate on reducing those mistakes that result in the largest loss. Some example predictions in the multiple-view feature space are shown in Figure 7.6.

Figure 7.2 shows the evolution of the weight vector w during training for each of the four

predictors. The primary conclusion to draw from this figure is that the weights do in fact converge, and that the single-view scenario is, as expected, the more difficult learning problem. Finally, Figure 7.1 compares the error distribution on two error metrics for the two multiple-view predictors. The fact that the distributions are skewed along opposite axes shows that training with respect to different loss functions has indeed resulted in different prediction rules, which correctly minimise their respective losses on held-out instances.

Sequence	Multiple-view			Single-view		
	$f_{\text{depth}}^{\text{mview}}$	$f_{\text{labelling}}^{\text{mview}}$	Chapter 6 ¹	$f_{\text{depth}}^{\text{sview}}$	$f_{\text{labelling}}^{\text{sview}}$	Chapter 6 ²
ground	4.9	5.5	24.5	17.3	28.0	66.6
foyer1	6.1	4.5	31.0	25.1	29.2	6.6
foyer2	4.3	4.9	30.1	29.1	28.5	5.4
corridor	14.6	15.0	33.6	31.7	32.7	52.9
mcr	34.0	37.3	45.9	70.1	58.6	67.6
kitchen	16.8	19.3	26.2	25.1	25.8	23.6
Average	13.4	14.4	31.9	33.1	33.8	37.1

(a) Relative depth error

Sequence	Multiple-view			Single-view		
	$f_{\text{depth}}^{\text{mview}}$	$f_{\text{labelling}}^{\text{mview}}$	Chapter 6 ¹	$f_{\text{depth}}^{\text{sview}}$	$f_{\text{labelling}}^{\text{sview}}$	Chapter 6 ²
ground	2.8	2.9	10.4	14.8	7.8	12.4
foyer1	3.6	3.1	3.1	20.3	15.1	22.2
foyer2	3.3	3.7	4.0	18.6	15.9	18.6
corridor	11.4	9.5	19.2	23.7	19.3	24.8
mcr	15.4	15.8	16.2	24.5	26.7	20.8
kitchen	5.3	5.2	6.1	11.0	7.7	11.9
Average	7.0	6.7	9.8	18.8	15.4	18.4

(b) Labelling error

Table 7.2: Performance for each predictor, for each sequence, and for each error metric. The six predictors under comparison are listed in the top row of each table — see the main text for details on these symbols. Note that even when a predictor is trained with respect to a particular loss function, we may still evaluate it with respect to a different metric (though as expected, the predictors perform best when evaluated by the same error metric that they were trained for).

¹This column corresponds to using all sensor modalities

²This column corresponds to using photometric features alone

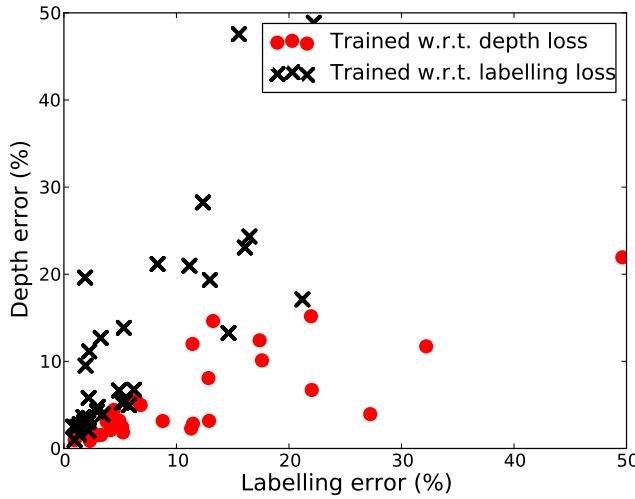


Figure 7.1: The effect of the loss function on training. We train two predictors, one with respect to Δ_{depth} and one with respect to $\Delta_{\text{labelling}}$, then evaluate both on all held-out instances. Each data point shows the error obtained by one predictor on one held-out instance. The differing distribution of errors shows that the two predictors trade off errors as expected.

7.6 Single View Results

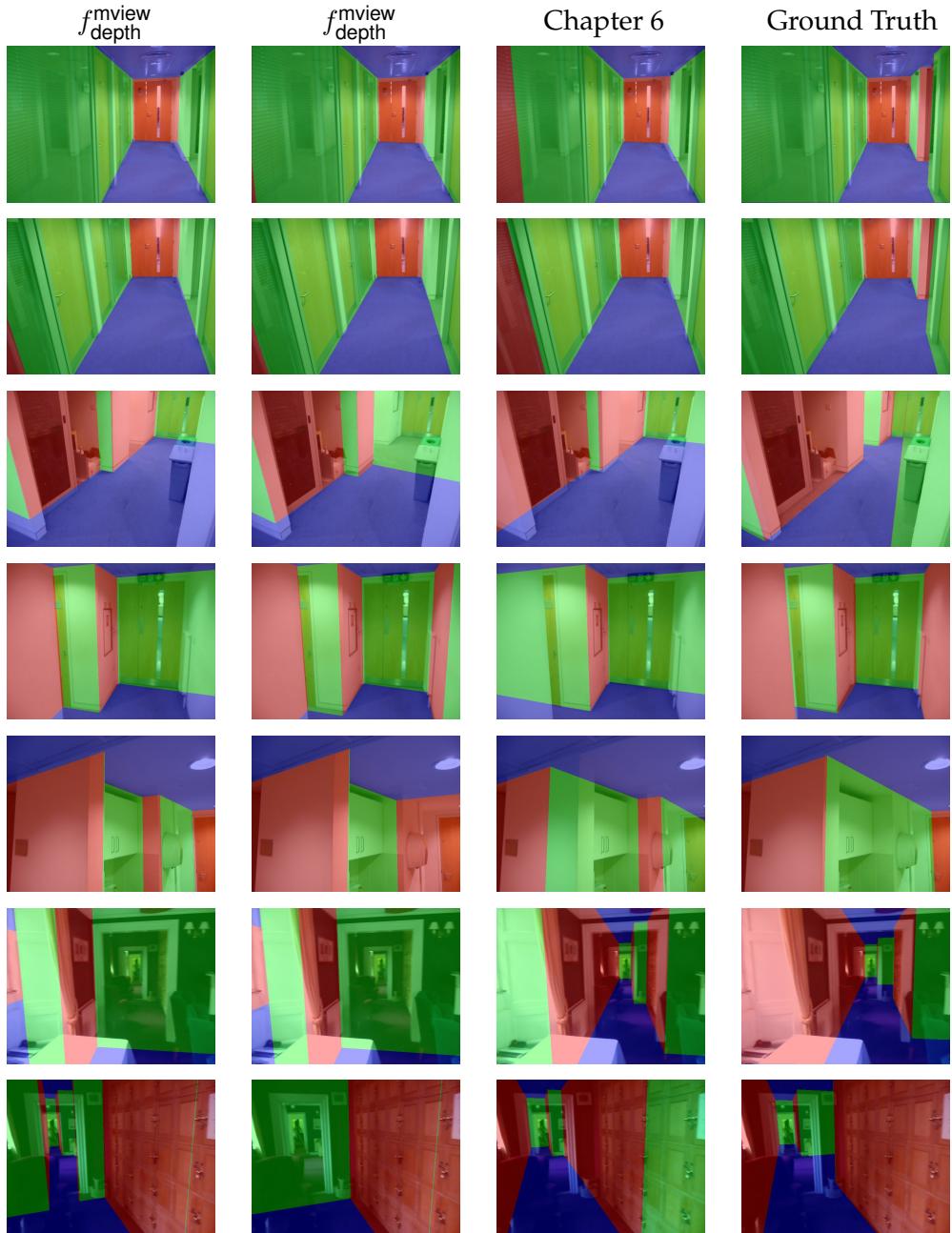
We evaluated our system for single-view reconstruction using the same data-set described in the previous section. We used the single-view features summarised in Figure 7.1. We compared our approach to the single-view approach described in the previous chapter and in [20], which uses only line-sweep features.

Performance for each algorithm is summarised in Table 7.2. When measured by labelling error, our approach out-performs the comparison system, but on the depth error metric our approach is inferior. While investigating this result we found that our learning algorithm assigns small weights to all but the line-sweep features, which are the same features used by the single-view-only instantiation of Chapter 6. This suggests that the hand-tuned weights are in fact close to optimal within this feature space, though one would expect that with additional feature engineering our learning algorithm would be able to leverage more salient information and reduce the error rate further.

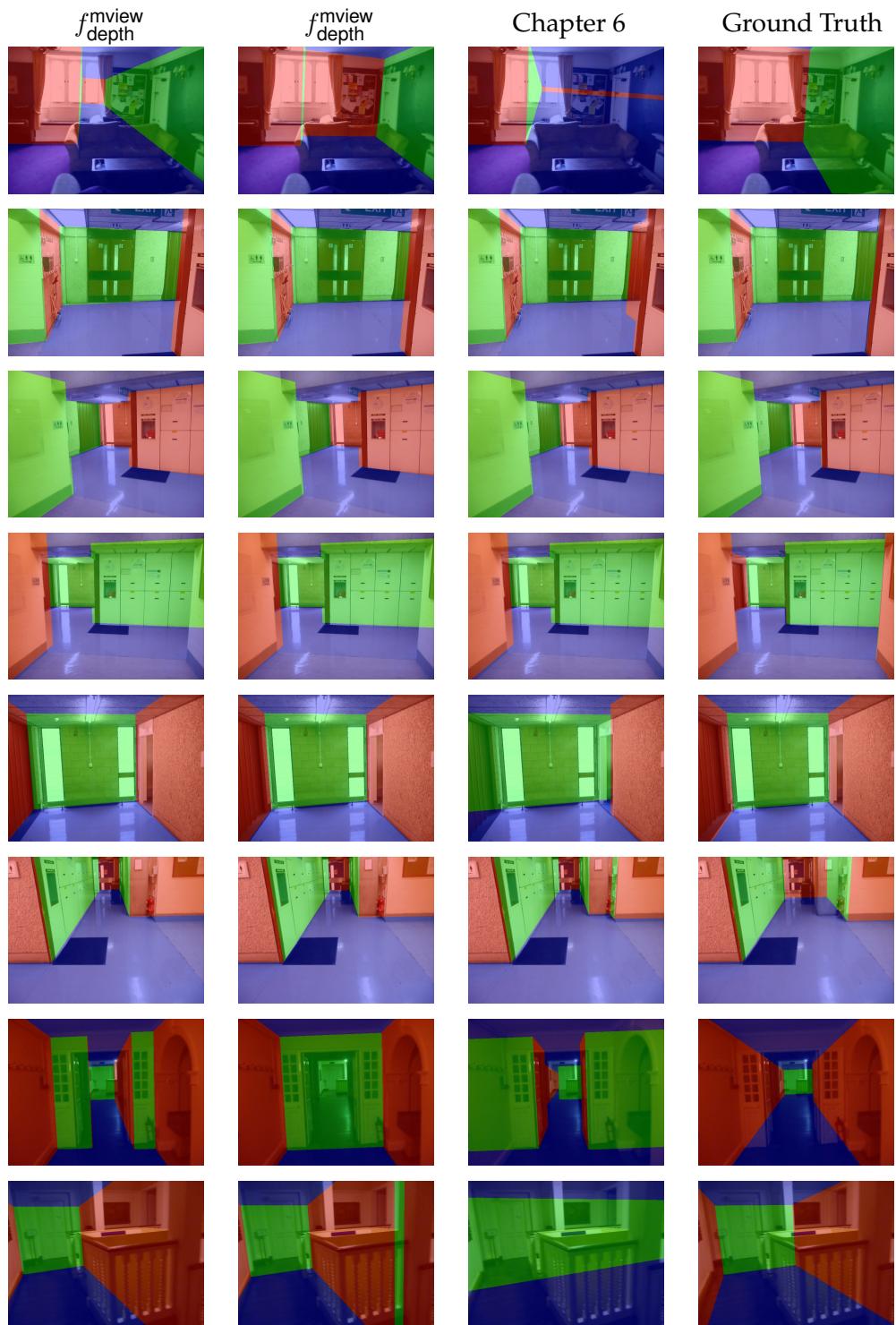
The evolution of the weight vector w over the course of training is shown in Figure 7.2. Due to the size of the single view feature space we show selected features only. The single view

learning problem is, as expected, the more difficult problem, as shown by the considerably longer training times and the higher volatility during the exploration phase.

Table 7.3: Reconstructions predicted by our system using the multiple view feature space; comparison with the bootstrapping approach described in Chapter 6 and in [23]. The first two columns correspond to the predictors trained on Δ_{depth} and $\Delta_{\text{labelling}}$ respectively, the third column is from Chapter 6, and the fourth column is ground truth. This figure shows samples from the hold-out set, and were selected uniformly at random for inclusion here.



Continued on next page



Continued on next page

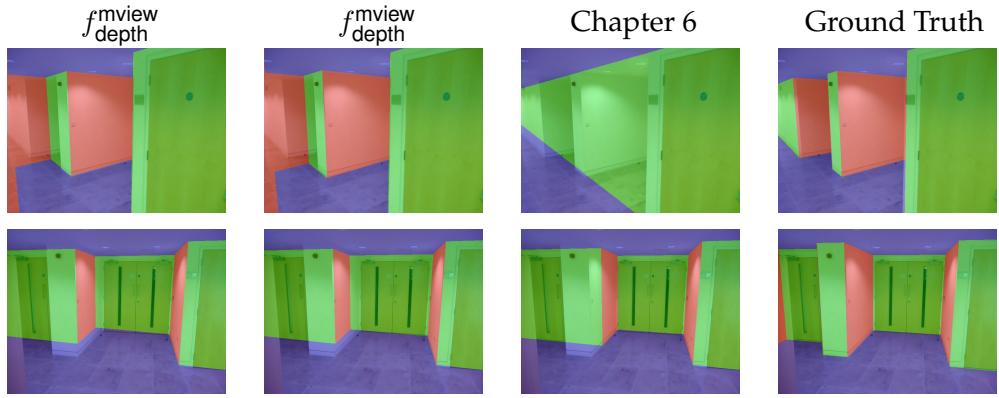
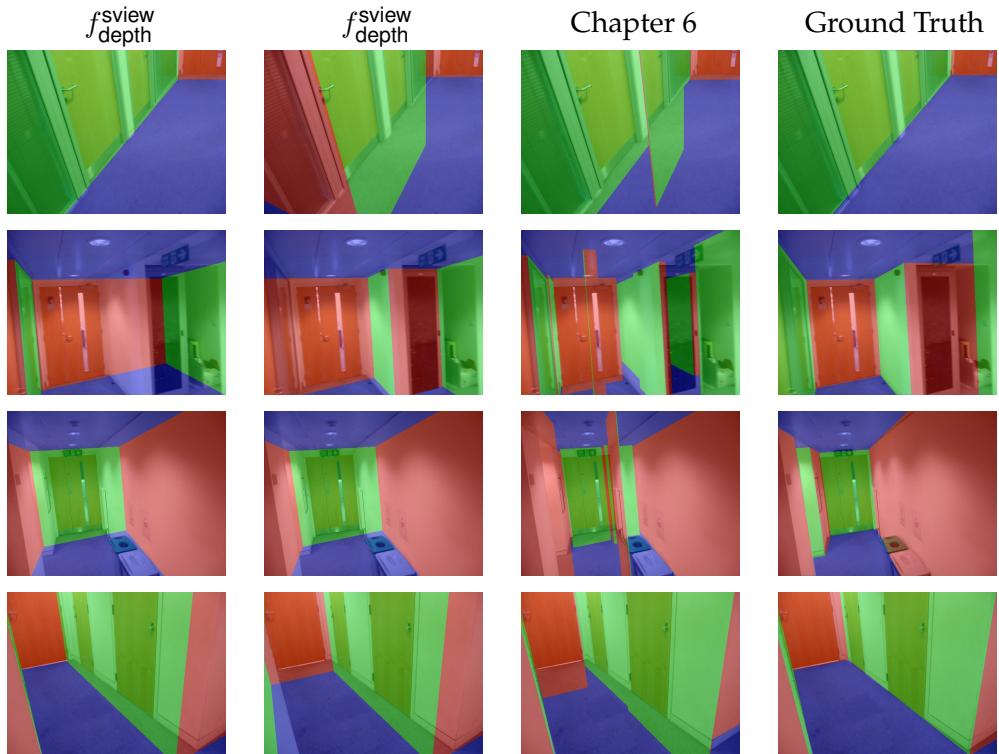
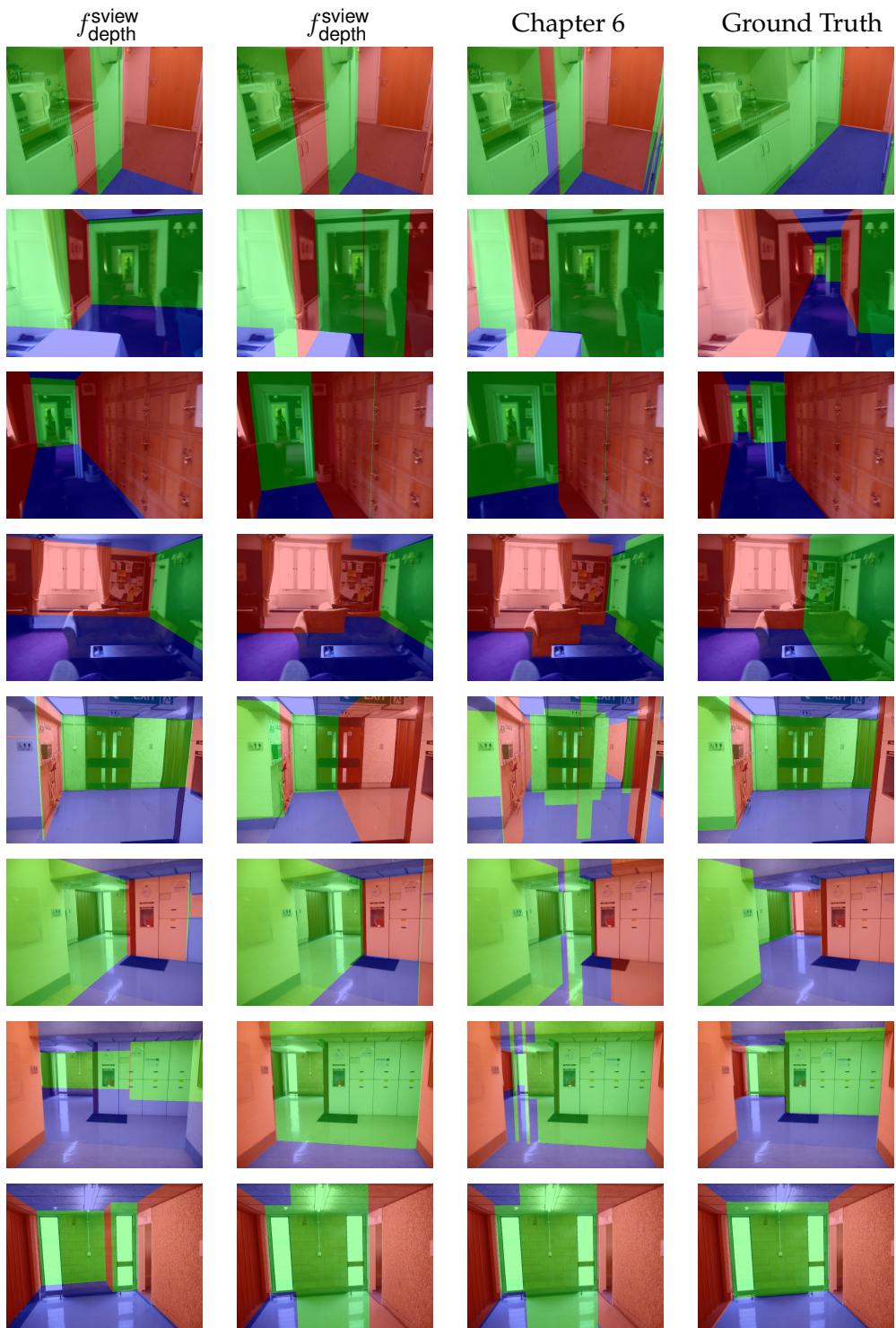


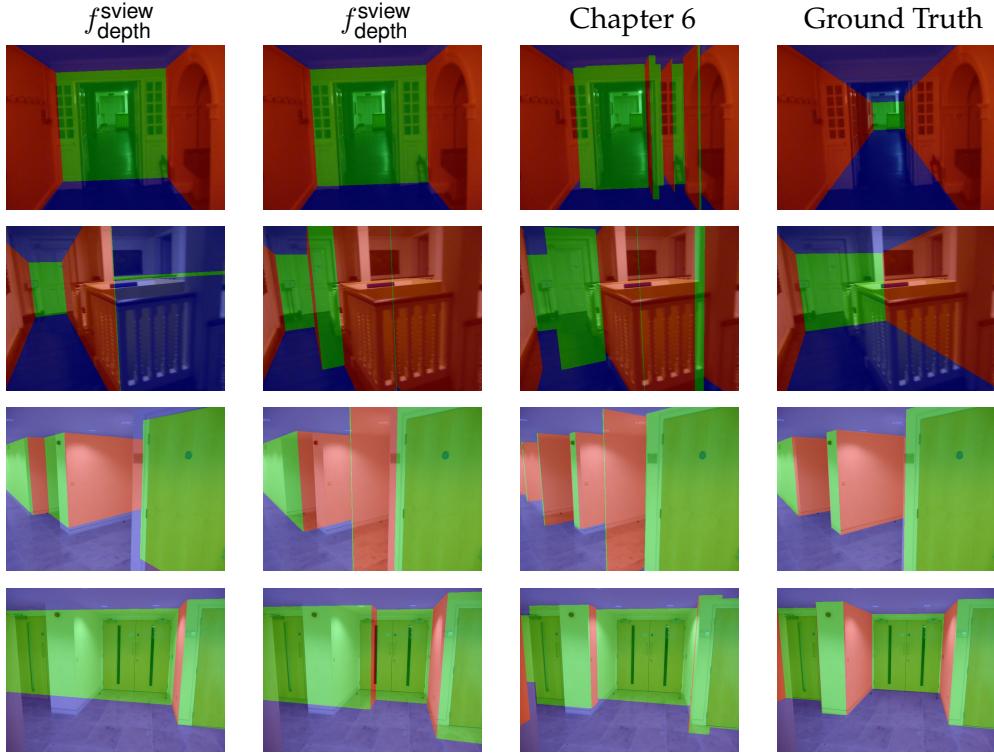
Table 7.4: Reconstructions predicted by our system using the single view feature space; comparison with the bootstrapping approach described in Chapter 6 and in [23]. The first two columns correspond to the predictors trained on Δ_{depth} and $\Delta_{\text{labelling}}$ respectively, the third column is from Chapter 6 using photometric features alone, and the fourth column is ground truth. This figure shows samples from the hold-out set, and were selected uniformly at random for inclusion here.



Continued on next page



Continued on next page



7.7 Discussion

The hypothesis class considered in this chapter is amongst the most complex (in terms of internal constraints on the output space) studied within the structured prediction framework. In this section we turn to some practical lessons learned that may be of value to other practitioners.

7.7.1 Condition in the joint feature space, not the input feature space.

A common pre-processing operation for statistical learning is to transform the observed features Φ to zero mean and unit variance. However, for structured prediction tasks it is the joint feature space Ψ , not the input feature space, that should be conditioned:

$$\Psi' = \frac{\Psi - \mu}{\sigma^2} . \quad (7.41)$$

Ideally one would sample directly from the joint feature space to determine the conditioning transformation, but the distribution of inputs and outputs is generally unknown in an empirical risk minimisation setting. Instead, we suggest using the training set as a proxy. We compute the empirical mean and variance of $\{\Psi(\Phi_k, S_k)\}_{k=1}^n$ for the training set at the outset, then apply the transformation (7.41) after each feature computation.

7.7.2 Condition the loss terms.

For any $\eta > 0$, the minimisation problem (7.34) is equivalent to the following (this is proved in Proposition 2 at the end of this chapter):

$$\begin{aligned} \min_{\mathbf{w}', \xi'} \quad & \frac{1}{2} \|\mathbf{w}'\|^2 + \eta C \sum_{k=1}^n \xi'_k \\ \text{s.t. } \forall k, S \neq S_k : \quad & \langle \mathbf{w}', \Psi(\Phi_k, S_k) \rangle - \langle \mathbf{w}', \Psi(\Phi_k, S) \rangle \geq \eta \Delta(S, S_k) - \xi'_k . \end{aligned} \quad (7.42)$$

Although any $\eta > 0$ preserves the correctness of the optimisation algorithm, we found that choosing $\eta = \text{Var}(\Delta)$ improved numerical stability, since this means that the loss terms will have roughly unit variance. Unfortunately, we cannot use the training set to estimate $\text{Var}(\Delta)$ since the loss for the ground truth output is always zero. Instead we computed $\Delta(\Phi_k, S_j)$ for each $k \neq j$ in the training set. This is not an ideal estimate, but we found that it worked well in practice.

7.7.3 Check that the hypothesis class contains the ground truth.

The algorithm described in [62] implicitly assumes that the hypothesis class \mathcal{Y} contains the ground truth labels S_k . This means that if S^+ is the maximiser of (7.35) then

$$\langle \mathbf{w}, \Psi(\Phi_k, S_k) \rangle - \langle \mathbf{w}, \Psi(\Phi_k, S^+) \rangle - \Delta(S^+, S_k) \leq 0 , \quad (7.43)$$

since otherwise we would have

$$\langle \mathbf{w}, \Psi(\Phi_k, S_k) \rangle + \Delta(S_k, S_k) > \langle \mathbf{w}, \Psi(\Phi_k, S^+) \rangle + \Delta(S^+, S_k), \quad (7.44)$$

contradicting S^+ as the maximiser of (7.35). However, our output space contains fundamentally real-valued quantities such as polygon vertices, which are recovered only to some finite precision by the inference algorithm, and since our ground truth labels were acquired by manual labelling, they sometimes exceed the maximum precision of the inference algorithm. In this case we effectively have $S_k \notin \mathcal{Y}$ (although there is always some $S' \in \mathcal{Y}$ close to S_k), so it is possible that S^+ violates (7.43). Our workaround here is simply to check the condition (7.43) each time we solve the separation problem and, if violated, substitute S_k for S^+ . This is justified by the observation that if (7.43) is violated for S^+ then it is violated for all $S \in \mathcal{Y}$. One could think of this as learning with respect to the hypothesis class $\mathcal{Y} \cup \{S_k\}$ but evaluating with respect to \mathcal{Y} . Again, this is not an ideal solution but we found it to work well in practice.

7.7.4 Equivalence of (7.34) and (7.42)

Here we prove the equivalence of the minimisation problems (7.34) and (7.42) stated above. For clarity we re-state the claim in the following proposition.

Proposition 2. *Let \mathbf{w}, ξ be the solution to*

$$\begin{aligned} \min_{\mathbf{w}, \xi} \quad & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{k=1}^n \xi_k \\ \text{s.t. } & \forall k, S \neq S_k : \langle \mathbf{w}, \Psi(\Phi_k, S_k) \rangle - \langle \mathbf{w}, \Psi(\Phi_k, S) \rangle \geq \Delta(S, S_k) - \xi_k. \end{aligned} \quad (7.45)$$

Let \mathbf{w}', ξ' be the solution to

$$\begin{aligned} \min_{\mathbf{w}', \xi'} \quad & \frac{1}{2} \|\mathbf{w}'\|^2 + \eta C \sum_{k=1}^n \xi'_k \\ \text{s.t. } & \forall k, S \neq S_k : \langle \mathbf{w}', \Psi(\Phi_k, S_k) \rangle - \langle \mathbf{w}', \Psi(\Phi_k, S) \rangle \geq \eta \Delta(S, S_k) - \xi'_k. \end{aligned} \quad (7.46)$$

Then

$$\mathbf{w}' = \eta \mathbf{w} \quad (7.47)$$

$$\boldsymbol{\xi}' = \eta \boldsymbol{\xi}. \quad (7.48)$$

Proof. Substituting for \mathbf{w}' and $\boldsymbol{\xi}'$ in (7.46),

$$\min_{\mathbf{w}, \boldsymbol{\xi}} \frac{1}{2} \|\eta \mathbf{w}\|^2 + \eta C \sum_{k=1}^n \eta \xi_k \quad (7.49)$$

$$s.t. \quad \forall k, S \neq S_k : \langle \eta \mathbf{w}, \Psi(\Phi_k, S_k) \rangle - \langle \eta \mathbf{w}, \Psi(\Phi_k, S) \rangle \geq \eta \Delta(S, S_k) - \eta \xi_k$$

and dividing the objective by η^2 and the constraints by η we obtain the desired result. \square

7.8 Conclusion

We have presented a unified learning framework for recovering semantically meaningful geometric models from single and multiple views of a scene. We have chosen to work with the indoor Manhattan class in order to leverage the simple parametrisation and efficient inference algorithm made possible by this hypothesis class. Our approach to learning performs a single optimisation with respect to a clearly defined loss function. Experiments show our system out-performing the approach of the previous chapter in the multiple-view setting (by a large margin) and on one metric in the single-view setting.

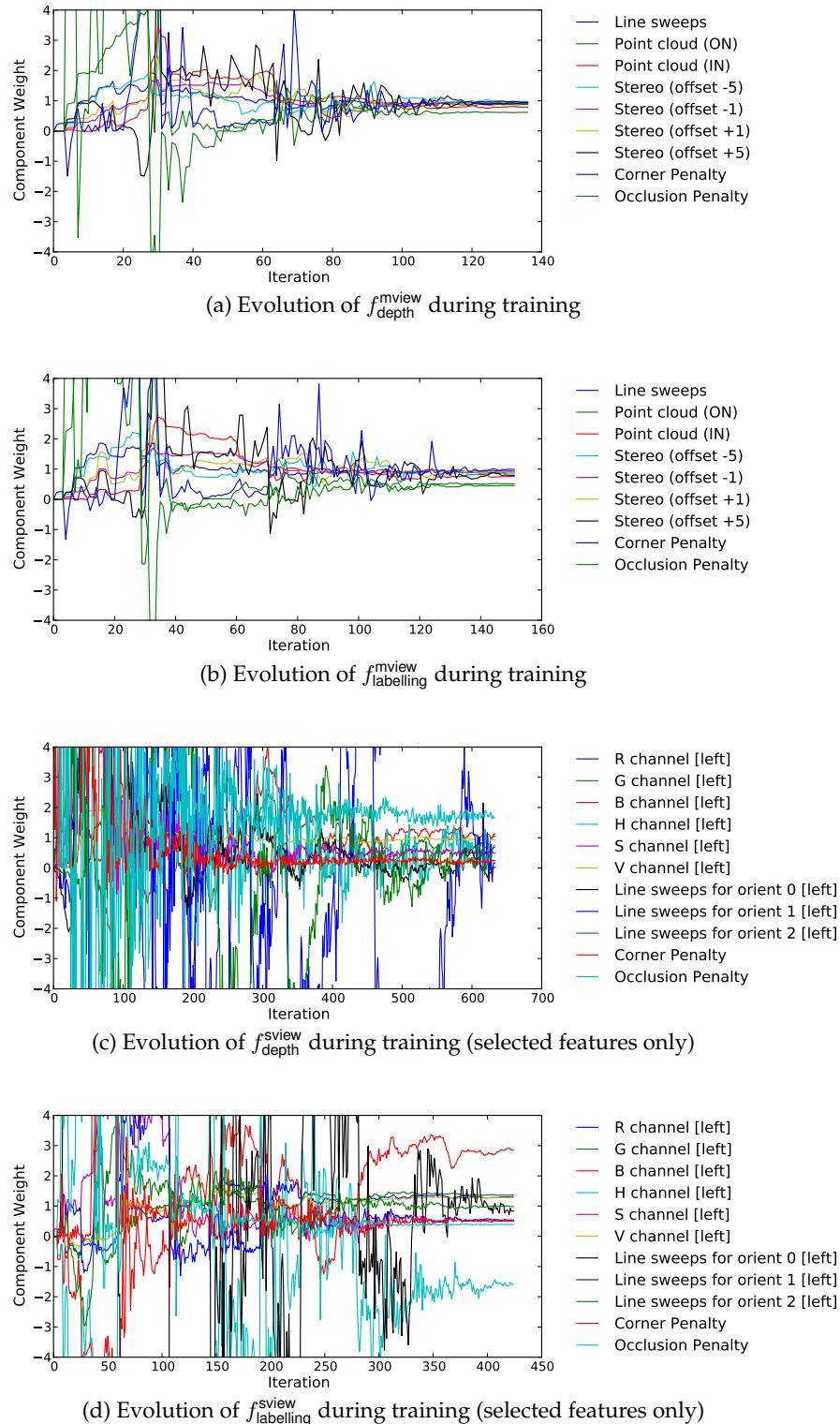


Figure 7.2: Evolution of model weights w during training of each predictor. See main text for description of the four predictors we trained.

8

Conclusion

In this thesis we have presented ideas for inference of high-level scene structure in the context of a moving observer. Drawing inspiration from scene understanding ideas that have traditionally been the purview of single-view computer vision, we have demonstrated a number of promising approaches towards multiple-view scene understanding. In this concluding chapter we will briefly review the key results and present ideas for future work.

8.1 Key Results

Chapter 3 presented textons as a simple and flexible basis for inferring contextual variables such as scene categories. Building on ideas from texture analysis, we presented a probabilistic model and accompanying inference procedure, and evaluated our approach on two classification problems. In both cases our system out-performed the Gist descriptor of Torralba *et al.*.

In Chapter 4 we gave a formal account of the indoor Manhattan model — the first comprehensive analysis that we are aware of in the literature (though not the first description *per se*). This chapter did not present any empirical results; its primary contribution was the development of the “vertex” and “seam” representations, and the decomposability property of indoor

Manhattan scenes.

Next we discussed recovery of the Manhattan world’s orientation with respect to a camera, given multiple calibrated views of a scene. We showed that reasoning from photometric information rather than geometric information, and integrating multiple images into a joint optimisation improved substantially upon existing approaches. Our algorithm built upon ideas from single-view vanishing point estimation; our contribution was in showing how to extend this to multiple calibrated views.

Chapter 6 focussed on inferring indoor Manhattan scene structure from single and multiple views of a scene. We presented a probabilistic model relating Manhattan scene structure to three sensor modalities, and showed that the likelihoods for all three sensor modalities can be written in terms of payoff functions. Beyond the specific model that we proposed, this approach suggests a general strategy by which further sensor modalities can be integrated with our model without modifying the inference algorithm. The other contribution of this chapter was the dynamic programming algorithm to solve the payoff optimisation problem. We showed that a decomposition into sub-problems permitted an efficient and exact solution to both MAP and ML inference. Experiments in this chapter showed our system substantially out-performing two other systems.

Finally, Chapter 7 described an approach to learning indoor Manhattan models from training examples. We described a structured prediction learning algorithm for the indoor Manhattan hypothesis class, and showed that the two required inference problems were solvable by the dynamic programming algorithm of Chapter 6. Among learning approaches to geometric scene context, our contribution is among a small number that implement a single optimisation with respect to a clearly defined loss function. We showed a clear improvement over the system proposed in Chapter 6.

A theme of this thesis has been the use of geometry for scene understanding purposes. We have shown how to extract semantically meaningful scene structure using ideas drawn from both the geometry literature and the scene understanding literature. The ideas from geometry allowed

us to decompose our reconstructions into a representation amenable to dynamic programming, and the ideas from single-view scene understanding allowed us to connect our hypothesis class to observed image features.

8.2 Future Work

In this section we discuss directions for future work.

8.2.1 Applications of Indoor Manhattan Models

Perhaps the clearest direction for future work is to apply the indoor Manhattan model to the semantic inference tasks for which it was originally designed, namely scene category recognition and contextual object detection.

Scene category recognition is the problem of classifying scenes into categories such as “bedroom”, “bathroom”, or “office”. There is a significant literature on this problem for the single-view case (as reviewed in Chapter 2), but little effort has been directed at leveraging multiple-view geometry to this end. Since humans greatly out-perform computers at this task, and humans appear to use various notions of geometry to achieve their high performance, the use of geometry for scene categorisation appears to be a particularly compelling direction for future work.

Indoor Manhattan models could be leveraged for scene categorisation in a number of ways. At the simplest level, one could recover an indoor Manhattan representation for a query scene and compute a series of geometric features to capture the shape of the scene (ratios of length, breadth, and height, for example) and the position of the camera within it. These could be appended to a vector of photometric features (such as the many proposed in the single-view literature), then a classic multiple-label classifier could be trained to distinguish the various categories. A more sophisticated approach might link the recovered geometry with photometric information more closely. A typical photo captured by a camera located close to the ceiling

would be expected to appear different to that of a camera located close to the floor, even within one scene category. Photometric approaches rely on a classifier to learn such variations automatically, but with an indoor Manhattan model at hand one could model such variations explicitly. For example, one could build up separate appearance models of the floor, wall, and ceilings, using known scene geometry and camera position. Finally, the possibility exists of improving upon simple Markovian models to integrate information over time. A system that recognises when the camera moves between rooms (such as by identifying doorways) could segment a video sequence according to which frames are situated in which rooms. Such a system could perform data association between observations and room categories much more accurately than under a hidden Markov model with a homogeneous transition function such as that used in [61].

The second application to which we intend to apply indoor Manhattan models is object recognition. Contextual information appears to greatly improve object recognition (see Chapter 2 for a review), and the prospect of leveraging the high-level geometric information captured by the indoor Manhattan for this purpose seems promising. A simple approach would be to append geometric features derived from the recovered indoor Manhattan model to traditional appearance features. These geometric features might include the distance of the object from the floor plane (as a percentage of the distance from floor to ceiling), the distance from the closest wall, shape features of the environment, and so on. However, once again the possibility exists of using Manhattan geometry in a more sophisticated way. For example, one could integrate observations from different viewpoints to improve the estimate of an object’s identity. The floorplan provided by an indoor Manhattan model could provide a natural basis for integrating such observations.

Several robotics applications may also benefit from the ability to recover Manhattan structure. Path planning requires knowledge of “traversable” surfaces in the environment as well as boundaries that will obstruct movement. The floorplan provided by an indoor Manhattan model seems a natural representation of such information. One might divide the floor plane into a series of cells and separately estimate traversability of each. The occupancy grid estima-

tion might benefit substantially from knowing the shape of the environment about each cell.

A related to path planning is the problem of *active exploration*, in which an agent must plan a path to find an object or location of interest as quickly as possible. Typically this problem is approached from the perspective of information gain rather than shortest path. One might use Manhattan models as a basis for active exploration by reasoning explicitly in terms of rooms and corridors. For example, if a robot seeking a teapot recognises that it is in a bathroom then it could move to a different room immediately, not bothering to continue searching the current environment.

8.2.2 Relaxing Geometric Constraints

A common criticism levelled against indoor Manhattan models is the strong geometric constraints that one must make. We have argued that even in their present form, indoor Manhattan models are surprisingly versatile. Nevertheless, there are several promising approaches to relaxing these assumptions. The difficulty, of course, is finding relaxations that retain the fast inference algorithm of Chapter 6, which is based on the left-to-right decomposability of indoor Manhattan models.

A first relaxation is to permit walls oriented in more than two directions. This could be accomplished by identifying vanishing points on the horizon line corresponding to additional wall orientations, then extending the domain of the orientation variable in the inference algorithm to account for these. These vanishing points could be found either by clustering lines with respect to their intersection at the horizon, or by modifying the rotation estimation algorithm to jointly estimate these wall orientations together with the scene orientation.

A second relaxation would be to permit walls that do not project to straight lines on the floor plane. This could be accomplished in a range of ways depending on the restriction that is chosen to replace straightness. One could parametrise walls in pixel coordinates and simply allow walls to change direction arbitrarily. While this would permit a wide range of environments to be represented, it is not clear that this is desirable since it removes the implicit penalisa-

tion of highly complex models. A different approach would be to permit parametric curves so that pillars and circular wall segments could be represented. This could be accomplished without expanding the state-space of the dynamic programming at all; it would simply entail integrating the parametric family into the maximisation in (6.75).

Another promising direction for relaxation would be to target outdoor areas by removing the assumption of a fixed ceiling plane. Many built-up areas could be represented by a horizontal ground plane and series of vertical facades. The key difference between indoor environments is that building facades typically have varying height. By adding a height parameter to the state space one could jointly estimate the orientation and height of each facade using only a slight modification to the algorithm presented in Chapter 6.

8.2.3 Extensions Of The Probabilistic Model

In addition to the geometric extensions suggested above, there are several promising ways in which the probabilistic models underlying Manhattan structure recovery could be extended. In Chapter 7 we considered learning from labelled examples, but other types of background information could also be helpful for fixing parameters. For example, a database of architectural floorplans could be leveraged to learn about the architectures an agent is most likely to encounter, and hence to form a more accurate prior over building structures. This would require a Bayesian model connecting architectural floorplans to our prior over building structures

In a different direction, one might consider *marginalizing* over indoor Manhattan reconstructions for certain purposes rather than just using a single MAP estimate. That is, for the purpose of, say, scene categorisation or object search, we would consider *all* possible indoor Manhattan reconstructions and their implications for the task at hand, weighted by their plausibility under the model presented in Chapter 6. In this case we would see the reconstruction as a latent variable. In many cases such a marginalisation is intractable, but in the case of indoor Manhattan models there is a promising approach by which the decomposition of Chapter 6 might be leveraged to perform exact marginalisation for a large family of models.

Finally, there is much room for integrating the estimation of indoor Manhattan models over time. Currently our inference procedure is a batch operation that re-estimates the model from scratch at each time step. Extending this to a recursive estimation framework would allow us to integrate new information in a more principled way, and avoid discarding the computational effort invested at previous time steps. Recursive state estimators maintain a distribution over an underlying hypothesis class that can be efficiently updated from new evidence. A filter for indoor Manhattan environments might represent hypotheses in the seam representation discussed in chapter Chapter 4 and summarise a distribution over this space using a payoff matrix as discussed in Chapter 6. Updating the payoff matrix over time requires integrating image evidence from new viewpoints, but we have already presented the basic geometry that such updates would require during our discussion of stereo features in Chapter 6. Significant work would be required to deal with the occlusion artefacts that integrating information over long periods would introduce.

8.3 Final Remarks

The meeting between geometric ideas from structure–from–motion and scene understanding ideas from single view computer vision seems to hold promise as an exciting research direction over the coming years. This will be driven as much by the new sensor modalities discussed in the introduction as by the maturity of both existing SLAM systems and scene understanding as separate areas of study. We look forward to seeing this unfold.

Bibliography

- [1] A. P. DEMPSTER, N. M. L., AND RUBIN, D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society 39*, 1 (1977), 1–38.
- [2] BAKIR, G. H., HOFMANN, T., SCHLKOPF, B., SMOLA, A. J., TASKAR, B., AND VISHWANATHAN, S. V. N. *Predicting Structured Data*. MIT Press, 2007.
- [3] BARINOVA, O., KONUSHIN, V., YAKUBENKO, A., LEE, K., LIM, H., AND KONUSHIN, A. Fast automatic single-view 3-d reconstruction of urban scenes. In *ECCV* (2008), pp. 100–113.
- [4] BARNARD, S. Interpreting perspective images. *Artificial Intelligence 21*, 4 (Nov. 1983), 435–462.
- [5] BLASCHKO, M., AND LAMPERT, C. Learning to localize objects with structured output regression. *Computer Vision ECCV 2008* (2008), 2–15.
- [6] BROSTOW, G. J., SHOTTON, J., FAUQUEUR, J., AND CIPOLLA, R. Segmentation and recognition using structure from motion point clouds. In *Proc 10th European Conf on Computer Vision* (Berlin, Heidelberg, 2008), Springer-Verlag, pp. 44–57.
- [7] BUSCHKA, P., AND SAFFIOTTI, A. A virtual sensor for room detection. In *Intelligent Robots and System, 2002. IEEE/RSJ International Conference on* (2002), vol. 1, pp. 637–642 vol.1.
- [8] CANNY, J. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-8*, 6 (Nov. 1986), 679–698.
- [9] CORTES, C., AND VAPNIK, V. Support-vector networks. *Machine Learning 20*, 3 (1995), 273–297.
- [10] COUGHLAN, J., AND YUILLE, A. Manhattan world: compass direction from a single image by bayesian inference. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on* (1999), vol. 2, pp. 941–947 vol.2.
- [11] CRIMINISI, A. *Accurate visual metrology from single and multiple uncalibrated images*. Springer-Verlag New York, Inc., New York, NY, USA, 2001.
- [12] CUMMINS, M., AND NEWMAN, P. Fab-map: Probabilistic localization and mapping in the space of appearance. *Int. J. Rob. Res. 27*, 6 (2008), 647–665.
- [13] DAVID CHANGSOO LEE, M. H., AND KANADE, T. Geometric reasoning for single image structure recovery. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2009).
- [14] DELAGE, E. A dynamic bayesian network model for autonomous 3d reconstruction from a single indoor image. *Computer Vision and Pattern ...* (2006).

- [15] DENIS, P., ELDER, J., AND ESTRADA, F. Efficient edge-based methods for estimating manhattan frames in urban imagery. *Computer VisionECCV 2008* (2008), 197–210.
- [16] EVERINGHAM, M., VAN GOOL, L., WILLIAMS, C. K. I., WINN, J., AND ZISSERMAN, A. The PASCAL Visual Object Classes Challenge 2009 (VOC2009) Results. <http://www.pascal-network.org/challenges/VOC/voc2009/workshop/index.html>.
- [17] FEI-FEI, L. A Bayesian hierarchical model for learning natural scene categories. In *Proc 23rd IEEE Conf on Computer Vision and Pattern Recognition* (2005), pp. 524–531.
- [18] FELZENZWALB, P., GIRSHICK, R., MCALLESTER, D., AND RAMANAN, D. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2010).
- [19] FELZENZWALB, P. F., AND HUTTENLOCHER, D. P. Efficient graph-based image segmentation. *International Journal of Computer Vision* 59 (2004), 2004.
- [20] FLINT, A., MEI, C., REID, I., AND MURRAY, D. A dynamic programming approach to reconstructing building interiors. In *Proc 12th European Conf on Computer Vision* (2010).
- [21] FLINT, A., MEI, C., REID, I., AND MURRAY, D. Growing semantically meaningful models for visual slam. In *Proc 28th IEEE Conf on Computer Vision and Pattern Recognition* (2010).
- [22] FLINT, A., REID, I., AND MURRAY, D. Learning textons for real-time scene context. In *Proc 27th IEEE Conf on Computer Vision and Pattern Recognition* (2009).
- [23] FLINT, A., REID, I., AND MURRAY, D. Manhattan scene understanding using monocular, stereo, and 3d features. In *Proc 13th IEEE Int Conf on Computer Vision* (2011).
- [24] FORSYTH, D., AND PNCE, J. *Computer vision: A modern approach*. Prentice Hall, 2002.
- [25] FURUKAWA, Y., CURLESS, B., SEITZ, S., AND SZELISKI, R. Manhattan-world stereo. *CVPR* 0 (2009), 1422–1429.
- [26] GUPTA, A., EFROS, A., AND HEBERT, M. Blocks world revisited: Image understanding using qualitative geometry and mechanics. In *Proc 12th European Conf on Computer Vision* (2010), vol. 6314, Springer Berlin / Heidelberg, pp. 482–496.
- [27] HARTLEY, R. I., AND ZISSERMAN, A. *Multiple View Geometry in Computer Vision*, second ed. Cambridge University Press, 2004.
- [28] HEDAU, V. Recovering the spatial layout of cluttered rooms. *Computer vision, 2009 IEEE ...* (2009).
- [29] HEDAU, V. Thinking inside the box: Using appearance models and context based on room geometry. *Computer VisionECCV 2010* (2010).
- [30] HEDAU, V., HOIEM, D., AND FORSYTH, D. Recovering the spatial layout of cluttered rooms. In *ICCV* (2009), vol. 2.
- [31] HEITZ, G., AND KOLLER, D. Learning spatial context: Using stuff to find things. In *Proc 10th European Conf on Computer Vision* (2008), pp. 30–43.
- [32] HOIEM, D., EFROS, A. A., AND HEBERT, M. Automatic photo pop-up. *ACM Transactions on Graphics* 24, 3 (2005), 577.
- [33] HOIEM, D., EFROS, A. A., AND HÉBERT, M. Geometric context from a single image. In *Proc 10th IEEE Int Conf on Computer Vision* (2005), pp. 654–661.

- [34] HOIEM, D., EFROS, A. A., AND HÉBERT, M. Putting objects in perspective. In *Proc 24th IEEE Conf on Computer Vision and Pattern Recognition* (2006), pp. 2137–2144.
- [35] HUYNH, D. Q. Metrics for 3D Rotations: Comparison and Analysis. *Journal of Mathematical Imaging and Vision* 35, 2 (June 2009), 155–164.
- [36] JEBARA, T. Images as bags of pixels. In *Proc 9th IEEE Int Conf on Computer Vision* (2003), pp. 265–272.
- [37] JULESZ, B. Textons, the elements of texture perception, and their interactions. *Nature* 290 (Mar. 1981), 91–97.
- [38] KANATANI. *Geometric Computation for Machine Vision*. Oxford University Press, USA, 1993.
- [39] KLEIN, G., AND MURRAY, D. Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)* (Nara, Japan, November 2007).
- [40] KOLMOGOROV, V., AND ZABIH, R. What energy functions can be minimized via graph cuts? *PAMI* (2002).
- [41] KOSECKÀ, J., AND ZHANG, W. Video compass. In *Proc 7th European Conf on Computer Vision* (2002), vol. 2353 of *Lecture Notes in Computer Science*, Springer, pp. 4: 476–490.
- [42] LEJEUNE-DIRICHLET, P. *Vorlesungen ber Zahlentheorie*, 2 ed. Vieweg, Braunschweig, 1863.
- [43] LI, Y., AND HUTTENLOCHER, D. Learning for stereo vision using the structured support vector machine. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on* (2008), IEEE, pp. 1–8.
- [44] LINDEBERG, T. *Scale-Space Theory in Computer Vision*. Kluwer Academic Publishers, Norwell, MA, USA, 1993.
- [45] LOWE, D. Object recognition from local scale-invariant features. pp. 1150–1157.
- [46] MACKAY, D. J. C. *Information Theory, Inference, and Learning Algorithms*, vol. 22. Cambridge University Press, 2003.
- [47] MALIK, J., BELONGIE, S., SHI, J., AND LEUNG, T. Textons, contours and regions: cue integration in image segmentation. In *Proc 7th IEEE Int Conf on Computer Vision* (1999), vol. 2, pp. 918–925.
- [48] MIRZAEI, F. M., AND ROUMELIOTIS, S. I. Optimal estimation of vanishing points in a Manhattan world, 2011.
- [49] MOZOS, O., STACHNISS, C., AND BURGARD, W. Supervised learning of places from range data using adaboost. In *Proc 2005 IEEE Int Conf on Robotics and Automation* (2005), pp. 1730–1735.
- [50] PARZEN, E. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics* 33, 3 (1962), 1065–1076.
- [51] POSNER, I., SCHROETER, D., AND NEWMAN, P. Online generation of scene descriptions in urban environments. *Robot. Auton. Syst.* 56, 11 (2008), 901–914.
- [52] REN, X., AND MALIK, J. Learning a classification model for segmentation. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on* (Oct. 2003), pp. 10–17 vol.1.

- [53] ROBERT E. SCHAPIRE, YOAV FREUND, P. B., AND LEE, W. S. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics* 26, 5 (1998), 1651–1686.
- [54] ROBERTS, L. *Machine perception of 3-d solids*. PhD Thesis, 1965.
- [55] SAXENA, A., SUN, M., AND NG, A. Y. Make3d: Learning 3d scene structure from a single still image. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31, 5 (2009), 824–840.
- [56] SCHARSTEIN, D., AND SZELESKI, R. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. In *IJCV* (2001).
- [57] SCHINDLER, G., AND DELLAERT, F. Atlanta world: An expectation maximization framework for simultaneous low-level edge grouping and camera calibration in complex man-made environments.
- [58] SHUFELT, J. Performance evaluation and analysis of vanishing point detection techniques. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 21, 3 (Mar 1999), 282–288.
- [59] STACHNISS, C., MARTNEZ-MOZOS, O., ROTTMANN, A., AND BURGARD, W. Semantic labeling of places. In *in Proceedings of the International Symposium on Robotics Research* (2005).
- [60] TASKAR, B., KLEIN, D., COLLINS, M., KOLLER, D., AND MANNING, C. Max-margin parsing. In *Proc. EMNLP* (2004), pp. 1–8.
- [61] TORRALBA, A. Contextual priming for object detection. *International Journal of Computer Vision* 53, 2 (2003), 169–191.
- [62] TSOCHANARIDIS, I., HOFMANN, T., JOACHIMS, T., AND ALTUN, Y. Support vector machine learning for interdependent and structured output spaces. *ICML* 36, 4 (2004), 104.
- [63] VAPNIK, V. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.
- [64] VAPNIK, V. N. *Statistical Learning Theory*, vol. 2 of *Wiley Series on Adaptive and Learning Systems for Signal Processing, Communications and Control*. Wiley-Interscience, 1998.
- [65] VARMA, M., AND ZISSERMAN, A. A statistical approach to texture classification from single images. *International Journal of Computer Vision* 62, 1-2 (2005), 61–81.
- [66] VASUDEVAN, S., GÄCHTER, S., NGUYEN, V., AND SIEGWART, R. Cognitive maps for mobile robots—an object based approach. *Robot. Auton. Syst.* 55, 5 (2007), 359–371.
- [67] WERNER, T., AND ZISSERMAN, A. New Techniques for Automated Architectural Reconstruction from Photographs. 541–555.
- [68] YANG, W., TRIGGS, W., DAI, D., AND XIA, G.-S. Scene Segmentation with Low-dimensional Semantic Representations and Conditional Random Fields. *EURASIP Journal on Advances in Signal Processing* 2010 (2010), 1–14.
- [69] ZHU, S., GUO, C., WANG, Y., AND XU, Z. What are textons? *IJCV* (2005), 121–143.