Contextual Reasoning for Egocentric Vision

Alexander John Flint
Exeter College

Robotics Research Group
Department of Engineering Science
University of Oxford

Hilary 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
Hilary Term 2012

Contextual Reasoning for Egocentric Vision

Abstract

This thesis describes stuff aplenty.

Contents

Acknowledgements

This thesis is the culmination of many year's work and would not have been possible without the help and support of many people.
First and foremost I would like to thank my supervisor, Dr X, for his mindless support and useless proof-reading. Blah, gibber ...
Many of the ideas presented are the outcome of discussions with members of the Active Vision Lab and beyond. In particular I owe much to X and Y who ...
My research was funded by a ...
Throughout this work I have enjoyed the unfailing support of family ...

Notation

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

- **2D and general Vectors** are written in lower case bold: $\mathbf{a}$, and a unit vector with a hat $\hat{\mathbf{a}}$. Where important to differentiate between homogeneous and non-homogeneous vectors, the latter will appear as $\tilde{\mathbf{a}}$. Vectors are usually column vectors, with elements specified by subscript index (eg. $\mathbf{x} = (x_1, x_2)^\top$).
- **3D Vectors** are written in upper case bold: $\mathbf{A}$, with similar conventions to 2D vectors.
- **Matrices** are written in teletype: $\mathtt{A}$, and may have size indicated, $\mathtt{A}_{3\times4}$. Where a matrix is square $\mathtt{A}_3$ is a 3×3 matrix. The entry in the ith row and jth column of the matrix is $\mathtt{A}_{ij}$.
- **Tensors** are written in bold calligraphic: $\boldsymbol{\mathcal{T}}_i^{jk}$,$\boldsymbol{\mathcal{Q}}^{ijkl}$.
- **Quaternions** are written as $\mathring{\mathrm{a}}$.
- **Projective Equality** (see Appendix **??**) is denoted $\overset{P}{=}$.

Abbreviations

The following abbreviations have been adopted:

ADC	analogue to digital converter
SFM	structure from motion
ZOH	zero order hold

1

Introduction

Copied? --> Humans are visual creatures and as such our world is built up around visual cues. The reason we use written street signs rather than, say, smells or sounds to navigate the road network is that our biological vision system has both higher fidelity and larger bandwidth than our other senses. In order for computers to parse our world we have the choice of imbuing computers with vision–processing systems or re–signposting (in the most general sense) our world with cues more amenable to automatic extraction. Computer vision could be seen as the endeavour to acomplish the former route.

This thesis was originally motivated by the specific application area of augmented reality (AR). In the broadest possible terms, AR is the idea of teaching computers to become fluent in the visual experience of a human, rather than having humans interact with computers on the computer's terms. The great vision of AR is to have computers parse, track, and understand the visual field–of–view of a human in real–time, and seamlessly overlay text and graphics for the benefit of the wearer.

The problem addressed in this thesis is the development of 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 an architectural blueprint or a city roadmap, complete with annotations and symbols, rather than, say, the level of a set of points floating in space. Inspired originally by the application of these ideas to AR we will pay special attention to continuous video streams (rather than individual frames) and to frame–rate processing capacity (rather than batch processing). We will refine this problem statement later in this chapter, then again, quantitatively, in later chapters.

1.1 Motivation

The research presented in thesis has been motivated from a variety of angles. An early motivation was the extension of visual tracking and mapping systems to high–level reasoning about the visual environment. Simultaneous localisation and mapping (SLAM) — the problem of

precisely mapping a novel environment whilst simultaneously locating oneself within it — has seen terrific progress over the past several decades. In particular, visual SLAM, in which the sensing apparatus is a camera, has taken great strides in the last ten years. The typical experiments expected of a visual SLAM paper in a top conference have moved from a single room to an entire city; real–time processing is now expected rather than impressive; and one can even expect to see system that reconstruct every pixel in a video stream rather than just a set of interest points.

Visual SLAM is a challenging requirement for a convincing AR system, since to place augmentations contextually within the wearer's field–of–view, an AR system must first be able to locate itself pricesly and quickly within the environment. It is neither surprising nor objectionable, therefore, that so much of the effort towards AR has focussed on visual SLAM; but now that high–performance SLAM systems *are* available, the question naturally presents itself: "are we done yet?" Or, in other words, does a "SLAM oracle" that precisely locates the camera at every point in time completelty solve the AR problem?

The answer I will argue for in this thesis is that no, SLAM is not the end of the story. While SLAM is certainly a central and important problem, there remain interesting, challenging, and general computer vision problems to solve before AR can be fully realised. For example, the relevance of some piece of information to the wearer varies with the type of environment within which the wearer is moving. 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 a house, the visual augmentations most likely to be desirable while 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 room. A human would not search for a lost set of keys on the ceiling first, not outside a window, but to make this inference one must first identify the orientation of the floor and ceiling and the position of room boundaries. These problems are not immediately solved by the ability to locate oneself within an arbitrary 3D coordinate frame — though that ability almost certainly helps, as we will show.

Although this thesis was originally motivated from the perspective of real–time AR, I found much inspiration from the single–view computer vision community, 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 much shorter history within the multiple–view and SLAM literature. In the single–view context, scene understanding is necessarily cast in terms of photometric sensor data. The question that naturally presented itself during the development of this research was, "can we do scene understanding using multiple views of a scene?".

The enormity of the scene understanding literature is testament to the number of different ways that the problem can be cast. Everything from scene category recognition to semantic segmentation and single–view reconstruction has been cast within the circle of scene understanding. The extension of scene understanding to multiple views opens up perhaps an even

broader multitude of approaches, so it would be foolish to try to present a definitive account of multiple–view scene understanding within a single thesis. Instead, I have carved off a piece of the problem that I believe to be large enough to be interesting, small enough to be tractible, and hopefully insightful enough to be instructive for future research in this new and exciting domain. The piece of the puzzle that I have chosen is the extraction of high–level geometric information within indoor–type environments, with relevance to scene categorisation and object localisation. I have worked with both pure photometric models and joint photometric/geometric models.

A final motivation that bears mentioning is the exciting acceleration over the past several years of new platforms suitable for computer vision applications. Firstly, video on the internet has become increasingly important over the past several years, heightening the relevance of computer vision systems that process video streams rather than single images. It is perhaps startling to be reminded that the popular internet video website, YouTube, was founded a mere 6 years ago in 2006, and did not gain widespread popularity until 2008. Secondly, mobile phones have become enormously more powerful as computing devices and the cameras now ubiquitously attached to them have improved in quality by several orders of magnitude. The reality and relevance of algorithms for understanding and providing feedback on video streams in real–time have never been clearer. Finally, depth sensing cameras such as the Kinect have made an explosive entry into the consumer hardware market and provide a compelling case for moving beyond

1.2 The Need For Geometry

We have thus far made the argument that AR needs scene understanding, but does scene understanding need geometry? Perhaps we could completely discard structure–from–motion and build AR systems using the purly photometric approaches of single–view scene understanding. This seems to me also to be an unreasonably extreme position. Much ~~amount of~~ work has been dedicatd to understanding the geometry of cameras and multiple viewpoints; this thesis tries to show that this work can indeed be leveraged in pursuit of solutions to semantic–level computer vision problems.

The presentation of this thesis follows, roughly, the chronological order of the research itself. Chapter 3 discusses an early phase in which I pursued an "appearance–only" model for drawing high–level inferences from video input. While I was impressed by how far this approach could be pushed, I ultimately decided some ~~coarse~~ knowledge of the geometric structure of the environment was needed to answer the semantical–level questions I was pursuing, and as a result the later chapters are conspicuously geometry–heavy.

The distinction between a "geometry–less" and a "geometry–laden" approach is of course a soft and conceptual one: In either case one is, at the end of the day, drawing probabilistic inferences from ~~one or more input~~ arrays of pixel intensities. The distinction is ~~over of~~ how those inputs are combined, which conditional independences are assumed, how a hypothesis

class is formulated, and so on.
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 geometric representation reflects this. We seek a representation which is both suited to semantic–level scene understanding, and general enough to encompass an interesting range of environments. A paper by David Lee [DCLK09] in 2009 provided the pivotal insight that the so–called "indoor Manhattan model" provided an excellent trade–off between generality and concision.
Under the indoor Manhattan assumption, the world is composed of a floor plane, ceiling plane, and a set of vertical walls that meet at vertical edges. While a strong assumption, this hypothesis class can represent exactly or approximately a surprisingly broad range of indoor environments, as well as some outdoor environments dominated by human structures. Furthermore, the location of the floor, walls, and ceiling are among the geometric cues most relevant to the semantic–level scene understanding questions that originally motivated this thesis. As an illustration of this, consider the image shown in figure TODO. Despite the low information content of the image (it consists of just three colours and a set of lines defined by a mere handful of vertices), it seems that an average human would be able to make some reasonable inferences about this environment. For example, we claim that a human would be able to say *something* in response to the following questions (in order of increasing uncertainty).

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?
4. Is this an office or a house?
5. What is the absolute scale of the environment?

~~Of course, counter–examples could be constructed to render the answer to any of these questions arbitrarily erronous.~~ We do not claim that any of these questions could be answered with certainty, only that the information in figure TODO would provide ~~non–trivial~~ evidence in support of answers to the questions above.

1.3 Our Approach

In this thesis we present two approaches to extending the competence of AR system to semantic–level scene understanding. The first is a purely photometric approach based on textons. We consider a video sequence to be a simple time–varying signal, ignoring geometric constraints. We associate each pixel with a representative texton, which are exemplars of image patch clusters identified in a training phase. We show how textons can help answer two semantic–level questions: scene recognition and object search. Despite the simplicity of this approach, we show that this appearance–only method is surprisingly effective for these questions.

To push this approach further, the second part of this thesis shows how to integrate coarse geometric information including major surfaces and scene boundaries. We argue that the recently proposed "indoor Manhattan model" provides a compelling trade–off between robustness, flexibility, and efficiency. Under this model the environment is represented by a small number of major surfaces such as the floor, wall, and ceiling of an indoor environment. There is little prior work on this model (just 2 published conference papers besides our own at the time of writing), so we dedicate a full chapter to defining the model formally, working through some simple corollaries, and discussing two important parametrisations. We then turn to the problem of probabilistic reasonining within this hypothesis class, which occupies the final three chapters. In order, we define a probabilistic model, reduce inference in this model to a particular class of optimisation problem, solve this optimisation problem using dynamic programming, then present a discriminative training regime to fix the free parameters.
The overall approach to the work in this thesis is basic on three underlying assumptions. First, we believe that probability is the appropriate language for reasoning from image evidence. As far as possible we try to present graphical models that make clear our assumptions before discussing any algorithmic details. Second, we believe that it is worth leveraging the considerable literature on camera geometry wherever possible. Finding minimal parametrisations pays off as robustness and speed of the final algorithm. Finally, we see computational efficiency as a crucial consideration that rightly affects both the choice of hypothesis class and the design of inference algorithms.
Overall - Bayesian - "proof in the pudding": experiments justify our modelling approximations - the hypothesis we test: this model will perform well in practice - mind projection falacy? - separate modelling from inference - but choose model judiciously - relationship between geometry and probability - define a hypothesis class using hard geometric constraints - connect with sensor data using probabilistic models - devise inference algorithms using algorithmic insight - mind projection falacy: the wall is not soft, but our estimate of its position is - inference from "generative" standpoint - learning from "discriminative" standpoint - speed is important - cannot really separate performance from speed - concentrate on algorithmic complexity not constant-time speedup

1.4 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 then 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 to scene categories and object locations.
We then begin the presentation of our work on indoor Manhattan models. Chapter 4 presents

the model formally, and covers ~~all~~ the geometric insights necessary for the later chapters.
Chapter 5 ~~switches into the language of probability. We~~ present graphical models relating sensor data to the geometric primitives we wish to extract. We give separate probabilistic models for photometric features, stereo data, and point clouds. We then show how to combine any combination of these into a joint model, allowing our system to be easily tailored to many different contexts.
Chapter 6 presents a dynamic programming algorithm that solves the inference problem for the model presented in chapter 5. It is both fast and exact, making a compellinge alternative to previous approaches that fitted neither of these descriptions. We present compelling comparisons with previous approaches.
Chapter 7 presents a learning routine in which we learn to reconstruct Manhattan environments from training examples. We cast the learning problem in a discriminative framework and use state–of–the–art tools from the structured prediction literature to achieve polynomial–time training.
Finally, chapter 8 summarises key results, discusses their ramifications, and suggests directions for future work.

2

Review

2.1 Introduction

Need clear connection between "semantially meaningful geometry" and "context"

In this section we provide a review of the literature relevant this thesis. We begin with a review of simultaneous localisations and mapping (SLAM) systems, which is relevant because the research presented in the remainder of this report involves reasoning processes that utilise SLAM systems to provide known camera poses.

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. Following this we examine the more detailed literature on context in computer vision. This area is of great interest since many of the techniques may be transferable to our setting.

Re-write given emphasis from introduction

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 of the 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.* [Tor03] 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 [Tor03] they show how the location and scale of objects can be predicted from the gist vector. The Expectation–Maximisation algorithm [APDR77] 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 [Vap95] 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 [APDR77]. 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 model parameters are maximised with respect to the distribution computed in the first step (the M–step). These steps are repeated until convergence.

Semantic Segmentation

TODO: CamVid

Scene Classification

Fei–Fei and Perona [Fei05] 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

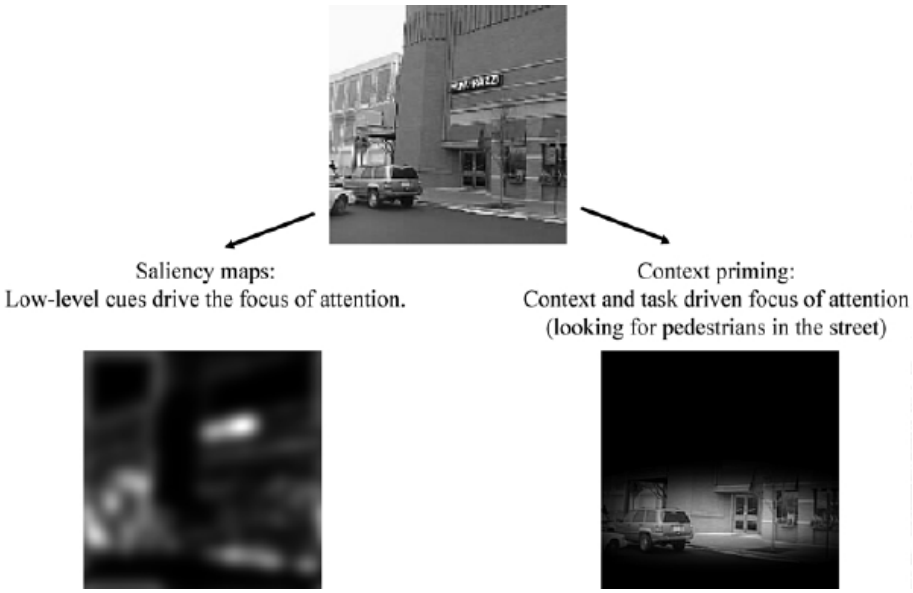


Figure 2.1: Torralba *et al.* [Tor03] is able to improve upon traditional interest point detectors by learning the relationship between gist features and likely object locations. Image taken from [Tor03].

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 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 [Low99] 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 sizes, SIFT builds a scale–space representation of the image [Lin93] 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. SIFT is now used for many tasks throughout computer vision.

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

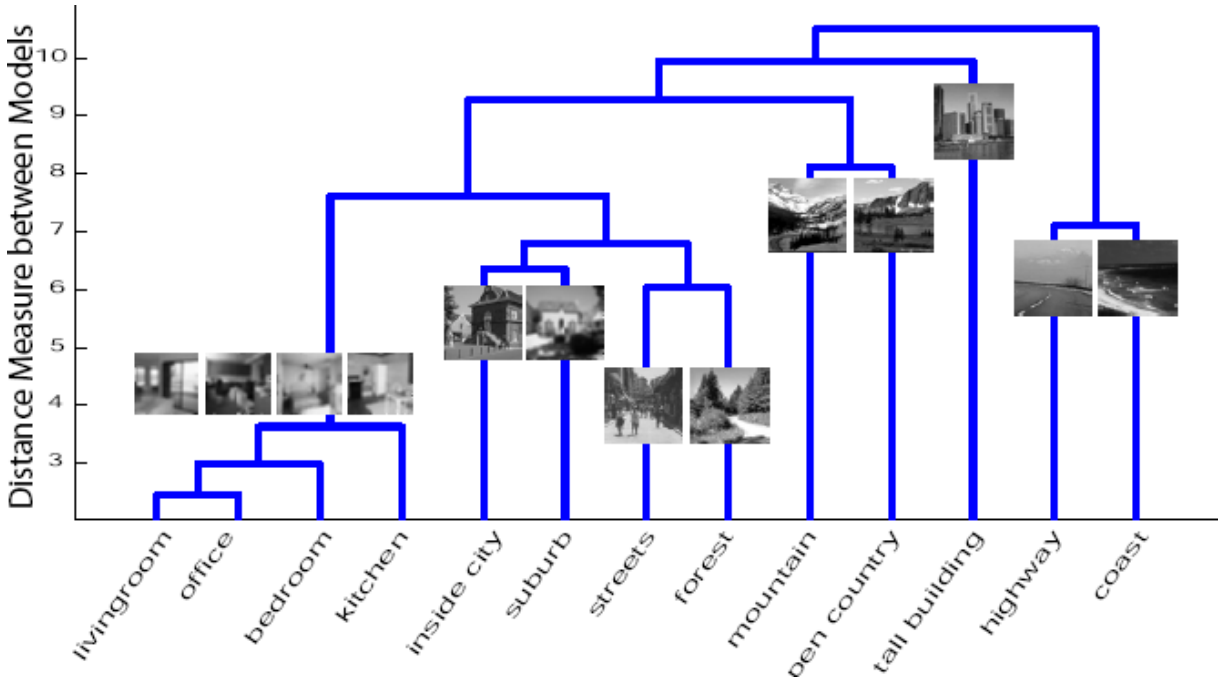


Figure 2.2: Dendogram showing the relationships between scene categories learnt by the system of Fei–Fei and Perona [Fei05]. The model matches human intuition well. Image taken from [Fei05].

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.* [HEH06] 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 over segmented into superpixels using the algorithm of Felzenszwalb *et al.* [FH04]. 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

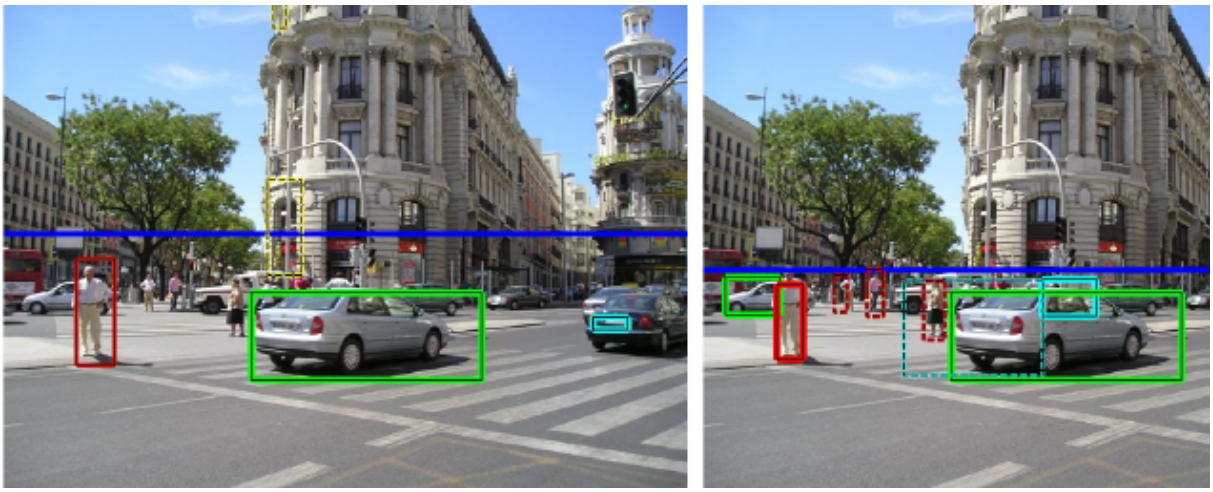


Figure 2.3: The left image shows car and pedestrian detections when using local features only. The right image shows detections when geometric context is leveraged: several false–positives are eliminated and several new correct detections are made. Image taken from [HEH06].

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 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.* [SSN09]. 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.
Simpler to Hoiem's approach, 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 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

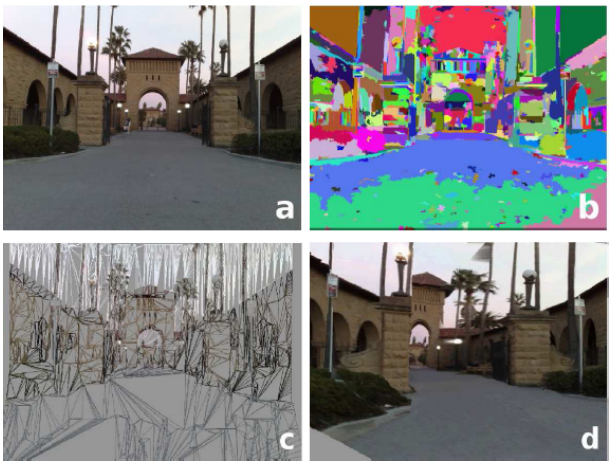


Figure 2.4: Results from the geometric context system of Saxena *et al.* [SSN09]. From top left to bottom right the panes show (a) the original, (b) the superpixels, (c) the inferred 3D model, (d) a re–projection of the 3D scene.

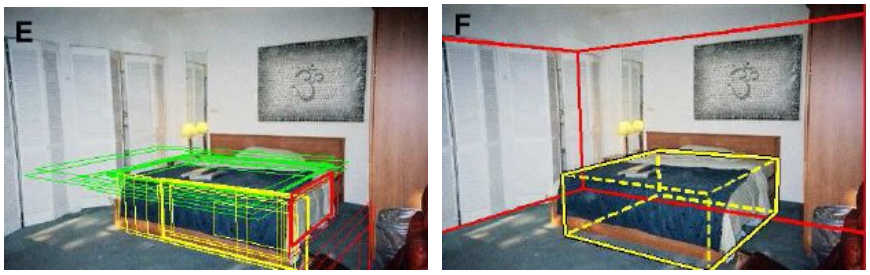
that share a boundary in the image. The node potentials are learnt from the superpixel features and edge potentials are learnt from the boundary features. The authors allow for coplanar, connected, and disconnected relationships across boundaries, with the former preferred *a priori* over the latter. 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 [CY99], 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 [KZ02b], single–view reconstruction [DCLK09, **?**], and multiple–view reconstruction [FCSS09, **?**]. In general, the Manhattan world assumption is appealing for scene understanding ~~tasks~~ because as it reduces the size of the hypothesis class one must search over. consider, while retaining the flexibility to model many man-made environments.

Cuboid Models

Hedau *et al.* [**?**] 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.



(a) Object hypotheses (b) Scene layout and object boundary

Figure 2.5: Results from the cuboid model of Hedau *et al.* [**?**, **?**]. Image taken from [**?**].

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 8 of this thesis, though we learn within a much more general hypothesis class.
In later work [**?**] 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 hypothesized, 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.* [DCLK09] 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:

- 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

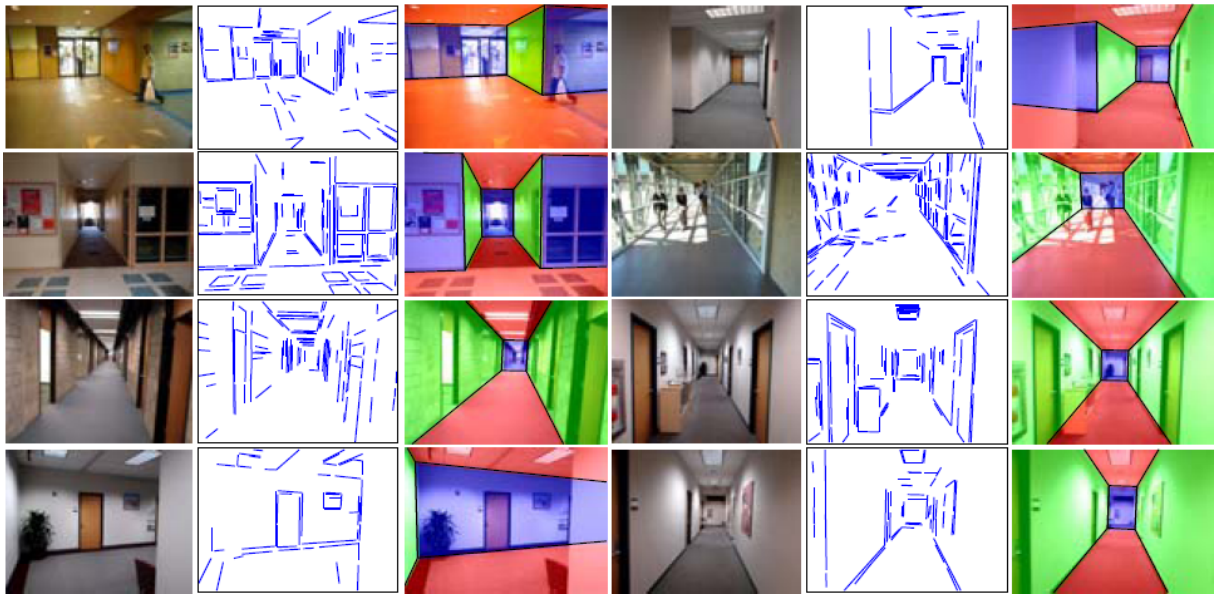


Figure 2.6: Inferred scene structure obtained for single images by Lee *et al.* [DCLK09]. Each triplet shows the original image, the detected line segments, and the final scene geometry.

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 Manhattan 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.* [FCSS09] 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 each 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 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–

(a) Input image (b) Inferred scene structure (c) 3D rendering

Figure 2.7: Results from the blocks world model of Gupta *et al.* [GEH10]. Image taken from [GEH10].

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 [Rob65] proposed building scene representations up from cuboid building blocks. This idea has recently been revisited using modern probabilistic techniques by Gupta *et al.* [GEH10]. 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.* [**?**] 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.* [BKY+08] 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 using 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 [FGMR10] 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 7.

2.2.4 Texture Recognition

Heitz and Koller [HK08] 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 [RM03] 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 explicit 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 [EVGW$^+$]. 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 [CN08] and scene description [PSN08]. Contextual reasoning, in which observations are understood in terms of their relationship to "the bigger picture", 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

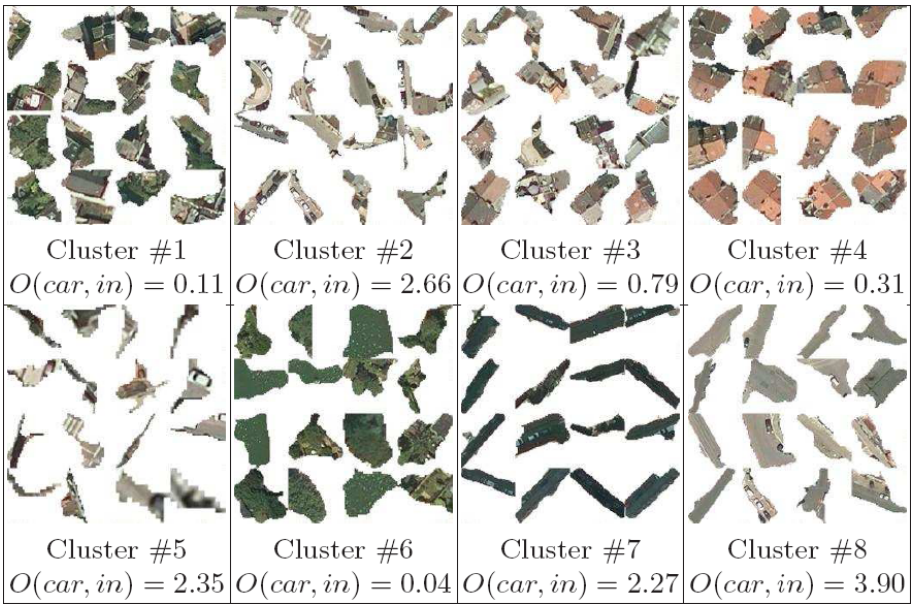


Figure 2.8: Superpixels clusters generated by the model of Heitz and Koller [HK08]. Shown below each panel is the odds ratios for a car appearing near a superpixel from that class. The odds ratio is higher for road–like regions than for foliage– or water–like regions.

time–indexed series of observations.

2.3.1 Ego–centric approaches

Martinez *et al.* [MSB05] 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 [RESL98] 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 up to 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.* [SMmRB05] 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 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

Figure 2.9: Semantic labels assigned by the range data classification system of Martinez *et al.* [MSB05]. 82% of places were correctly classified, despite the environment not having been seen during training.

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.* [PSN08] 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 distri-

Figure 2.10: Semantic labels output by the system of Posner et al [PSN08].

bution. 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.
Buschka and Saffiotti [BS02] 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 [FP02]. 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.* [VGNS07] 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. An example map and the robot's inferences about place categories is shown in Figure 2.11.

2.4 SLAM

In order for a robot to navigate within an unknown environment it must build a map of its environment and simultaneously localise itself within that map. The representation of the map and robot pose, and the techniques used to estimate these quantities, form the problem known as simultaneous localisation and mapping (SLAM). These problems are difficult because the

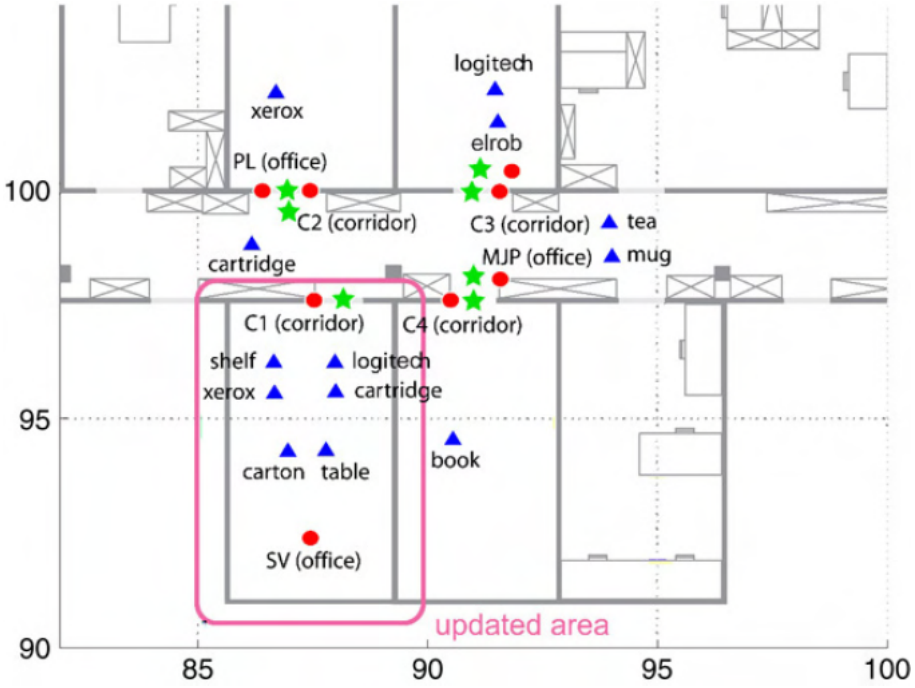


Figure 2.11: Example of an object–centric map of [VGNS07]. The blue triangles show object detections, the red and green stars show doorways the system has identified, and the red dot shows the robot's inferred place category for the outlined room, which in this case is an office.

solution to either part seems to rely upon the solution to the other, yet it is also a problem of great importance, since a working SLAM system is a first requirement for many robotics and vision tasks, such as navigation, path planning, spatial reasoning, and augmented reality.
Despite the considerable progress of SLAM systems over the past two decades, implementations remain imperfect and cannot be treated as infallible black boxes. However, our work is intended to *utilise* SLAM rather than improve upon the state–of–the–art so we keep this section brief, simply outlining the major paradigms within the literature.

2.4.1 Bundle Adjustment

The SLAM problem can be modelled as a maximum–likelihood inference problem over robot pose and landmark locations. The standard probabilistic model for SLAM consists of a sequence of poses s_t organised as a Markov chain, a set of landmark locations x_i, and at each time step, and a set of landmark observations z_j connecting landmarks and poses. Assuming Gaussian observation errors, maximum likelihood inference can be written as a nonlinear least squares optimisation problem known as bundle adjustment. This is in fact the standard approach to the offline variant of SLAM known as structure–from–motion, but for online applications, full bundle adjustment is too expensive in both time and space. Each SLAM algorithm described below can be viewed as a particular approximation to the full maximum likelihood solution.

2.4.2 Kalman Filtering

The Kalman filter [Kal60] is a recursive state estimator that represents the posterior over states as a Gaussian distribution. The Extended Kalman Filter (EKF) generalizes this approach to non–linear observation models, which was applied to SLAM by Smith and Cheeseman [SC87]. The relationship between the EKF and the maximum likelihood approach is as follows. If we approximate the posterior on states as a Gaussian distribution then the EKF is the maximum likelihood estimator. Unfortunately, the state vector in SLAM includes entries for all landmarks as well as the robot pose, and in practice the posterior over these quantities becomes highly non–Gaussian. Another drawback of the EKF is the necessity of maintaining a full covariance matrix over the system state, which in the case of SLAM grows with the number of landmarks. Montemerlo *et al.* [MTKW02] have proposed a system known as FastSLAM to try to overcome some of these problems. FastSLAM maintains a particle filter over robot poses, with each particle maintaining a separate EKF over landmark locations.

2.4.3 Particle Filtering

Particle filters are a class of recursive state estimator that approximate the posterior over states by sampling from it. At the first time step, a series of samples are drawn from the prior over states, then at each time step samples are updated according to new observations. There are several methods used to perform this update; for example, the Sequential Importance Resampling algorithm of Gordon *et al.* [**?**] propagates samples through a proposal distribution, then assigns weights to each sample by evaluating the posterior. Over time, samples in low–probability regions of the distribution are deleted and replaced by cloning samples in high–probability regions.

2.4.4 Parallel Tracking and Mapping

Klein and Murray [KM07] developed an approach to SLAM in which the posterior over states is approximated by applying bundle adjustment to a sample of frames from the input sequence. Their system, known as parallel tracking and mapping (PTAM), uses bundle adjustment to resolve the 3D locations of a set of landmarks, which are then tracked through the remainder of the video stream. During the tracking phase, landmarks are considered fixed, so only the camera pose is being estimated. When necessary, an incoming frame is selected as a new key–frame, which adds a new set of landmarks to the map. The bundle adjuster then updates the position of all landmarks and camera poses in a separate thread, while the landmarks continue to be tracked in real–time. This allows the system to maintain an up–to–date estimate of the camera position without conceding any of the approximations involved with Kalman filtering or particle filtering.

PTAM possesses a number of advantages over other SLAM systems. First, the division of tracking and mapping into separate threads allows PTAM to maintain very dense maps (thousands of points), which would be intractable for real–time performance under other SLAM

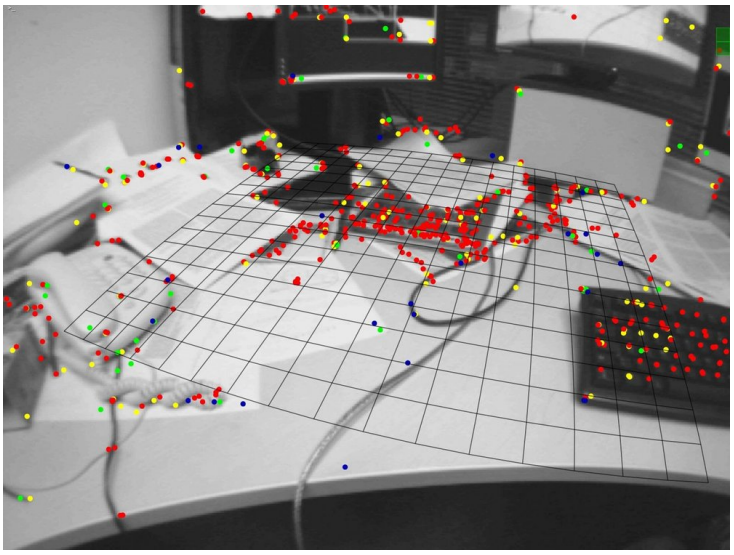


Figure 2.12: A set of landmarks features identified by PTAM [KM07]. The colours indicate confidence in the localisation of each landmark.

paradigms. Second, PTAM gains robustness to measurement errors because frames containing low–quality observations will not be selected as key–frames. PTAM also gains robustness from the large number of landmarks it tracks, which allows a robust estimator to reliably find inliers even in the face of many outliers.
For these reasons we use PTAM throughout the experiments in this thesis, though all of our work could equally well utilise any other SLAM system.

2.5 Conclusions

Modern SLAM systems generate accurate metric maps of an environment, and can do so robustly and efficiently using visual input alone [KM07]. The higher–level problem of deriving 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 [MSB05] and Stachniss [SMmRB05], who show that place categories can be obtained from photometric and geometric cues, and Posner [PSN08], 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 [Tor03] has led to the development of many types of contextual cues, including geometric [HEH05, SSN09], textural [HK08], and model–based [DCLK09] 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

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 because knowledge of the wearer's location will give a strong indication of the activities the wearer might undertake and the objects with which they are likely to interact.
In this chapter we propose a texture analysis approach to this problem. First we introduce the notion of textons, which form the basis of our approach. Next we motivate the use of textons for this particular problem by exploring the relationship between textons and scenes. In Section **??** we describe our model relating textons to place recognition in detail. We then conclude this chapter with a short discussion of some code optimisations for our algorithm.

3.2 Textons

The notion of textons as atomic texture elements was born in the neuroscience community when in 1981 Julesz [Jul81] 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 [ZGWX05] for an extensive discussion), including both topological and statistical descriptions. We follow the statistical account first proposed by Malik *et al.* [MBSL99]. 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 Euclidean sense. The image is henceforth represented as an array of texton indices (the "texton map"); the remainder of the image data is discarded.

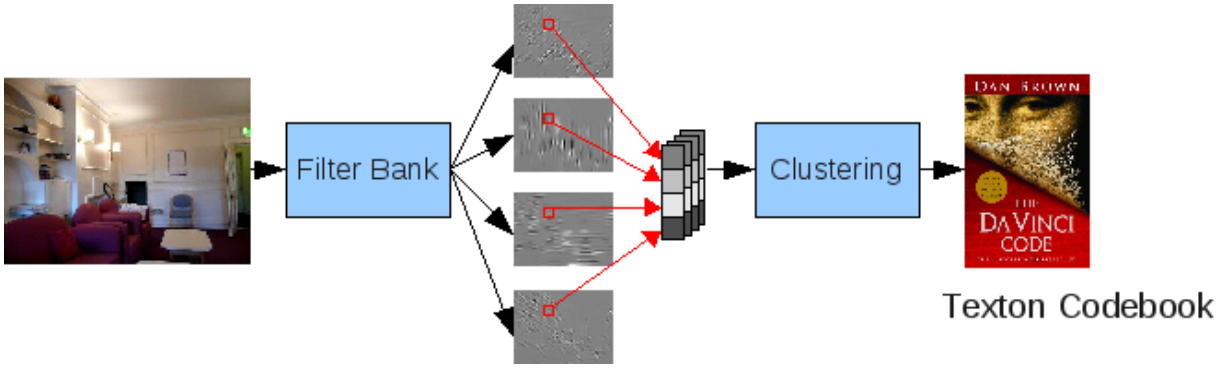


Figure 3.1: The texton generation process.

3.2.1 Texture Recognition

In the past textons have been successfully used to classify close–up photos of materials such as wood, paper, and glass. Varma and Zisserman [VZ05] 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 as the bag–of–features method. One of the advantages of this general methodology is the dimensionality reduction obtained through quantisation: high–dimensional feature vectors are projected to discrete labels, allowing accurate learning from small datasets and avoids the other symptoms of the curse of dimensionality.

3.2.2 Textons for High–Level Reasoning

Textons are a widely used and well–understood within the computer vision community [ZGWX05, VZ05, MBSL99]. 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 [Fei05, CN08]. 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 unleveraged 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 hardly a 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.

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 essentially 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 (the carpet, for example) or there simply is no texture (the white walls, 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 [FP02], whereas in this case we can see that 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 exotic image structures such as junctions and line endpoints. These structures are also sought explicitly in many image understanding algorithms [FP02], whereas the use of textons

selects these structures automatically and unsupervised.

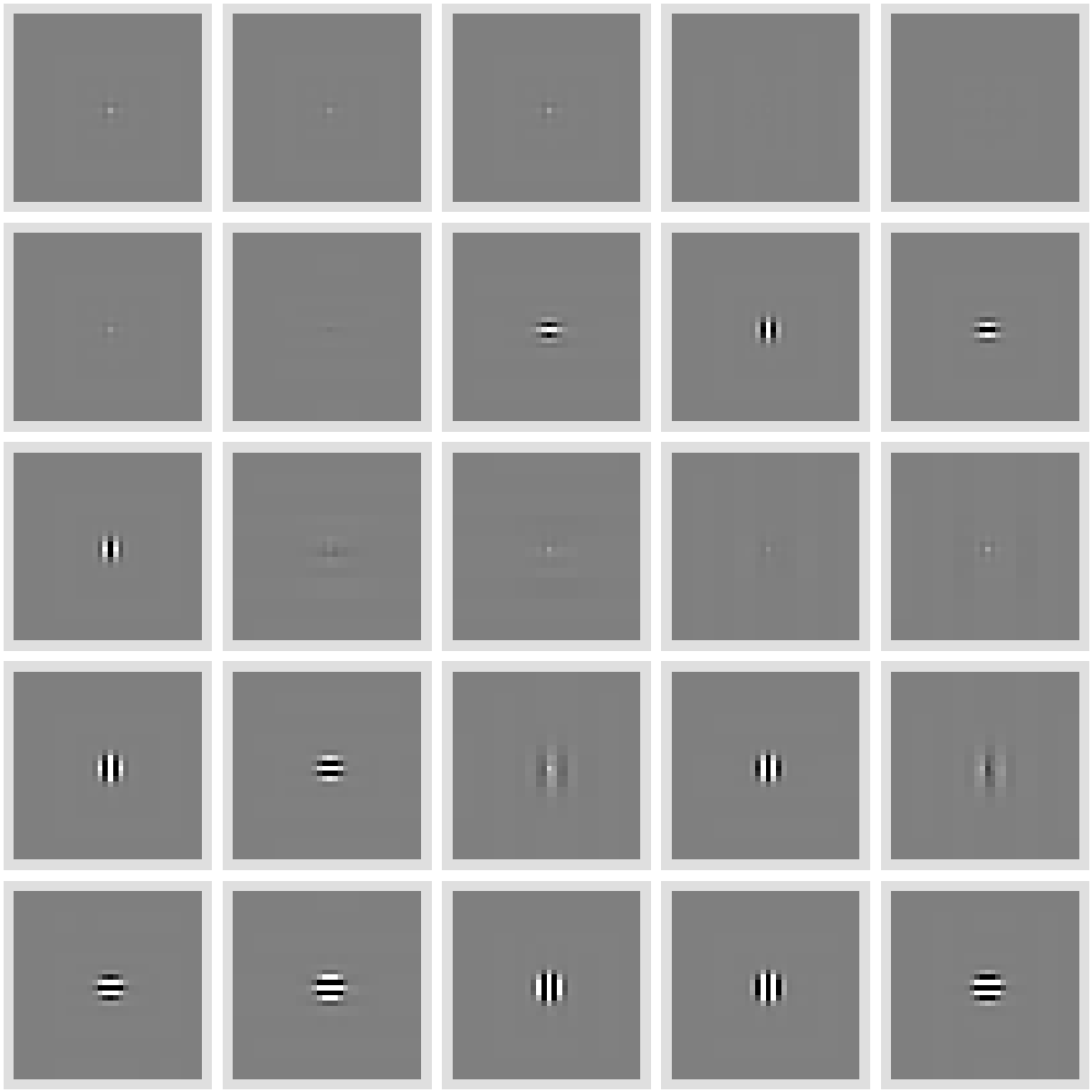


Figure 3.3: Textons generated for our dataset. Sorted from left to right, top to bottom by descending frequency.

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.

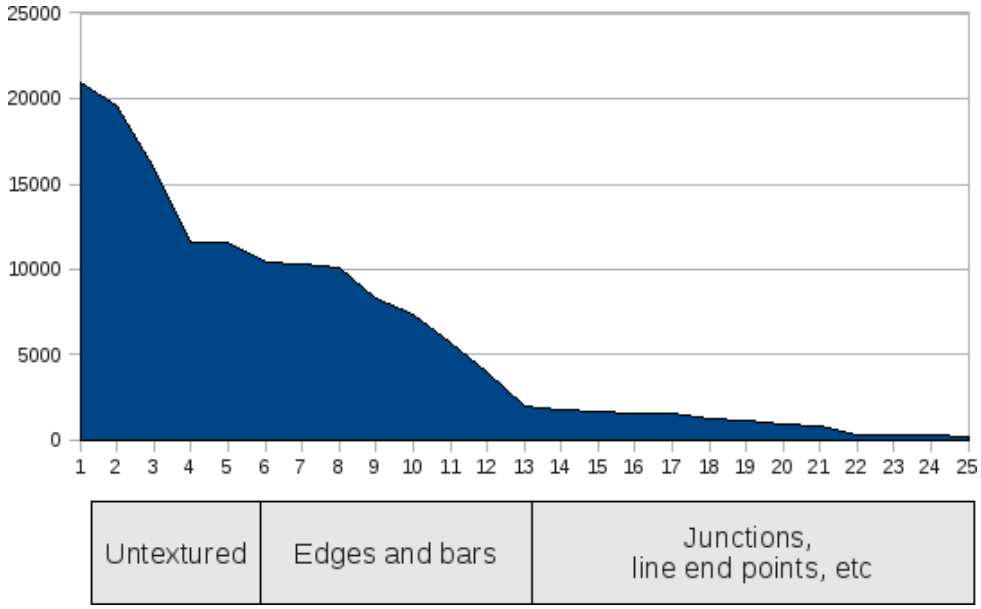


Figure 3.4: Frequency distribution of textons.

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–texton heatmaps are shown in Figure 3.5. The first two rows show heatmaps 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 heatmap 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 illustration 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.

3.3 Learning and Inference

Having motivated the use of textons for image understanding we now describe our probabilistic model that relates textons to scenes. We wish to model image categories according to the locations in which textons appear. One popular approach is the bag–of–features model [Jeb03] 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 the 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. We model the likelihood given class c as

$$p(I \mid c) = \prod_{i=0}^{N} \prod_{j=0}^{N} p(t_i,\ t_j,\ s_{i,j} \mid c) \tag{3.1}$$

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 [Par62] 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

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

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.

3.4 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. As an added challenge we gave all frames captured in corridors the same label (there were four different corridors in the sequence with considerable variations in appearance).
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 localisations 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

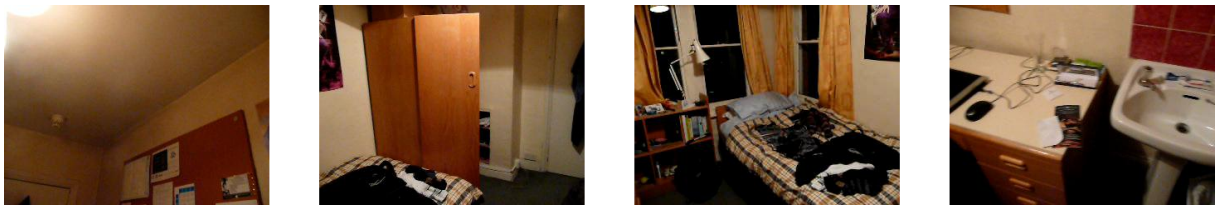


Figure 3.6: Four frames with the "bedroom" label. There are almost no overlapping scene parts but the system is required to (and did successfully) recognise each of them as part of the same place.

the class–conditional likelihood in feature space by building Gaussian mixture models with the Gaussians constrained to be spherical, exactly as described in [Tor03].
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, namely 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 our 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.

# train frames	Our system	Torralba *et al.*	KNN
103	**81%**	—	45%
230	**83%**	62%	52%
565	**85%**	70%	55%

Table 3.1: Place recognition results with varying numbers of training frames (total for all labels). For the experiment with 103 training frames (top row) we were unable to estimate the Gaussian mixtures required for Torralba's system due to the sparsity of the training examples in feature space.

3.5 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 a classification task with three possible labels: "up", "straight", and "down" (see Figure 3.10).

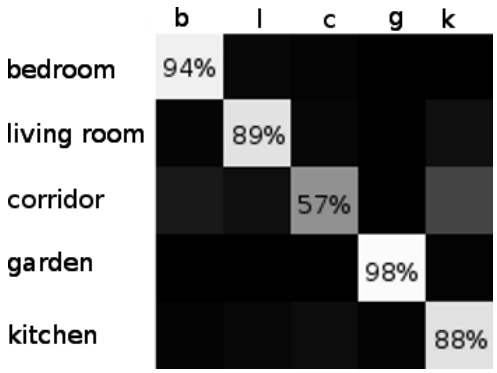


Figure 3.7: Confusion matrix for place recognition.

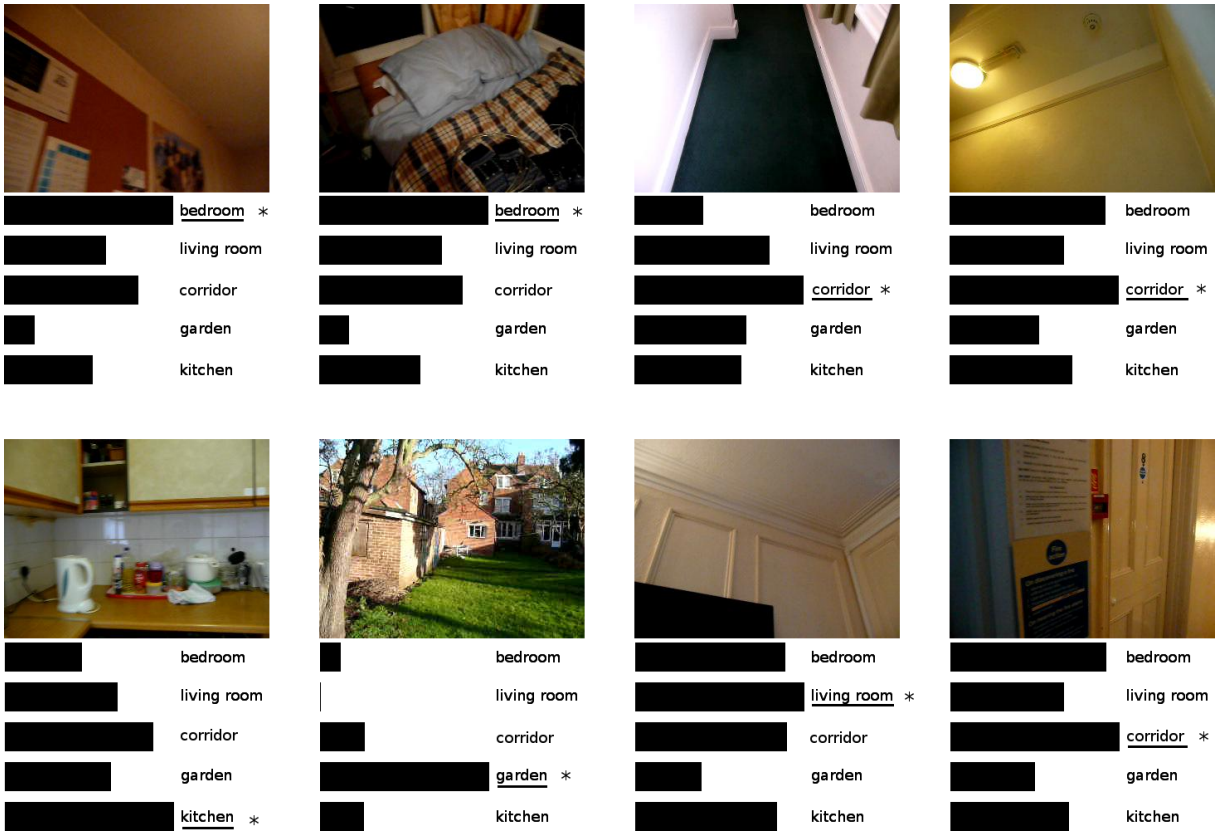


Figure 3.8: Example frames for which our classifier succeeded. The ground truth label is underlined and the output from our system is starred. We show the log likelihoods not the actual posterior because the large number of terms in (3.1) causes the posterior to always be sharply peaked and hence the log likelihood is more informative for visualisation. Note the variation between frames with the same label, and the poverty of the information contained in many frames.

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

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.

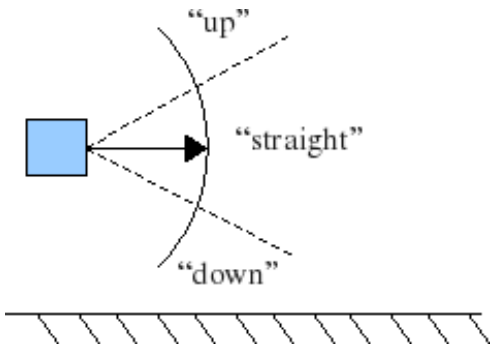


Figure 3.10: Labels for camera orientation classification.

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 experiments 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.

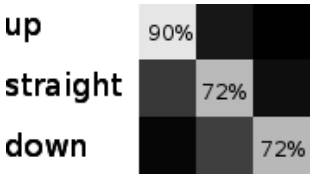


Figure 3.11: Confusion matrix for camera orientation classification.

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.

# train frames	Our system	Torralba *et al.*	KNN
88	**70%**	61%	59%
728	**79%**	63%	59%

Table 3.2: Camera orientation classification results using small and large training sets. We were able to estimate the Gaussians for Torralba's system using only 88 training examples because there were fewer labels than in the place recognition problem.

3.6 Code Optimisations

The most computationally demanding aspect of our system turns out to be the convolutions needed to generate the pixel features. In this section we describe several optimisations for the convolution operation. Timing results are shown in Figure 3.13 at the end of this chapter.

3.6.1 Separable Kernels

A function of two variables is *separable* if it can be factorised into two functions of one variable each. Convolving an image with a separable 2D kernel can be decomposed into two 1D convolutions along each axis [Lin93]. Separable convolutions are widely used in signal processing and computer vision since executing two 1D convolutions is faster than a full 2D convolution. 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,

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

where and g_2 represents the two–dimensional Gaussian function. Making appropriate substitutions for a and b,

$$\begin{aligned} H_{real}(x,y) &= g_2(x,y)\cos((ax)+(by)) && (3.4)\\ &= g_1(x)g_1(y)\Big(\cos(ax)\cos(by) - \sin(ax)\sin(by)\Big) && (3.5)\\ &= \Big(g_1(x)\cos(ax)\Big)\Big(g_1(y)\cos(by)\Big) - \Big(g_1(x)\sin(ax)\Big)\Big(g_1(y)\sin(by)\Big) && (3.6) \end{aligned}$$

where in (3.5) we have used the cosine expansion formula and replaced the Gaussian function with its well–known separated form. (3.6) shows that the real Gabor function can be expressed as the difference between two separable functions. We implement this by running two separated convolutions (consisting of two 1D convolutions each) and then taking the difference between the results, which is significantly faster than performing the original convolution.

3.6.2 Parallelization over filters

Generating the pixel features requires involving an input image with 12 different filters. To leverage the parallelization capabilities of modern multi–core CPUs we implemented a version of the code that executes 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.

3.6.3 Parallelization over images

During training all images are available at the outset, so it is possible to parallelize over images rather than over filters. Since each image can be processed independently strategy outperforms parallelization over filters, which requires synchronisation at certain points. The reduced synchronisation results in higher processor utilisation and a corresponding increase in performance. This strategy is, of course, only possible during training since at evaluation time frames arrive one–by–one and must be processed as they arrive.

3.6.4 Parallelization over pixels

Modern graphics hardware is designed to allow efficient ultra–small–scale parallelization, such as per–pixel parallelization. This is is well suited to performing convolutions since each output pixel is functionally independent. To leverage this we implemented our convolutions in CUDA, which is a C–like language designed by NVidia for programming graphics hardware. We transmit frames to the GPU as they arrive, wherein the convolutions are performed and then transferred back to the CPU. This results in a substantial performance gain over the CPU implementation.
We note that further improvements may be possible since in the current implementation, the majority of the time is spent transferring data between the CPU and GPU, in comparison to which the time spent performing the convolutions is negligible. 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. We plan to implement this if it becomes necessary in order to attain frame rate performance.

3.6.5 Timing results

Results from timing evaluations are shown in Figure 3.13. The GPU strategy outperforms all others and this is the implementation we use throughout the rest of this report. 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. As well as meeting our frame–rate performance goal, this speedup allows us to rapidly test new algorithms within the rest of our system, which improves productivity substantially.
It is interesting to note that the when the image is enlarged to include four times as many pixels, the three strategies that execute on the CPU all perform very close to four times slower, whereas the GPU implementation degrades by a factor closer to three. This indicates that this tasks saturates the CPU performance but that a significant overhead remains in the GPU implementation. This matches our observation in the previous section transmitting data between the CPU and GPU consumes much of the time taken by that strategy, and that further performance improvements are possible.

	640 × 480		640 × 480	
Strategy	**Time per frame**	**FPS**	**Time per frame**	**FPS**
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

Figure 3.13: 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.

3.7 Conclusions

In this section we have shown that texture structure is useful for deducing high–level information about indoor 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 highly–parallel graphics hardware our system is capable of operating at video frame rate. In the next chapter we extend our system to reason about object locations.

4

Geometry of Indoor Environments

4.1 Introduction

TODO The need for coarse geometry - make sure not to repeat introduction chapter.

4.2 Indoor Manhattan Environments

In this section we introduce the notion of indoor Manhattan environments. Let a reconstruction M consist of a set of polygons P_i with vertices $\boldsymbol{v}_j \in \mathbb{R}^3$. An *indoor Manhattan reconstruction* is a reconstruction M with the following properties.

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

$$\boldsymbol{n}_i \cdot \boldsymbol{n}_j \in \{0, 1\} \tag{4.1}$$

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

$$\boldsymbol{v} \cdot \boldsymbol{n}_v \in \{z_f, z_c\} \tag{4.2}$$

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

$$\boldsymbol{v} \cdot \boldsymbol{n}_v \in \{z_f, z_c\} \tag{4.3}$$

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

$$\boldsymbol{u} \cdot \boldsymbol{n}_v = z_c \tag{4.4}$$

and

$$\boldsymbol{u} = \boldsymbol{v} + \lambda \boldsymbol{n}_v \tag{4.5}$$

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

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

$$\boldsymbol{x} = H \begin{bmatrix} R & \boldsymbol{t} \end{bmatrix} \boldsymbol{X} \ , \tag{4.6}$$

where H is a 3×3 homography, R is a 3–dimensional rotation matrix, and $\boldsymbol{t}$ 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 $\boldsymbol{c}$ is located between the floor and ceiling planes,

$$z_f < \boldsymbol{c} \cdot \boldsymbol{n}_v < z_c \tag{4.7}$$

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

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 frustrutrum remains an indoor Manhattan reconstruction as defined above. [XXX not true if view frustrum clips walls so no longer meet floor or ceiling]

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 F as a tuple (R_w, Z, T), where

- R_w is a 3–dimensional rotation matrix;
- $Z = (z_f, z_c)$ is a pair of scalars defining the floor and ceiling planes,

$$P_{\mathsf{floor}} = \{\boldsymbol{x} \mid \boldsymbol{x} \cdot [0\ 0\ 1]^T = z_f\} \tag{4.8}$$
$$P_{\mathsf{ceil}} = \{\boldsymbol{x} \mid \boldsymbol{x} \cdot [0\ 0\ 1]^T = z_c\} \ ; \tag{4.9}$$

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

A floorplan F is converted to a reconstruction as follows. First, trace each line segment in the floor plane. Next, extrude those line segments vertically to meet the ceiling plane, resulting in a set of vertical planes. Now add polygons for the floor and ceiling planes, which may simply be sufficiently large quads. Finally, apply the coordinate transform R_w. Formally,

$$M = \{(\boldsymbol{p}_i, \boldsymbol{q}_i, \boldsymbol{r}_i, \boldsymbol{s}_i)\} \tag{4.10}$$

$$\boldsymbol{p}_i = R_w \begin{bmatrix} \boldsymbol{u}_i \\ z_f \end{bmatrix} \tag{4.11}$$

$$\boldsymbol{q}_i = R_w \begin{bmatrix} \boldsymbol{v}_i \\ z_f \end{bmatrix} \tag{4.12}$$

$$\boldsymbol{r}_i = R_w \begin{bmatrix} \boldsymbol{v}_i \\ z_c \end{bmatrix} \tag{4.13}$$

$$\boldsymbol{s}_i = R_w \begin{bmatrix} \boldsymbol{u}_i \\ z_c \end{bmatrix} \tag{4.14}$$

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 is a point in an image.

4.3.1 Images of Floorplans

In this section we discuss some geometric properties of floorplans projected into images.
Let I be the image formed by camera $C = (H, R, \boldsymbol{t})$ viewing floorplan F. Let $I(x, y)$ be the index of the wall that projects to the location (x, y) in the image plane, or 0 if no wall plane projects to (x, y). In the latter case it must be that a point on the floor or ceiling plane projects to (x, y) due to our assumption of a closed environment.
In homogeneous image coordinates, the vanishing points associated with the three Manhattan orientations are given by

$$\boldsymbol{v}_i = HRR_w\boldsymbol{e_i} \tag{4.15}$$

where $e_i \in \mathbb{R}^3$ has i$^{\text{th}}$ coordinate equal to 1 and zeros elsewhere. Indoor Manhattan environments distinguish a unique "vertical" direction. To capture the semantic distinction of this direction from the other two cardinal orientations we label $\boldsymbol{v_v} = \boldsymbol{v}_3$.

Rectification

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:

$$H_{\text{rect}} = \begin{pmatrix} \boldsymbol{v_v} \times \boldsymbol{e}_3 \\ \boldsymbol{v_v} \\ \boldsymbol{v_v} \times \boldsymbol{e}_3 \times \boldsymbol{v_v} \end{pmatrix} . \tag{4.16}$$

Let I' be the image formed by applying the homography H_{rect} as a coordinate transform to I,

$$I'(\boldsymbol{x}) = I(H_{\text{rect}}\boldsymbol{x}) \tag{4.17}$$

where we are implicitly working in homogeneous coordinates. We say that I' is *vertically rectified* and we make the following claim. If two world points are separated by an offset 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 $\boldsymbol{v}$ and $\boldsymbol{u}$ be points with $\boldsymbol{u} = \boldsymbol{v} + \lambda\boldsymbol{e}_3$ and consider their projections in I',

$$\begin{aligned} \boldsymbol{p} &= H_{\text{rect}}\big(R\boldsymbol{v} + \boldsymbol{t}\big) && (4.18)\\ \boldsymbol{q} &= H_{\text{rect}}\big(R\boldsymbol{u} + \boldsymbol{t}\big)\ . && (4.19) \end{aligned}$$

Now

$$\begin{aligned} \boldsymbol{q} - \boldsymbol{p} &= H_{\text{rect}}\big(R\boldsymbol{u} + \boldsymbol{t}\big) - H_{\text{rect}}\big(R\boldsymbol{v} + \boldsymbol{t}\big) && (4.20)\\ &= H_{\text{rect}}\Big(R(\boldsymbol{v} + \lambda\boldsymbol{e}_3) + \boldsymbol{t} - R\boldsymbol{v} - \boldsymbol{t}\Big) && (4.21)\\ &= \lambda H_{\text{rect}}R\boldsymbol{e}_3\ . && (4.22)\\ &= \lambda H_{\text{rect}}\boldsymbol{v}_{\boldsymbol{v}}\ . && (4.23) \end{aligned}$$

The x–coordinate of (4.23) is

$$\lambda(\boldsymbol{v}_{\boldsymbol{v}} \times \boldsymbol{e}_3)^T\boldsymbol{v}_{\boldsymbol{v}} = 0\ , \tag{4.24}$$

since $\boldsymbol{v}_{\boldsymbol{v}} \times \boldsymbol{e}_3$ is orthogonal to $\boldsymbol{v}_{\boldsymbol{v}}$, and the dot–product between orthogonal vectors is zero.

Sampling Issues

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 a non–uniformly sampled image plane. Worse, if the vertical vanishing point $\boldsymbol{v}_{\boldsymbol{v}}$ is within the original image boundary, then the transformation H_{rect} will introduce a singularity at which a finite region of the input image is mapped to an infinite region in the output image.

Thankfully, the validity of the algorithms presented in the remainder of this thesis do not actually depend on performing the rectification (4.17) — 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 (4.17) 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.

TODO: Talk about the fact that we'll assume R and Z known.

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 preceeding 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 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 that scale ambiguity. In the single view setting we simply avoid resolving scale.

To define μ we first need to introduce the *Manhattan correspondence*. For the purpose of this section we will say that two image points $\boldsymbol{p_f}$ and $\boldsymbol{p_c}$ 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 **??**. Formally, $\boldsymbol{p_f}$ and $\boldsymbol{p_c}$ are in correspondence if and only if there exists $\boldsymbol{X_f}, \boldsymbol{X_c} \in \mathbb{R}^3$ with

$$\boldsymbol{p_f} = H(R\boldsymbol{X_f} + \boldsymbol{t}) \tag{4.25}$$

$$\boldsymbol{p_c} = H(R\boldsymbol{X_c} + \boldsymbol{t}) \tag{4.26}$$

$$\boldsymbol{X_c} = \boldsymbol{X_f} + r\boldsymbol{e}_3 \quad r \in \mathbb{R} \tag{4.27}$$

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

$$\boldsymbol{p_f} = H_a \boldsymbol{p_c} \tag{4.28}$$

if and only if $\boldsymbol{p_f}$ and and $\boldsymbol{p_c}$ are in correspondence [Cri01]. We define μ as the unique real number such that

$$H_a = I + \mu \frac{\boldsymbol{v_v}\boldsymbol{h}^T}{\boldsymbol{v_v} \cdot \boldsymbol{h}} \tag{4.29}$$

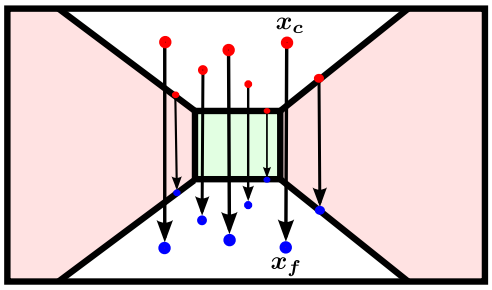


Figure 4.1: The mapping H_a transfers points between the ceiling and floor.

where

$$\boldsymbol{h} = HR\boldsymbol{e}_1 \times HR\boldsymbol{e}_2 \tag{4.30}$$

is the horizon line. Given any corresponding pair $(\boldsymbol{p_f}, \boldsymbol{p_c})$, μ is given by [Cri01]

$$\mu = \langle \boldsymbol{v_v}, \boldsymbol{p_c}, \boldsymbol{p_f}, (\boldsymbol{p_c} \times \boldsymbol{p_f}) \times \boldsymbol{h} \rangle \ , \tag{4.31}$$

where

$$\langle \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d} \rangle = \frac{(\boldsymbol{a} - \boldsymbol{c}) \cdot (\boldsymbol{b} - \boldsymbol{d})}{(\boldsymbol{b} - \boldsymbol{c}) \cdot (\boldsymbol{a} - \boldsymbol{d})} \tag{4.32}$$

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 givey by $\boldsymbol{h}' = |M|M^{-T}\boldsymbol{h}$, the vertical vanishing point is $\boldsymbol{v_v}' = M\boldsymbol{v_v}$, and the Manhattan homology is

$$H'_a = I + \mu' \frac{\boldsymbol{v_v}'\boldsymbol{h}'^T}{\boldsymbol{v_v}' \cdot \boldsymbol{h}'} \ . \tag{4.33}$$

We also have that

$$\begin{aligned} H'_a &= MH_aM^{-1} && (4.34)\\ &= M(I + \mu \frac{\boldsymbol{v_v}\boldsymbol{h}^T}{\boldsymbol{v_v} \cdot \boldsymbol{h}})M^{-1} \ . && (4.35)\\ & && (4.36) \end{aligned}$$

Substituting $\boldsymbol{h} = \frac{1}{|M|}M^T\boldsymbol{h}'$ and $\boldsymbol{v_v} = M^{-1}\boldsymbol{v_v}'$ gives

$$\begin{aligned} H'_a &= I + \mu \frac{M(M^{-1}\boldsymbol{v_v}')(\frac{1}{|M|}M^T\boldsymbol{h}')^T M^{-1}}{(M^{-1}\boldsymbol{v_v}')^T(\frac{1}{|M|}M^T\boldsymbol{h}')} && (4.37)\\ &= I + \mu \frac{\boldsymbol{v_v}'\boldsymbol{h}'^T}{\boldsymbol{v_v}' \cdot \boldsymbol{h}'} \ . && (4.38) \end{aligned}$$

Comparing (4.38) and (4.33) shows that $\mu' = \mu$. We therefore assume without loss of generality that $H = I$ 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) and vice versa. We begin by showing how to compute μ given z_f and z_c. Let

$$\boldsymbol{e_f} = z_f \, \boldsymbol{e}_3 \tag{4.39}$$

$$\boldsymbol{e_c} = z_c \, \boldsymbol{e}_3 \tag{4.40}$$

$$\boldsymbol{p_f} = R\boldsymbol{e_f} + \boldsymbol{t} \tag{4.41}$$

$$\boldsymbol{p_c} = R\boldsymbol{e_c} + \boldsymbol{t} \tag{4.42}$$

Then

$$\begin{aligned}
(\boldsymbol{p_c} \times \boldsymbol{p_f}) \times \boldsymbol{h} &= \big((R\boldsymbol{e_f} + \boldsymbol{t}) \times (R\boldsymbol{e_c} + \boldsymbol{t})\big) \times \big(R\boldsymbol{e}_1 \times R\boldsymbol{e}_2\big) && (4.43)\\
&= \big((R\boldsymbol{e_c} + \boldsymbol{t}) \times (R\boldsymbol{e_f} + \boldsymbol{t})\big) \times R(\boldsymbol{e}_1 \times \boldsymbol{e}_2) && (4.44)\\
&= \big(R(\boldsymbol{e_c} - \boldsymbol{e_f}) \times \boldsymbol{t}\big) \times R\boldsymbol{e}_3 \, . && (4.45)\\
& && (4.46)
\end{aligned}$$

We can now write

$$\begin{aligned}
\mu &= \langle \boldsymbol{v_v}, \boldsymbol{p_c}, \boldsymbol{p_f}, (\boldsymbol{p_c} \times \boldsymbol{p_f}) \times \boldsymbol{h} \rangle && (4.47)\\
&= \frac{\big(\boldsymbol{v_v} - \boldsymbol{p_f}\big)^T \big(\boldsymbol{p_c} - (\boldsymbol{p_c} \times \boldsymbol{p_f}) \times \boldsymbol{h}\big)}{\big(\boldsymbol{p_c} - \boldsymbol{p_f}\big)^T \big(\boldsymbol{v_v} - (\boldsymbol{p_c} \times \boldsymbol{p_f}) \times \boldsymbol{h}\big)} && (4.48)\\
&= \frac{\big(R\boldsymbol{e}_3 - (R\boldsymbol{e_f} + \boldsymbol{t})\big)^T \big(R\boldsymbol{e_c} + \boldsymbol{t} - (R(\boldsymbol{e_c} - \boldsymbol{e_f}) \times \boldsymbol{t}) \times R\boldsymbol{e}_3\big)}{\big(R\boldsymbol{e_c} + \boldsymbol{t} - R\boldsymbol{e_f} - \boldsymbol{t}\big)^T \big(R\boldsymbol{e}_3 - (R(\boldsymbol{e_c} - \boldsymbol{e_f}) \times \boldsymbol{t}) \times R\boldsymbol{e}_3\big)} && (4.49)\\
&= \frac{\big(\boldsymbol{e}_3 - \boldsymbol{e_f} - R^T\boldsymbol{t}\big)^T R^T \big(R\boldsymbol{e_c} + \boldsymbol{t} - (R(\boldsymbol{e_c} - \boldsymbol{e_f}) \times \boldsymbol{t}) \times R\boldsymbol{e}_3\big)}{\big(\boldsymbol{e_c} - \boldsymbol{e_f}\big)^T R^T \big(R\boldsymbol{e}_3 - (R(\boldsymbol{e_c} - \boldsymbol{e_f}) \times \boldsymbol{t}) \times R\boldsymbol{e}_3\big)} && (4.50)\\
&= \frac{\big(\boldsymbol{e}_3 - \boldsymbol{e_f} - R^T\boldsymbol{t}\big)^T \big(\boldsymbol{e_c} + R^T\boldsymbol{t} - ((\boldsymbol{e_c} - \boldsymbol{e_f}) \times R^T\boldsymbol{t}) \times \boldsymbol{e}_3\big)}{(\boldsymbol{e_c} - \boldsymbol{e_f})^T\boldsymbol{e}_3 - (\boldsymbol{e_c} - \boldsymbol{e_f})^T(((\boldsymbol{e_c} - \boldsymbol{e_f})R^T\boldsymbol{t})\boldsymbol{e}_3)} && (4.51)\\
&= \frac{(\boldsymbol{e}_3 - \boldsymbol{e_f} - R^T\boldsymbol{t})^T(\boldsymbol{e_c} + R^T\boldsymbol{t}) + (z_c - z_f)(\|\boldsymbol{t}\|^2 - (\boldsymbol{e}_3^T R^T\boldsymbol{t})^2)}{z_c - z_f} && (4.52)\\
&= \frac{1}{z_c - z_f}\Big((z_c + z_\rho)(1 - z_f - z_\rho) + z_\rho^2 - \|\boldsymbol{t}\|^2\Big) + \|\boldsymbol{t}\|^2 - z_\rho^2 && (4.53)
\end{aligned}$$

where in the last line we substituted $z_\rho = \boldsymbol{e}_3^T R^T \boldsymbol{t}$.
TODO: show the reverse mapping.

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.

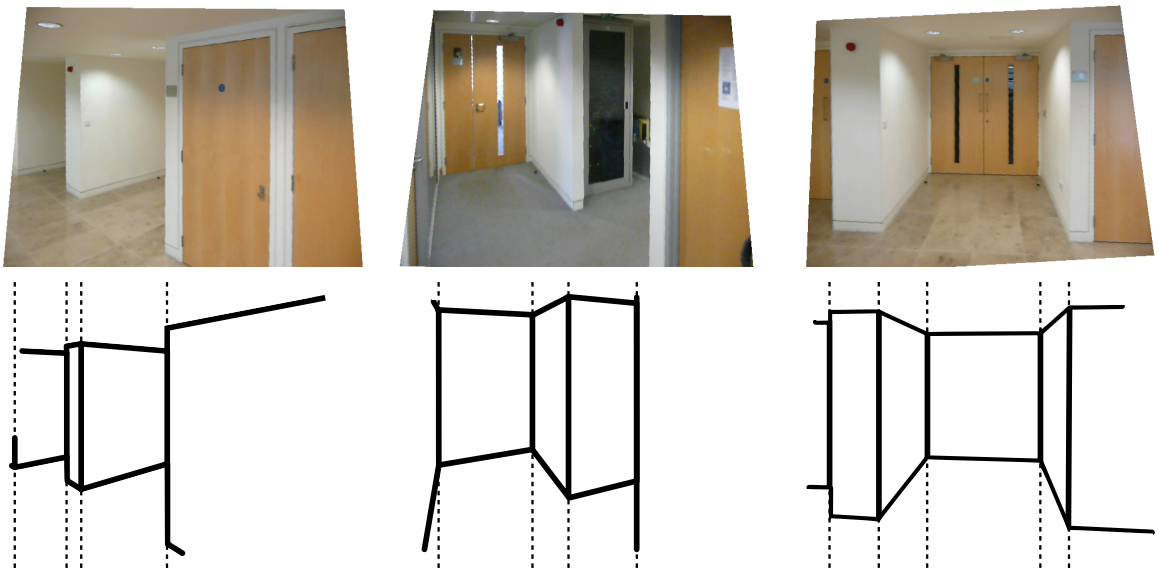


Figure 4.2: Three input images and the indoor Manhattan models we seek. Notice how each image column intersects exactly one wall.

Consider the scenes shown in Figure **??**. 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 $\boldsymbol{v}_i$ and the Manhattan homology H_a are known, the quadrangle outlining a wall segment in rectified image coordinates is fully specified by just four scalars,

$$W = (x_0, y, o, x_1) \tag{4.54}$$

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

$$\boldsymbol{p_i} = \begin{bmatrix} x & y & 1 \end{bmatrix}^t \tag{4.55}$$

$$\boldsymbol{q_i} = \boldsymbol{p_i} \times \boldsymbol{v_a} \times \begin{bmatrix} 1 & 0 & -c_{i+1} \end{bmatrix}^T \tag{4.56}$$

$$\boldsymbol{r_i} = H_a \boldsymbol{q_i} \tag{4.57}$$

$$\boldsymbol{s_i} = H_a \boldsymbol{p_i} \, . \tag{4.58}$$

Figure **??** illustrates the geometry of the parametrisation (4.59).

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

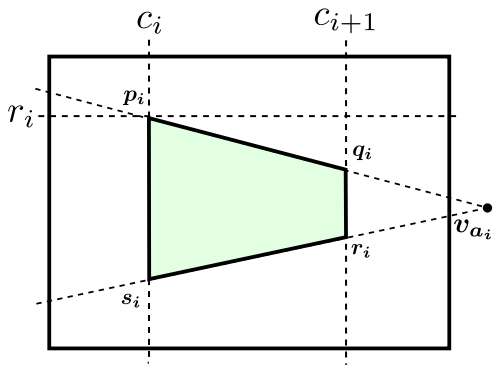


Figure 4.3: The row/column indices c_i, c_{i+1}, r_i, together with the vanishing point index $a_i \in \{l, r\}$ are sufficient via the homology H_a to determine the four vertices defining a wall.

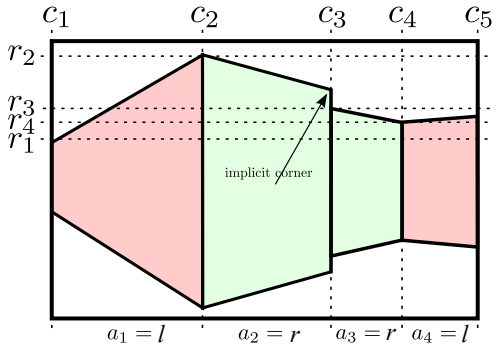


Figure 4.4: An illustration of the model $M = \{x_1, y_1, o_1, \ldots, x_4, y_4, o_4, c_5\}$.

right edge of a wall (x_1 in equation (7.22)) is redundant because it is always coincident with the left edge of the wall to its right. A scene therefore is fully specified by

$$M = (x_1, y_1, o_1,\ x_2, y_2, o_2,\ \ldots,\ x_{n-1}, y_{n-1}, o_{n-1},\ x_n)\ . \tag{4.59}$$

An example scene and its parametrisation is shown in Figure **??**. 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.* $\boldsymbol{r_i} \neq \boldsymbol{s_{i+1}}$ in general. For example, notice the second edge in Figure **??**. Ordinarily we will always have $x_1 = 0$ and $x_n = W$, but for the benefit of later chapter where we consider partial scenes we leave these values in the parametrisation.

Physical Realisability

TODO: Discuss unrealisable models? Or just permit all models?
Not all scenes in image coordinates M are physically realisable as metric reconstructions (TODO examples), but those that are not can be discarded using simple tests on the locations of walls and vanishing points as enumerated by Lee *et al.* [DCLK09]. 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 significant detail during Chapter 7.

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 (4.59) 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 assocating a pair of scalars s, o to each iamge column j,

$$S = \{(s_j, o_j)\}_{j=0}^{W} , \tag{4.60}$$

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 (which we will simply call its orientation). 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.

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 $\boldsymbol{s_i}$ and $\boldsymbol{r_i}$ according to equations (4.58) and (4.57), after which the position of the wall seam at column j is given by

$$s_j = (\boldsymbol{r_i} \times \boldsymbol{s_i}) \times \begin{bmatrix} 0 & 1 & -j \end{bmatrix}^T . \tag{4.61}$$

Finally, the orientation o_j at column j is equal to the orientation of the identified wall segment, o_i.

The constructive process 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 Observations

In this section we make several geometric observations that will be useful in the chapters to follow.

4.5.1 Classification of Corners

Lee *et al.* [DCLK09] showed that corners between adjacent walls (as viewed by some camera) can be divided into three categories as follows.

- A *convex corner* occurs where two wall segments meet exactly and form an internal angle not greater than $180°$, *i.e.*

$$\boldsymbol{r_i} = \boldsymbol{s_{i+1}} \tag{4.62}$$

$$\text{and} \tag{4.63}$$

$$\frac{(\boldsymbol{s_i} - \boldsymbol{r_i}) \cdot (\boldsymbol{q_i} - \boldsymbol{r_i})}{\|\boldsymbol{s_i} - \boldsymbol{r_i})\|\|(\boldsymbol{q_i} - \boldsymbol{r_i}\|} \leq \frac{(\boldsymbol{r_{i+1}} - \boldsymbol{r_i}) \cdot (\boldsymbol{q_i} - \boldsymbol{r_i})}{\|\boldsymbol{r_{i+1}} - \boldsymbol{r_i})\|\|(\boldsymbol{q_i} - \boldsymbol{r_i}\|} . \tag{4.64}$$

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

$$\boldsymbol{r_i} = \boldsymbol{s_{i+1}} \tag{4.65}$$

$$\text{and} \tag{4.66}$$

$$\frac{(\boldsymbol{s_i} - \boldsymbol{r_i}) \cdot (\boldsymbol{q_i} - \boldsymbol{r_i})}{\|\boldsymbol{s_i} - \boldsymbol{r_i})\|\|(\boldsymbol{q_i} - \boldsymbol{r_i}\|} > \frac{(\boldsymbol{r_{i+1}} - \boldsymbol{r_i}) \cdot (\boldsymbol{q_i} - \boldsymbol{r_i})}{\|\boldsymbol{r_{i+1}} - \boldsymbol{r_i})\|\|(\boldsymbol{q_i} - \boldsymbol{r_i}\|} . \tag{4.67}$$

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

$$\boldsymbol{r_i} \neq \boldsymbol{s_{i+1}} \tag{4.68}$$

TODO: figure from presentation

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 M. We show this by demonstrating how to convert a scene M to a floorplan F, after which we may apply equations (4.55), (4.56), (4.57), and (4.58) to recover a polygonal reconstruction.

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

$$P\boldsymbol{y} = \boldsymbol{0} \tag{4.69}$$

$$\|\boldsymbol{y}\| = 1 \tag{4.70}$$

where

$$P = \begin{bmatrix} & HR & & H\boldsymbol{t} - \boldsymbol{p} \\ 0 & 0 & 1 & -z_f \end{bmatrix} . \tag{4.71}$$

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

$$\text{backproject}(\boldsymbol{p}; z) = \text{backproject}(\boldsymbol{p}; \begin{bmatrix} 0 & 0 & 1 & -z \end{bmatrix}) . \tag{4.72}$$

To recover a floorplan F from a scene M we simply need to back–project $\boldsymbol{q_i}, \boldsymbol{s_i}$ pairs onto the floor plane. This is formalized in Algorithm **??**.

Algorithm 1 An algorithm for recovering a metric 3D scene from the vertex representation M.

Require: $M = (x_1, y_1, o_1, \ldots, x_n)$ {a scene in vertex representation}
Ensure: $F = \{(\boldsymbol{u_i}, \boldsymbol{v_i})\}$
  $F \leftarrow \emptyset$
  for $i = 1 \ldots n-1$ **do**
    $\boldsymbol{u_i} = \text{backproject}(\boldsymbol{s_i}; z_f)$
    $\boldsymbol{u_i} = \text{backproject}(\boldsymbol{r_i}; z_f)$
    $F \leftarrow F \cup (\boldsymbol{u_i}, \boldsymbol{v_i})$
  end for

4.5.3 Recovering Orientation and Depth

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

To compute either the orientation $a_{\boldsymbol{p}}$ or depth $d_{\boldsymbol{p}}$ for an image point $\boldsymbol{p}$ we first need to identify a wall that contains $\boldsymbol{p}$ within its (image–domain) outline. If there is such a wall then $\boldsymbol{p}$ has the orientation of that wall and we back–project $\boldsymbol{p}$ onto the wall in 3D to recover depth. If $\boldsymbol{p}$ is not contained by any wall then $\boldsymbol{p}$ 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 $\boldsymbol{p}$ is above or below the horizon. This process is formalized in Algorithm **??**.

These calculations are considerably simpler in the seam representation, though in this context depth and orientation can only be computed at integer pixel coordinates. For our purposes in the ensuing chapters this will suffice since we will not try to resolve the scene structure beyond the resolution of the image itself.

Suppose we are given a scene in seam representation S and a integer pixel location $\boldsymbol{p} \in \mathbb{Z}^2$. We first look up the the pair $(s_j, o_j) \in S$ for the column in which $\boldsymbol{p}$ falls, which we denote $(s_{\boldsymbol{p}_x}, o_{\boldsymbol{p}_x})$. Then the floor–wall seam in column $\boldsymbol{p}_x$ is at $s_f = s_{\boldsymbol{p}_x}$ and the ceiling–wall seam is can be recovered using the Manhattan homology. From here the orientation and depth at $\boldsymbol{p}$ are obtained using the procedure shown in Algorithm **??**.

TODO: figure showing step-by-step recovery of depth and orientation

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 **??** 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 $\boldsymbol{p}$ are functionally independent of the location of the floor–wall seam in columns other than the one containing $\boldsymbol{p}$.

Algorithm 2 An algorithm for recovering orientation and depth for an image location $\boldsymbol{p}$ under a scene hypothesis M.

Require: $M = (x_1, y_1, o_1, \ldots, x_n)$ {a scene in vertex representation}
Require: $\boldsymbol{p} = (\boldsymbol{p}_x, \boldsymbol{p}_y) \in \mathbb{R}^2$
Require: $x_1 \leq \boldsymbol{p}_x < x_n$
Ensure: $i_{\boldsymbol{p}} \in [1, n], a_{\boldsymbol{p}} \in \{1, 2, 3\}, d_{\boldsymbol{p}} > 0$
  for $i = 1 \ldots n - 1$ **do**
    if $x_i \leq \boldsymbol{p}_x < x_{i+1}$ **then**
      {Compute the floor–wall and ceiling–wall seam in column $\boldsymbol{p}_x$}
      $s_f \leftarrow \boldsymbol{e}_2 \cdot (\boldsymbol{s_i} \times \boldsymbol{r_i}) \times \begin{bmatrix} 1 & 0 & -\boldsymbol{p}_x \end{bmatrix}^T$
      $s_c \leftarrow \boldsymbol{e}_2 \cdot (\boldsymbol{s_i} \times \boldsymbol{r_i}) \times \begin{bmatrix} 1 & 0 & -\boldsymbol{p}_x \end{bmatrix}^T$
      if $\boldsymbol{p}_y < s_c$ **then**
        $a_{\boldsymbol{p}} \leftarrow$ "$vertorient$"
        $X_{\boldsymbol{p}} \leftarrow \text{backproject}(\boldsymbol{p}; z_c)$
      else if $s_c < \boldsymbol{p}_y < s_f$ **then**
        $a_{\boldsymbol{p}} \leftarrow o_{\boldsymbol{p}_x}$
        $w_{\boldsymbol{p}} \leftarrow \begin{bmatrix} 1 & 0 & 0 & -\text{backproject}(\boldsymbol{x_f}; z_f) \cdot \boldsymbol{e}_1 \end{bmatrix}^T$
        $X_{\boldsymbol{p}} \leftarrow \text{backproject}(\boldsymbol{p}; w_{\boldsymbol{p}})$
      else
        $a_{\boldsymbol{p}} \leftarrow$ "$vertorient$"
        $X_{\boldsymbol{p}} \leftarrow \text{backproject}(\boldsymbol{p}; z_f)$
      end if
      $d_{\boldsymbol{p}} \leftarrow (RX_{\boldsymbol{p}} + \boldsymbol{t}) \cdot \boldsymbol{e}_3$
      return
    end if
  end for

4.6 Summary

Algorithm 3 An algorithm for recovering orientation and depth for an image location $\boldsymbol{p}$ under the seam representation S.

Require: $S = \{(s_j, o_j)\}$ {a scene in seam representation}
Require: $\boldsymbol{p} = (\boldsymbol{p}_x, \boldsymbol{p}_y) \in \mathbb{Z}^2$
Ensure: $a_{\boldsymbol{p}} \in \{1, 2, 3\}, d_{\boldsymbol{p}} > 0$
  {Compute the floor–wall and ceiling–wall seam in column $\boldsymbol{p}_x$}
  $s_f \leftarrow s_{\boldsymbol{p}_x}$
  $\boldsymbol{x_f} \leftarrow \begin{bmatrix} \boldsymbol{p}_x & s_f & 1 \end{bmatrix}^T$
  $\boldsymbol{x_c} \leftarrow H_a \boldsymbol{x_f}$
  $s_c \leftarrow \frac{\boldsymbol{x_c} \cdot \boldsymbol{e}_2}{\boldsymbol{x_c} \cdot \boldsymbol{e}_3}$
  if $\boldsymbol{p}_y < s_c$ **then**
    $a_{\boldsymbol{p}} \leftarrow$ "$vertorient$"
    $X_{\boldsymbol{p}} \leftarrow \text{backproject}(\boldsymbol{p}; z_c)$
  else if $s_c < \boldsymbol{p}_y < s_f$ **then**
    $a_{\boldsymbol{p}} \leftarrow a_{\boldsymbol{p}_x}$
    $\boldsymbol{w}_{\boldsymbol{p}} \leftarrow \begin{bmatrix} 1 & 0 & 0 & -\text{backproject}(\boldsymbol{x_f}; z_f) \cdot \boldsymbol{e}_1 \end{bmatrix}^T$
    $X_{\boldsymbol{p}} \leftarrow \text{backproject}(\boldsymbol{p}; \boldsymbol{w}_{\boldsymbol{p}})$
  else
    $a_{\boldsymbol{p}} \leftarrow$ "$vertorient$"
    $X_{\boldsymbol{p}} \leftarrow \text{backproject}(\boldsymbol{p}; z_f)$
  end if
  $d_{\boldsymbol{p}} \leftarrow (RX_{\boldsymbol{p}} + \boldsymbol{t}) \cdot \boldsymbol{e}_3$

5

Recovering Orientation

5.1 Introduction

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. Equivalent to the Manhattan world assumption is the statement that there exists a "canonical" coordinate frame in which world surfaces are axis–aligned. The topic of this chapter is the estimation of this coordinate frame from a sequence of input images.

We assume that an estimate of the camera pose for each frame is available, such as might be estimated by a structure–from–motion system. We will not use any point cloud generated from structure–from–motion in this chapter. We assume that provided camera poses are measured in a fixed but arbitrary coordinate frame. In practice this coordinate frame might be determined during a SLAM initialization phase, for example. The transform to the unknown canonical coordinate frame is therefore a 3–dimensional rotation R_w, which is applied consistently to all frames.

We estimate R_w using the assumption that a non–trivial number of line segments observed in the input frames correspond to one of the three Manhattan orientations. We present a graphical model relating line segments to R_w, then we present an expectation–maximization algorithm for inference within this model.

The problem of determining R_w in the conext of a single image is equivalent to detecting vanishing points since three vanishing points and a calibrated camera give R_w in closed form. However, in the multiple view context it makes sense to estimate R_w jointly using all available information, rather than to try post–hoc merging of single–view estimates. One view of the contribution of this chapter is therefore as a generalization of vanishing point detection to multiple calibrated views.

5.2 Background

Vanishing point detection has a long history in the computer vision literature, stretching at least as far back as the seminal work of Barnard *et al.* [**?**]. In this section we survey key contributions within this literature and discuss their relationship to our work.

5.2.1 Single View

Several approaches to single–image vanishing point detection have been proposed in the literature [**?**, KZ02b, Shu99]. Common among all such approaches is the use of line segments as the fundamental observation from which inference is performed. Each of the vanishing points under consideration are assumed to generate several assocated line segments, all of which meet at the vanishing point (modulo measurement errors of course). The actual estimation of vanishing points is approached in various ways. The classic approach due to Barnard [**?**] 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 introduced explicit error models for observed line segments [Shu99], but retained the voting–based estimation strategy. Hartley and Zisserman [**?**] first proposed a generative model relating vanishing point to line segments and cast maximum likelihood estimation as a non–linear estimation problem. We make these exact probabilistic assumptions in the present work. Kanatani [**?**] minimized the algebraic deviation of the vanishing point from observed line segments, which leads to an efficient linear least–squares solution at the cost of a less precise error model. Schinder and Dellaert [**?**] 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 [CY99] 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 [KZ02b] showed that the Manhattan world assumption can be exploited even in the case of an uncalbirated 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.* [**?**] took this a step further by exploiting a prior on the rotation R_w learned from training data. Whereas Coughland and Yuille used a discretization of the space of rotations to identify R_w, they showed how to exploit the Manhattan world assumption under the more principled Expectation–Maximization approach of Schindler and Dellaert. The M–step in their approach is a non–linear optimization over rotations, leveraging the genuine maximum likeli-

hood reprojection error of Hartley and Zisserman.
Just when it seemed that vanishing point detection had reached the pinnacle of pure, relentlessly uncompromising Bayesian inference, Mirzaei and Roumeliotis [**?**] showed that the estimation can be solved globally and in closed form using an off–the–shelf polynomial solver. Their approach assumes the algebraic error and it is not clear whether the reprojection error leads to a likelihood that is amenable to a polynomial solution. We discuss these ideas further in the last section of this chapter.

5.2.2 Mulitple Views

The literature concerning multiple views is considerably more sparse, and generally involve alternate approaches to recover R_w. Furukawa and Curless [FCSS09] 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 [**?**] identify vanishing points independently in several uncalibrated views, then resolve associations between views using a simple 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 directly 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 [KZ02b, **?**], we prefer to rely on the structure–from–motion system only 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 variable z bears special explanation. It 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 element corresponds to a spurious observation not

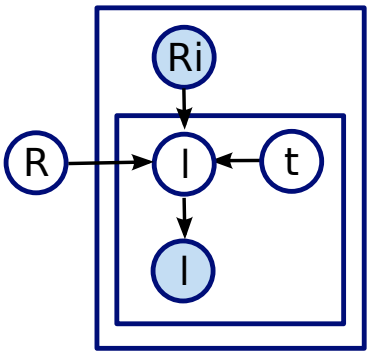


Figure 5.1: A graphical model relating line segments to vanishing points.

associated with any of the three Manhattan directions. We denote the (binary) value of the k^{th} 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 and the others set to zero. For clarity we write $z_i = \mathsf{sp}$ in place of $z_i = 4$.

The generative process operates as follows. First we sample a scene orientation R_w and the rotation component of the camera pose, R_i for each frame. (Camera translation plays no part when reasoning about vanishing points.) Next we sample each z_i from a categorical distribution with mixing parameters π. If $z_i = \mathsf{sp}$ then we the line segment $\bar{\boldsymbol{l}}$ is sampled uniformly from the image. Otherwise, we sample $\bar{\boldsymbol{l}}$ as follows.

The vanishing point $\boldsymbol{v}$ corresponding to z_i is

$$\boldsymbol{v}_{z_i} = R_i R_w \boldsymbol{e}_{z_i} \,. \tag{5.1}$$

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

$$\boldsymbol{a} \sim \mathcal{N}(\bar{\boldsymbol{a}}, \Sigma) \tag{5.2}$$

$$\boldsymbol{b} \sim \mathcal{N}(\bar{\boldsymbol{b}}, \Sigma) \,. \tag{5.3}$$

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

5.4.1 Likelihood for Line Segments

The likelihood $P(\boldsymbol{l}_i \mid z_i, R_w)$ can be simplified as follows

$$P(\boldsymbol{l}_i \mid z_i, R_w) = \int P(\boldsymbol{l}_i \mid \bar{\boldsymbol{l}})\, P(\bar{\boldsymbol{l}} \mid z_i, R_w)\, \mathrm{d}\bar{\boldsymbol{l}} \tag{5.4}$$

$$= \mathcal{N}(d(\boldsymbol{l})_i, \boldsymbol{v}_{z_i}); 0, \sigma) \tag{5.5}$$

where $d(\boldsymbol{l}, \boldsymbol{v})$ is the reprojection error illustrated in Figure 5.3. d is computed as follows. First, draw a line between the vanishing point $\boldsymbol{v}$ and the mid–point of the line segment $\boldsymbol{l}$,

$$\boldsymbol{m}_{ik} = \boldsymbol{v}_k \times \frac{\boldsymbol{a}_i + \boldsymbol{b}_i}{2} \tag{5.6}$$

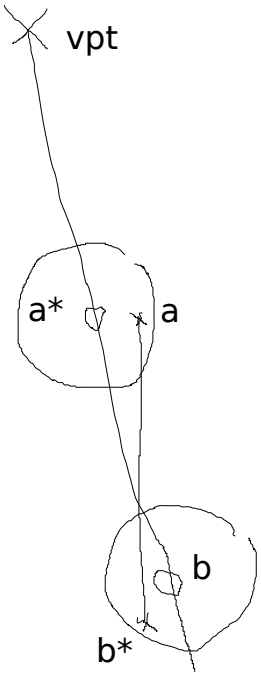


Figure 5.2: Illustration of our error model for line segments.

Now, d is the distance between the line $\boldsymbol{m}$ and either of the two end points of $\boldsymbol{l}$ (the two quantities are equal, as evident by inspecting Figure 5.3). So

$$d(\boldsymbol{l}_i, \boldsymbol{v}_k) = \frac{\boldsymbol{a}_i \cdot \boldsymbol{m}_{ik}}{(\boldsymbol{a}_i \cdot \boldsymbol{e}_3)\eta_{ik}} \tag{5.7}$$

$$\eta_{ik} = \sqrt{(\boldsymbol{m}_{ik} \cdot \boldsymbol{e}_1)^2 + (\boldsymbol{m}_{ik} \cdot \boldsymbol{e}_2)^2}\,. \tag{5.8}$$

5.4.2 Complete Data Likelihood

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

$$P(L, Z \mid R_w) = \prod_i P(\boldsymbol{l}_i, z_i \mid R_w) \tag{5.9}$$

The likelihood terms $P(\boldsymbol{l}_i, z_i \mid R_w)$ depends on the z_i in a complicated way, since $\boldsymbol{l}_i$ is drawn from completely different distributions depending on its value:

$$P(\boldsymbol{l}_i, z_i \mid R_w) = P(z_i \mid R_w)\, P(\boldsymbol{l}_i, \mid z_i, R_w) \tag{5.10}$$

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

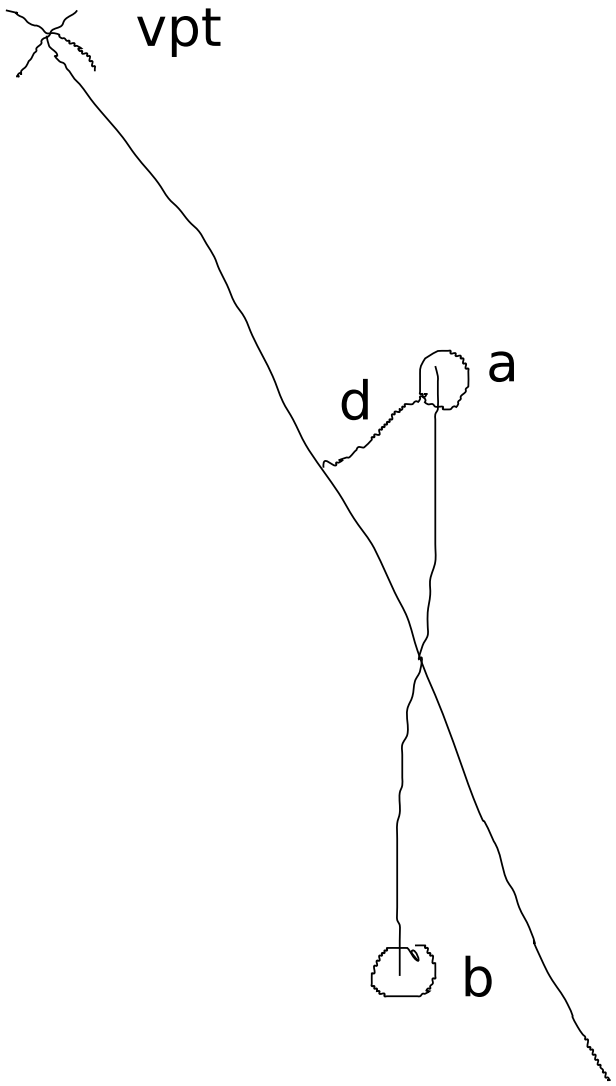


Figure 5.3: Illustration of the reprojection error.

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

$$P(\boldsymbol{l}_i, z_i \mid R_w) = \prod_{k=1}^{4} \Big(P(\boldsymbol{l}_i \mid z_i = k, R_w) P(z_i = k \mid R_w) \Big)^{z_{ik}} \tag{5.12}$$

5.5 Inference

We now turn to the inference algorithm that we use to perform a maximum–likelihood estimate of R_w under the probabilistic assumptions described in the previous section.

5.5.1 Detecting Line Segments

We begin by running the Canny edge detector [**?**], followed by an edge linking algorithm [KZ02b] to identify a set of straight line segments $\boldsymbol{l}_j = \{\boldsymbol{p} : \boldsymbol{l}_j \cdot \boldsymbol{p} = 0\}$.

5.5.2 Estimating the Manhattan Coordinate Frame

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

$$P(L \mid R_w, \{R_i\}) = \int P(L, Z \mid R_w, \{R_i\}) \, \mathrm{d}Z \; . \tag{5.13}$$

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 marginalize in order to perform inference on R_w. Integrating explicitly over z is intractible so we turn to the expectation–maximization 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 over a variable. For notational convenience we will treat q as a continuous function, though of course behind the scenes we are really just estimating 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 marginalizing it out, or defining a function over it.

The E–step

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

$$\begin{aligned} q^t(Z) &= P(Z \mid L, R_w^t) && (5.14) \\ &= \frac{\prod_i P(\boldsymbol{l}_i, z_i \mid R_w^t)}{\int \prod_i P(\boldsymbol{l}_i, z_i \mid R_w^t) \, \mathrm{d}Z} && (5.15) \\ && (5.16) \end{aligned}$$

Consider the term on the top line,

$$P(\boldsymbol{l}_i, z_i \mid R_w^t) = P(\boldsymbol{l}_i \mid z_i, R_w^t) P(z_i \mid R_w^t) \; . \tag{5.17}$$

As a function of z_i this is simply a categorical distribution, and since equation (5.15) is a normalized concatenation of categorical distributions, it is itself a categorical distribution, meaning

that the sufficient statistics for q^t are nothing more than the expectations

$$\begin{aligned}
\mathbb{E}_{q^t}\left[z_{ik}\right] &= \int z_{ik} q^t(Z)\,\mathrm{d}Z && (5.18)\\
&= \sum_{z_{ik}\in\{0,1\}} z_{ik} P(z_{ik} \mid \boldsymbol{l}_i, R_w^t) && (5.19)\\
&= P(z_i = k \mid \boldsymbol{l}_i, R_w^t) && (5.20)\\
&= \frac{P(\boldsymbol{l}_i, z_i = k \mid R_w^t)}{\sum_{j=0}^{4} P(\boldsymbol{l}_i, z_i = j \mid R_w^t)} && (5.21)
\end{aligned}$$

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),

$$\begin{aligned}
\mathbb{E}_{q^t}\left[z_{ik}\right] &= \frac{\prod_{k=1}^{4}\Big(P(\boldsymbol{l}_i \mid z_i = k, R_w^t)P(z_i = k \mid R_w^t)\Big)^{z_{ik}}}{\sum_{z_i=1}^{4}\prod_{k=1}^{4}\Big(P(\boldsymbol{l}_i \mid z_i = k, R_w^t)P(z_i = k \mid R_w^t)\Big)^{z_{ik}}} && (5.22)\\
&= \frac{\Big(\pi_{\mathsf{sp}} P(\boldsymbol{l}_i \mid z_i = \mathsf{sp}, R_w^t)\Big)^{z_{ik}} \prod_{k=1}^{3}\Big(\pi_k P(\boldsymbol{l}_i \mid z_i = k, R_w^t)\Big)^{z_{ik}}}{\sum_{z_i=1}^{4}\left[\Big(\pi_{\mathsf{sp}} P(\boldsymbol{l}_i \mid z_i = \mathsf{sp}, R_w^t)\Big)^{z_{ik}} \prod_{k=1}^{3}\Big(\pi_k P(\boldsymbol{l}_i \mid z_i = k, R_w^t)\Big)^{z_{ik}}\right]} && (5.23)
\end{aligned}$$

Substituting

$$\begin{aligned}
\rho_i &= \pi_{\mathsf{sp}} P(\boldsymbol{l}_i \mid z_i = \mathsf{sp}, R_w^t) && (5.24)\\
s_i &= \begin{cases} 1, & \text{if } z_i = \mathsf{sp} \\ 0, & \text{otherwise} \end{cases} && (5.25)\\
& && (5.26)
\end{aligned}$$

into (5.23) we have

$$\begin{aligned}
\mathbb{E}_{q^t}\left[z_{ik}\right] &= \frac{\rho_i^{s_i}\Big(\pi_{z_i}\mathcal{N}(d(\boldsymbol{l}, R_i R_w^t \boldsymbol{e}_k); 0, \sigma)\Big)^{1-s_i}}{\sum_{z_i=1}^{4}\rho_i^{s_i}\Big(\pi_{z_i}\mathcal{N}(d(\boldsymbol{l}, R_i R_w^t \boldsymbol{e}_{z_i}); 0, \sigma)\Big)^{1-s_i}} && (5.27)\\
&= \frac{\rho_i^{s_i}\Big(\pi_{z_i}\mathcal{N}(d(\boldsymbol{l}, R_i R_w^t \boldsymbol{e}_k); 0, \sigma)\Big)^{1-s_i}}{\rho_i + \sum_{z_i=1}^{3}\Big(\pi_{z_i}\mathcal{N}(d(\boldsymbol{l}, R_i R_w^t \boldsymbol{e}_{z_i}); 0, \sigma)\Big)}\,. && (5.28)
\end{aligned}$$

Clearly 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.

M–step

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

$$\log P(L, Z \mid R_w) = \sum_i \log P(\boldsymbol{l}_i, z_i \mid R_w) \; . \tag{5.29}$$

Substituting (5.12) gives

$$\begin{aligned}
\log P(L, Z \mid R_w) &= \sum_i \sum_{k=1}^{4} z_{ik} \Big(\log P(\boldsymbol{l}_i \mid z_i = k, R_w) + \log P(z_i = k \mid R_w) \Big) && (5.30)\\
&= \sum_i \sum_{k=1}^{4} z_{ik} \Big(\log \mathcal{N}(d(\boldsymbol{l}_i, R_i R_w \boldsymbol{e}_k); 0, \sigma) + \log \pi_k \Big) && (5.31)\\
&= \sum_i \sum_{k=1}^{4} z_{ik} \Big(\frac{-d(\boldsymbol{l}_i, R_i R_w \boldsymbol{e}_k)^2}{2\sigma^2} + \log \pi_k \Big) + c && (5.32)\\
& && (5.33)
\end{aligned}$$

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

$$\begin{aligned}
\mathbb{E}_{q^t} \Big[\log P(L, Z \mid R_w) \Big] &= \sum_i \sum_{k=1}^{4} \mathbb{E}_{q^t} \big[z_{ik} \big] \Big(\frac{-d(\boldsymbol{l}_i, R_i R_w \boldsymbol{e}_k)^2}{2\sigma^2} + \log \pi_k \Big) && (5.34)\\
&= \sum_i \sum_{k=1}^{4} r_{ik}^t \Big(\frac{-d(\boldsymbol{l}_i, R_i R_w \boldsymbol{e}_k)^2}{2\sigma^2} + \log \pi_k \Big) && (5.35)
\end{aligned}$$

where in the last line we have defined and substituted

$$r_{ik}^t = \mathbb{E}_{q^t} \big[z_{ik} \big] \; . \tag{5.36}$$

The maximization we wish to perform is

$$R_w^{t+1} = \operatorname*{argmax}_{R_w} \mathbb{E}_{q^t} \Big[\log P(L, Z \mid R_w) \Big] \; . \tag{5.37}$$

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 maximized in this step. Hence the expression being maximized in (5.37) is a function of R_w alone. Substituting (5.35),

$$\begin{aligned}
R_w^{t+1} &= \operatorname*{argmax}_{R_w} f(R_w) && (5.38)\\
f(R_w) &= \sum_i \sum_{k=1}^{4} r_{ik}^t \Big(\frac{-d(\boldsymbol{l}_i, R_i R_w \boldsymbol{e}_k)^2}{2\sigma^2} + \log \pi_k \Big) \; . && (5.39)
\end{aligned}$$

Importantly, note that r_{ik}^t is a constant with respect to R_w in the above. We perform the maximization (5.37) by gradient descent. Differentiating (5.39) with respect to R_w,

$$\frac{\partial f}{\partial R_w} = \sum_i \sum_{k=1}^{4} r_{ik}^t \Big(\frac{-d_{ik}}{\sigma^2} \frac{\partial d_{ik}}{\partial R_w} \Big) \tag{5.40}$$

$$\frac{\partial d_{ik}}{\partial R_w} = \frac{1}{\boldsymbol{a}_i \cdot \boldsymbol{e}_3} \left(\frac{(\boldsymbol{m}_{ik})_1 \boldsymbol{e}_1 + (\boldsymbol{m}_{ik})_2 \boldsymbol{e}_2}{{\eta_{ik}}^3} (\boldsymbol{m}_{ik} \cdot \boldsymbol{a}_i) - \frac{\boldsymbol{a}_i}{\eta_{ik}} \right) \Big[\frac{\boldsymbol{a}_i + \boldsymbol{b}_i}{2} \Big]_\times \frac{\partial \boldsymbol{v}_k}{\partial R_w} \tag{5.41}$$

where

$$d_{ik} = d(\boldsymbol{l}_i, R_i R_w \boldsymbol{e}_k) \tag{5.42}$$

$$\eta_{ik} = \sqrt{(\boldsymbol{m}_{ik} \cdot \boldsymbol{e}_1)^2 + (\boldsymbol{m}_{ik} \cdot \boldsymbol{e}_2)^2} \tag{5.43}$$

and

$$\big[\boldsymbol{x}\big]_\times = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix} \tag{5.44}$$

is the matrix form for the cross product.

For the purpose of optimization 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 as a member of the special orthogonal group $SO(3)$, so

$$R_w = R_0 R_\delta \tag{5.45}$$

$$R_\delta = \exp\Big(\sum m_i G_i \Big) \tag{5.46}$$

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 $\boldsymbol{v}$ with respect to $\boldsymbol{m}$ yields

$$\frac{\partial \boldsymbol{v}_k}{\partial \boldsymbol{m}} = R_i \frac{\partial R_w}{\partial \boldsymbol{m}} \boldsymbol{e}_k \, . \tag{5.47}$$

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

$$\left. \frac{\partial \boldsymbol{v}_k}{\partial \boldsymbol{m}} \right|_{\boldsymbol{m}=0} = \begin{bmatrix} R_i R_0 G_1 \boldsymbol{e}_k & R_i R_0 G_2 \boldsymbol{e}_k & R_i R_0 G_3 \boldsymbol{e}_k \end{bmatrix} \tag{5.48}$$

This completes the derivation of the gradient of the error function f with respect to the parametrisation $\boldsymbol{m}$. We proceed as normal with gradient steps of the form

$$\boldsymbol{m}^{t+1} = \boldsymbol{m}^t - \gamma \frac{\partial f}{\partial \boldsymbol{m}} \, . \tag{5.49}$$

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

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

Figure 5.4: 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 point–cloud procedure of [FCSS09] (NRM), Expectation–Maximization using the algebraic error (ALG), Expectation–Maximization 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.

quickly and robustly. Our algorithm initializes 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.

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 algebra $SO(3)$. 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.

Generalized EM

TODO

5.6 Results

Figure **??** shows the vanishing points identified in one of our sequences. Since each frame is informed by the entire sequence we are able to identify a globally consistent coordinate frame where single–image vanishing point detection fails. Figure **??** shows a side–by–side comparison with the single–image vanishing point detector of [KZ02b]. Recently proposed improvements to the single–image approach [Tar09] may improve slightly on these, but we found that in cases where the single–image approach fails there is often simply not enough information available in individual frames to identify the appropriate coordinate frame, so any single–image approach will necessarily fail.

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.

5.7.1 Identifying the floor and ceiling planes.

TODO An indoor Manhattan scene has exactly one floor and one ceiling plane, both with normal direction $\boldsymbol{v_v}$. It will be useful in the following sections to have available the mapping H_a between the image locations of ceiling points and the image locations of the floor points that are vertically below them (see Figure **??**). H_a is a planar homology with axis $\boldsymbol{h} = \boldsymbol{v_l} \times \boldsymbol{v_r}$ and vertex $\boldsymbol{v_v}$ [Cri01] and can be recovered given the image location of any pair of corresponding floor/ceiling points $(\boldsymbol{x_f}, \boldsymbol{x_c})$ as

$$H_a = I + \mu \frac{\boldsymbol{v_v} \boldsymbol{h}^T}{\boldsymbol{v_v} \cdot \boldsymbol{h}} \,, \tag{5.50}$$

where $\mu = \langle \boldsymbol{v_v}, \boldsymbol{x_c}, \boldsymbol{x_f}, \boldsymbol{x_c} \times \boldsymbol{x_f} \times \boldsymbol{h} \rangle$ is the characteristic cross ratio of H_a.
Although we do not have *a priori* any such pair $(\boldsymbol{x_f}, \boldsymbol{x_c})$, we can recover H_a using the following RANSAC algorithm. First, we sample one point $\hat{\boldsymbol{x}}_{\boldsymbol{c}}$ from the region above the horizon in the Canny edge map, then we sample a second point $\hat{\boldsymbol{x}}_{\boldsymbol{f}}$ collinear with the first and $\boldsymbol{v_v}$ from the region below the horizon. We compute the hypothesis map $\hat{H}_a$ as described above, which we then score by the number of edge pixels that $\hat{H}_a$ maps onto other edge pixels (according to the Canny edge map). 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 RANSAC 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.

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

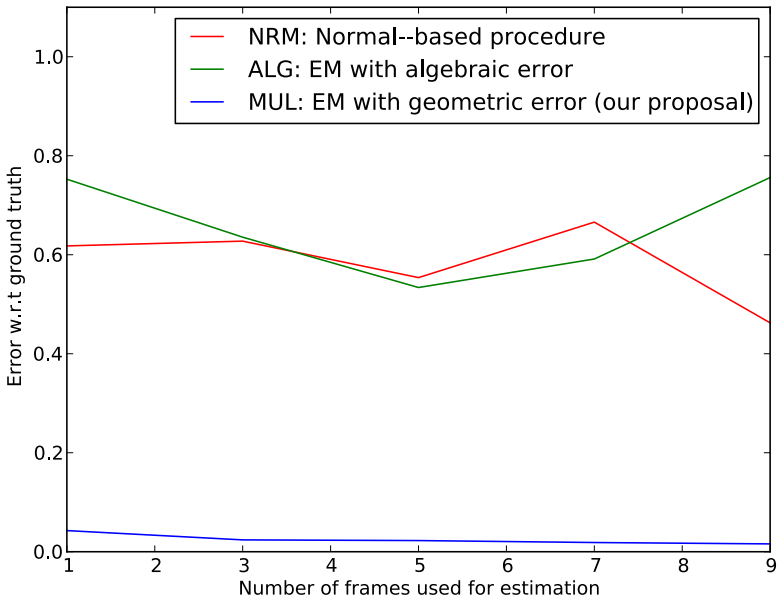
$$\boldsymbol{u}_j = R_w^{-T} R_i^{-T} \boldsymbol{l}_j \times \boldsymbol{e}_3 \,, \tag{5.51}$$

which would be its vanishing point if it were horizontal in the world. Vertical surfaces of a given orientation will generate identical $\boldsymbol{u}_j$ (modulo measurement error), so we may identify additional vertical orientations by clustering the intersections $\{\boldsymbol{u}_j\}$. We adopt a voting algorithm in which we parametrise $\boldsymbol{u}_j$ in terms of the angle θ_j about the z axis

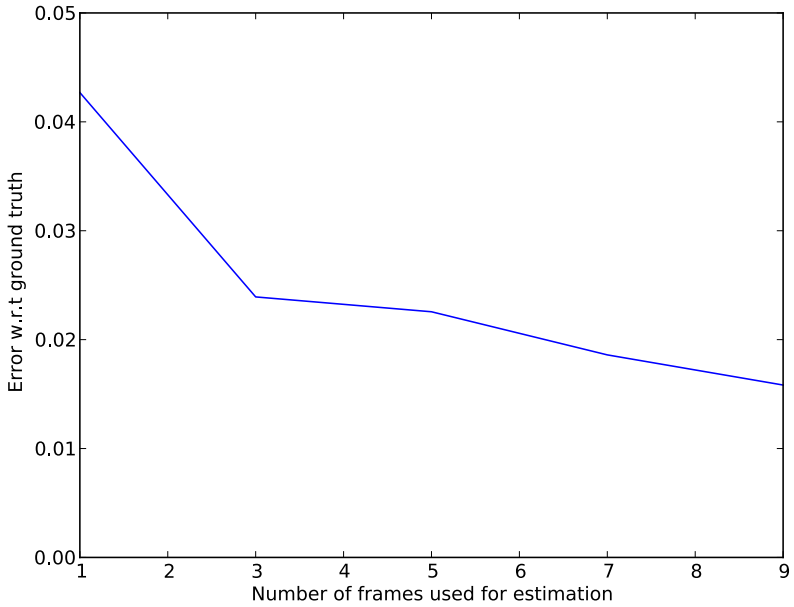
$$\theta_j = \text{atan}(\boldsymbol{e}_2{}^T \boldsymbol{u}_j,\ \boldsymbol{e}_1{}^T \boldsymbol{u}_j) \,. \tag{5.52}$$

We accumulate the θ_j into histogram bins and identify any local maxima θ_i^* above a threshold k. Each θ_i^* represents a cluster of line segments corresponding to an additional vertical orientation. Finally, we re–estimate the vanishing point for each cluster by minimising the likelihood (**??**) via least–squares.

5.9 Guided Line Search

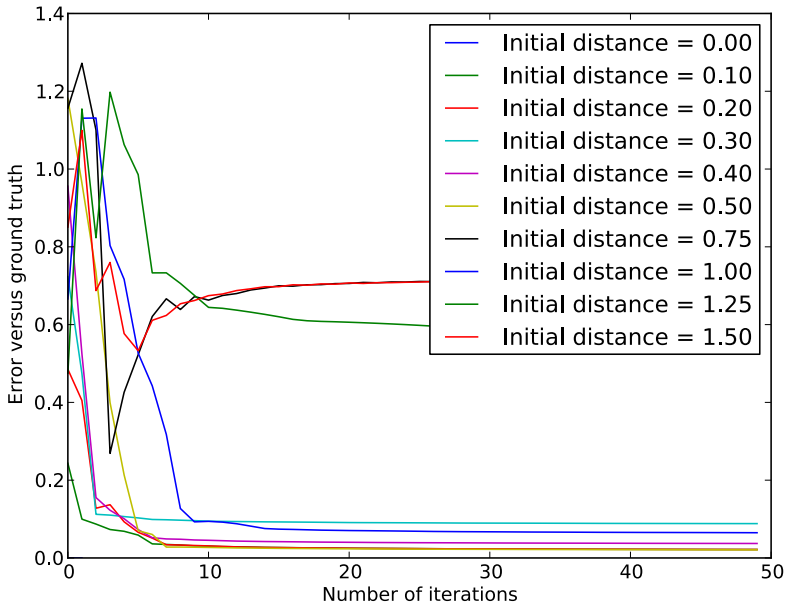


(a) Estimation error versus number of input frames.

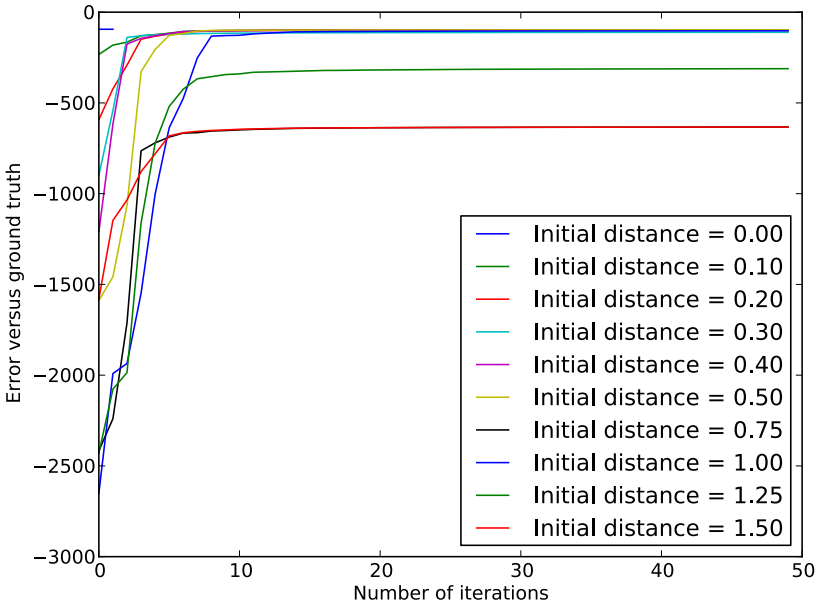


(b) A magnification of the third series above.

Figure 5.5: 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.

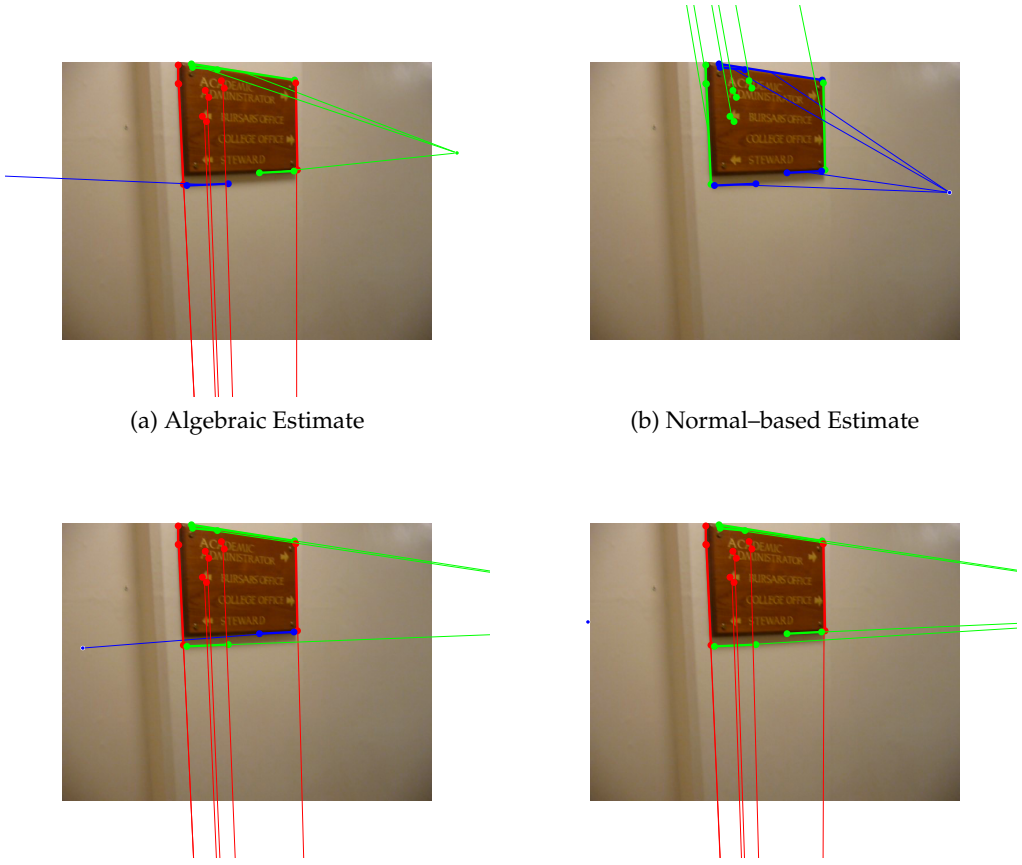


(a) Convergence of error



(b) Convergence of log–likelihood

Figure 5.6: 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 initialized a particular distance from the ground truth. We see that for distances less than 1 convergence is robust. This distance corresponds to a nearly maximal difference between the initial and ground truth rotation, indicating that the optima has a wide basin of attraction. Note that for this experiment the labels Z are known.



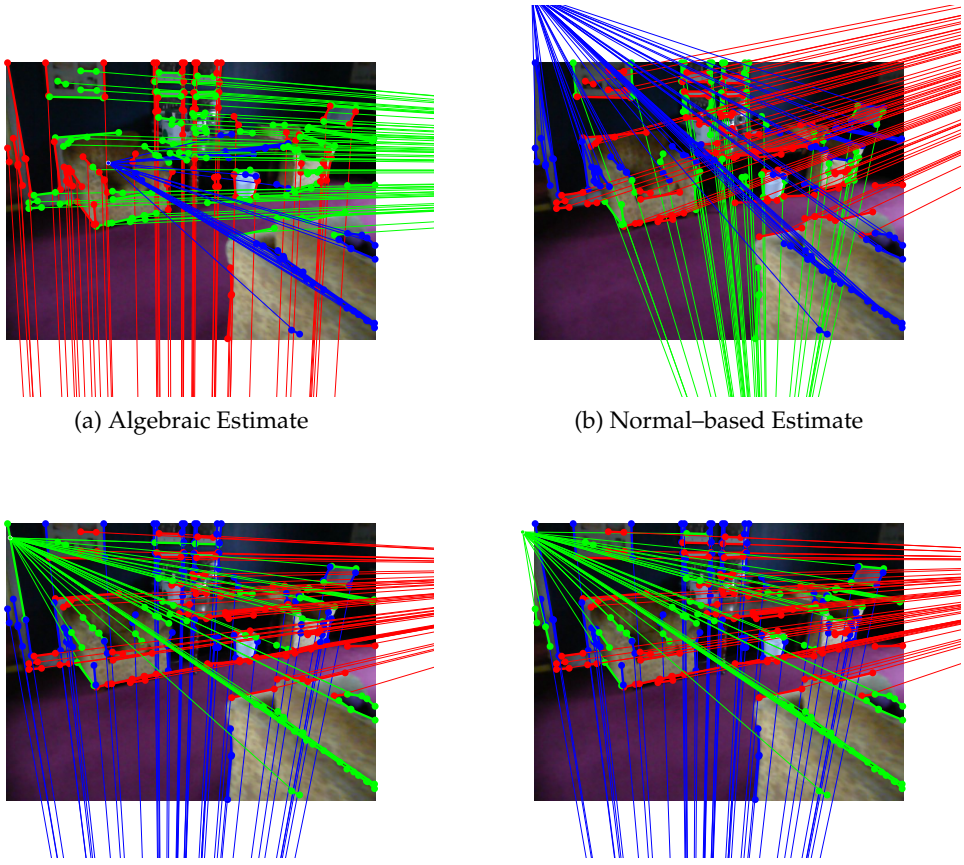
(a) Algebraic Estimate

(b) Normal–based Estimate

(c) Single Image Estimate

(d) Geometric (Proposed) Estimate

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



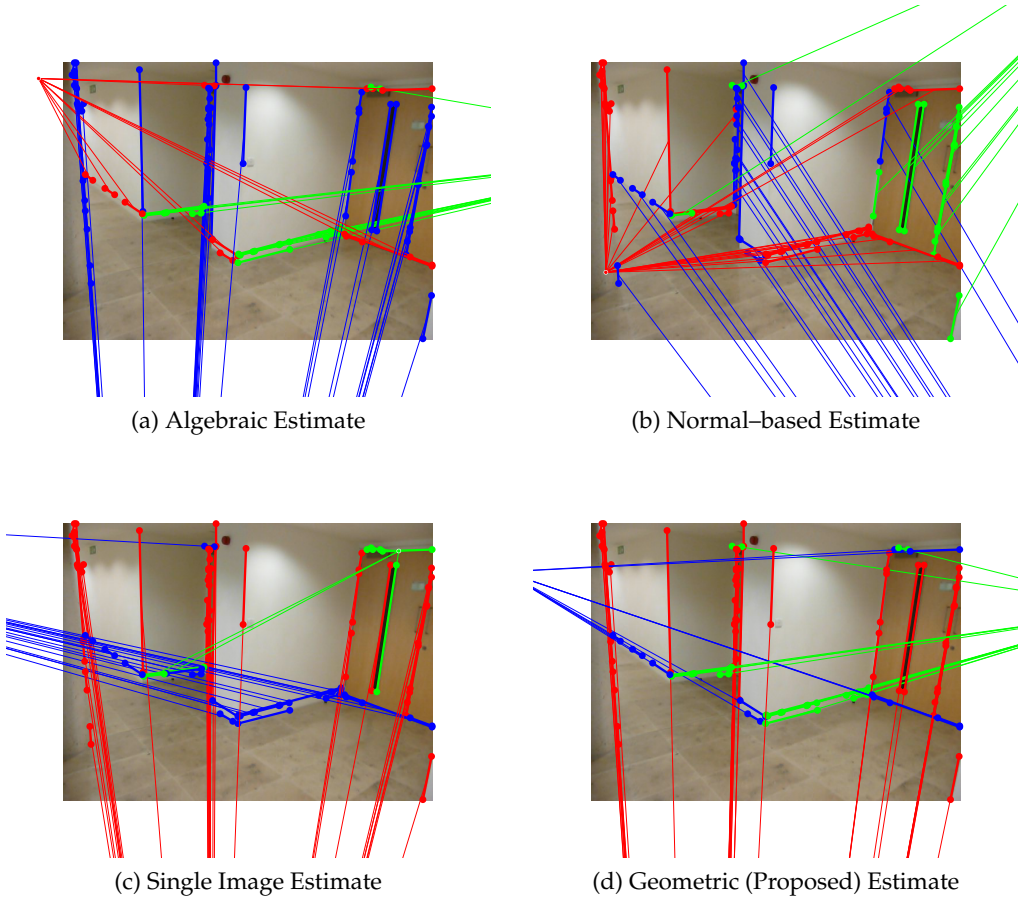
(a) Algebraic Estimate

(b) Normal–based Estimate

(c) Single Image Estimate

(d) Geometric (Proposed) Estimate

Figure 5.8: 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.



(a) Algebraic Estimate

(b) Normal–based Estimate

(c) Single Image Estimate

(d) Geometric (Proposed) Estimate

Figure 5.9: 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.

6

Probabilistic Models for Manhattan Worlds

6.1 Introduction

In this section we describe probabilistic models relating indoor Manhattan model to observations. We consider three sensor modalities: monocular image features, stereo features, and 3D point clouds. For each we present a generative model relating observed features to the Manhattan scene structure, which we denote M. We show in each case that the logarithm of the posterior $P(M \mid X)$ be written as a column–wise sum over a payoff function $\pi(x, y)$ together with a regularizer. This allows us to present a unified inference algorithm in chapter 7, which efficiently solves both maximum–apsteriori and maximum–likelihood inference for each sensor modality, as well as for the combination of all three.
Rather than presenting a graphical model at the outset as in the previous chapter, here we prefer to present models for each sensor modality one–by–one to aid comprehension, then present the joint model at the end.

6.2 Payoff Formulation

In this section we describe a class of optimization problems that we will refer to as *payoff form*. Throughout the remainder of this chapter it will become clear that a range of inference prob-

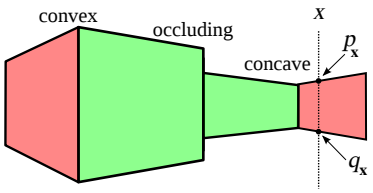


Figure 6.1: Each corner in an indoor Manhattan environment can be categorized as concave, convex, or occluding. Each vertical line intersects exactly one wall segment. We denote the intersection of column x with the top and bottom of the wall in that column $\boldsymbol{p}_x$ and the $\boldsymbol{q}_x$ respectively.

lems in the context of indoor Manhattan scenes can be expressed in this form, and in the following chapter it will become clear 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 $\pi : \mathcal{D} \times \mathcal{A} \to \mathbb{R}$ maps orientations $a \in \mathcal{A}$ together with integer pixel coordinates $\boldsymbol{p} \in \mathcal{D} \subset \mathbb{Z}^2$ to real numbers. The payoff for a scene M is defined in terms of the seam representation,

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

where $S = \{(s_j, o_j)\}$. In cases where the payoff function is independent of a_j we will write

$$\pi(\boldsymbol{p}) = \pi(\boldsymbol{p}_x, \boldsymbol{p}_y) \tag{6.2}$$

Figure XXX shows several example of the seam over which this sum is computed.

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

Next we define a penalty function $\Gamma : \mathcal{M} \to \mathbb{R}$ in terms of the vertex representation,

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

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. Finally, we define our objective

$$\begin{aligned} f(M) &= \Pi(M) - \Gamma(M) && (6.4) \\ &= \sum_{j=1}^{W} \pi(j, s_j, o_j) - \sum_{i=0}^{K-1} \gamma(i; M)\ . && (6.5) \end{aligned}$$

which we will optimize as in

$$\hat{M} = \operatorname*{argmax}_{M \in \mathcal{M}} f(M)\ . \tag{6.6}$$

Note that the objective f is defined partially in the scene representation and partially in the vertex representation. This poses no difficulty to evaluating the objective in the vertex representation since we can readily obtain the seam representation via equation (4.61). However, as discussed in Section **??** 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 optimization algorithm in Chapter 7, but for convenience we will work in seam representation to define the payoff Π, which is the topic of the remainder of this chapter.

6.3 Model For Estimating Manhattan Homology

6.4 Prior On Scenes

We turn first to the prior on models, $P(M \mid \boldsymbol{\lambda})$, which is governed by the hyper–parameters $\boldsymbol{\lambda}$. As discussed in Section 4.5.1, corners between successive walls can be clasified 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 $\boldsymbol{n}$,

$$P(M \mid \boldsymbol{\lambda}) = \frac{1}{Z} {\lambda_1}^{n_1} {\lambda_2}^{n_2} {\lambda_3}^{n_3} \tag{6.7}$$

$$Z = \frac{\mu(n_1, n_2, n_3)}{(1-\lambda_1)(1-\lambda_2)(1-\lambda_3)}\,, \tag{6.8}$$

where $\mu(n_1, n_2, n_3)$ measures the number of scenes with a certain number concave, convex, and occluding corners. The prior (6.7) may be interpreted as a process in which n_i is determined by the number of times that a biased coin comes up heads before the first observed tails, where the probability of heads in each round is λ_i. Our choice of (6.7) is motivated by the desire to penalize scenes for additional complexity, which we take to correspond roughly to the number walls. We discuss alternative priors, and some of the problems they raise during inference, in Section **??**.
Taking logarithms yields

$$\log P(M \mid \boldsymbol{\lambda}) = -\log Z + n_1 \log \lambda_1 + n_2 \log \lambda_2 + n_3 \log \lambda_3\,. \tag{6.9}$$

Comparison with (6.3) suggests the following penalty function:

$$\gamma(i; M) = \begin{cases} \log \lambda_1, & \text{if corner } i \text{ is concave} \\ \log \lambda_2, & \text{if corner } i \text{ is concex} \\ \log \lambda_3, & \text{if corner } i \text{ is occluding.} \end{cases} \tag{6.10}$$

The three cases can be decided by the algorithm described in Section 4.5.1. Note that the scene penalty differs from the log–posterior by a additive constant,

$$\Gamma(M) = \log P(M \mid \boldsymbol{\lambda}) + \log Z \tag{6.11}$$

but we will ignore this term since our ultimate goal is the optimization expressed in (6.6).

6.5 Photometric Sensor Model

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

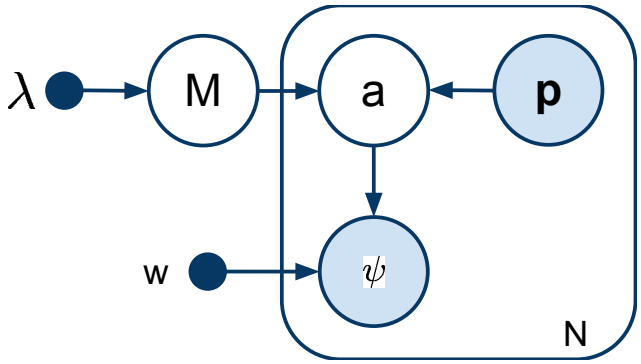


Figure 6.2: The graphical model relating scenes M to monocular image features $\boldsymbol{\phi}$. $\boldsymbol{p} = (x, y)$ is a pixel location and a is the orientation predicted (deterministically) by M at $\boldsymbol{p}$.

We assume the graphical model shown in Figure 6.2. The generative process begins by sampling a scene M, then samples pixels $\boldsymbol{p}$ and computes their orientation $a \in \{1, 2, 3\}$ (*c.f.* Algorithm **??**), which is deterministic given M and is included in the graphical model for notational convenience only. Finally a feature $\boldsymbol{\phi}$ is sampled from a distribution that is conditionally independent of M given a.
We assume a exponential–family likelihood for pixel features,

$$P(\boldsymbol{\phi} \mid a) \propto \exp(\boldsymbol{\omega}_a \cdot \boldsymbol{\phi}) \,. \tag{6.12}$$

Denoting the set of all observed pixel features Φ, pixels $\boldsymbol{P}$, and orientations A, the joint distribution is

$$P(\Phi, \boldsymbol{P}, A, M, \boldsymbol{\lambda}, \boldsymbol{\omega}) = P(M \mid \boldsymbol{\lambda}) \prod P(\boldsymbol{p}_i) P(a_i \mid M, \boldsymbol{p}_i) P(\boldsymbol{\phi}_i \mid a_i, \boldsymbol{\omega}) \,. \tag{6.13}$$

We do not model the distribution $P(\boldsymbol{p}_i)$ 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

$$P(\Phi \mid M) \propto \int \prod_i P(\boldsymbol{\phi}_i \mid a_i) P(a_i \mid M) \mathrm{d}A \,. \tag{6.14}$$

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 **??**. Therefore, denoting by a_i^* the orientation output by Algorithm **??** for pixel $\boldsymbol{p}_i$ under M we have

$$P(\Phi \mid M) \propto \prod_i P(\boldsymbol{\phi}_i \mid a_i^*) \,. \tag{6.15}$$

Taking logarithms gives

$$\log P(\Phi \mid M) = \sum_i \log P(\boldsymbol{\phi}_i \mid a_i^*) + c \tag{6.16}$$

where c corresponds to the constant of proportionality in (6.15), which we henceforth drop since it makes no difference to the optimization to come.

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

$$\pi_{\mathsf{mono}}(x, y, o) = \sum_{r=0}^{H} \log P(\boldsymbol{\phi}_{xr} \mid \bar{a}(x, r; y, o)\ ) \tag{6.17}$$

where the double–subscript in $\boldsymbol{\phi}_{xr}$ is a result of separately indexing rows and columns in . Now consider the scene payoff,

$$\begin{aligned} \Pi(M) &= \sum_{j=0}^{W} \pi_{\mathsf{mono}}(j, s_j, o_j) && (6.18)\\ &= \sum_{j=0}^{W} \sum_{r=0}^{H} \log P(\boldsymbol{\phi}_{jr} \mid \bar{a}(j, r; s_j, o_j)\ ) && (6.19)\\ &= \log P(\Phi \mid M) - c\,, && (6.20) \end{aligned}$$

which is simply the log–likelihood (6.16).

6.5.1 Features

RGB
HSV
Gabor

Line Sweeps

This feature adapts the simple and efficient line–sweep algorithm of Lee *et al.* [DCLK09], which produces a partial labelling of the image in terms of the three Manhattan surface orientation labels (corresponding to the three Manhattan orientations in the image).

6.6 Multiple–View Sensor Model

We now formulate the payoff function π_{stereo} for the case that multiple views of the scene are available. 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 8.3 that models parametrized in image coordinates specify unique 3D models. Any hypothesized model 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 $\mathrm{PC}(\cdot)$, which provides the likelihood $P(I_0, \ldots, I_M \mid M)$. The prior remains as in (6.7).
Optimizing over photo–consistency has been standard in the stereo literature for several decades [SS01]; our contribution is to show that (i) in the particular case of indoor Manhattan models,

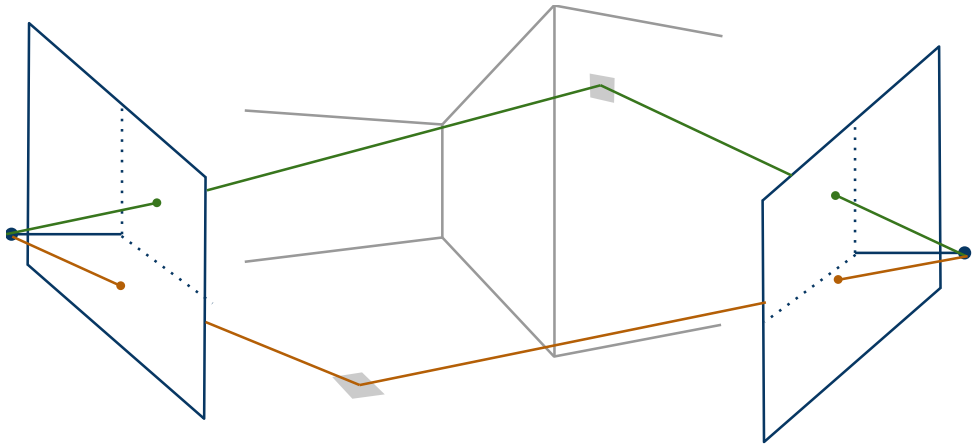


Figure 6.3: Pixel correspondences across multiple views are computed by back–projection onto the model M followed by re–projection into auxiliary views.

photo–consistency can be expressed as a payoff matrix; (ii) that we can therefore perform efficient and exact global optimization; and (iii) that this fits naturally within a Bayesian framework alongside monocular and 3D features.

We assume that there one frame is identified as the base frame I_0; the remaining K frames are denoted $I_1, \ldots, I_K$. We assume that all cameras are fully calibrated with the action of the i^{th} camera on a 3D points X being

$$H_i(R_i X + \boldsymbol{t}_i) \; . \tag{6.21}$$

We normalize images intensities to zero mean and unit variance.

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 use such a reconstruction to re–project any pixel $\boldsymbol{p}$ in the base view into any auxiliary view, as illustrated in Figure 6.3. Let $\boldsymbol{q}_k(\boldsymbol{p}; M)$ be the re–projection of $\boldsymbol{p}$ into view k via the scene hypothesis M. Let each pixel $\boldsymbol{p}_i$ in the base image be associated with a feature vector $\boldsymbol{\phi}_i \in \mathbb{R}^n$ and let each pixel $\boldsymbol{q}_{ki}$ in the k^{th} auxiliary view be simirlarly associated with a feature $\boldsymbol{\theta}_{ki} \in \mathbb{R}^n$. Let Φ be the set of all features in the base view and let Θ_k be the set of all features in the k^{th} auxiliary view. Finally, let

$$\bar{\boldsymbol{\theta}}_k(\boldsymbol{p}_i; M) = \boldsymbol{\theta}(\boldsymbol{q}_k(\boldsymbol{p}_i; M)) \tag{6.22}$$

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

$$P(\Phi, \Theta_1, \ldots, \Theta_K \mid M) = \prod_i \prod_{k=0}^{K} P(\boldsymbol{\phi}_i, \bar{\boldsymbol{\theta}}_k(\boldsymbol{p}_i; M) \mid \Sigma) \; . \tag{6.23}$$

and the feature likelihood is a zero–mean Gaussian, which is standard in the literature [SS01]:

$$P(\boldsymbol{\phi}_i, \bar{\boldsymbol{\theta}}_k(\boldsymbol{p}_i; M) \mid \Sigma) = \mathcal{N}(\boldsymbol{\phi}_i - \bar{\boldsymbol{\theta}}_k(\boldsymbol{p}_i; M); \Sigma) \tag{6.24}$$

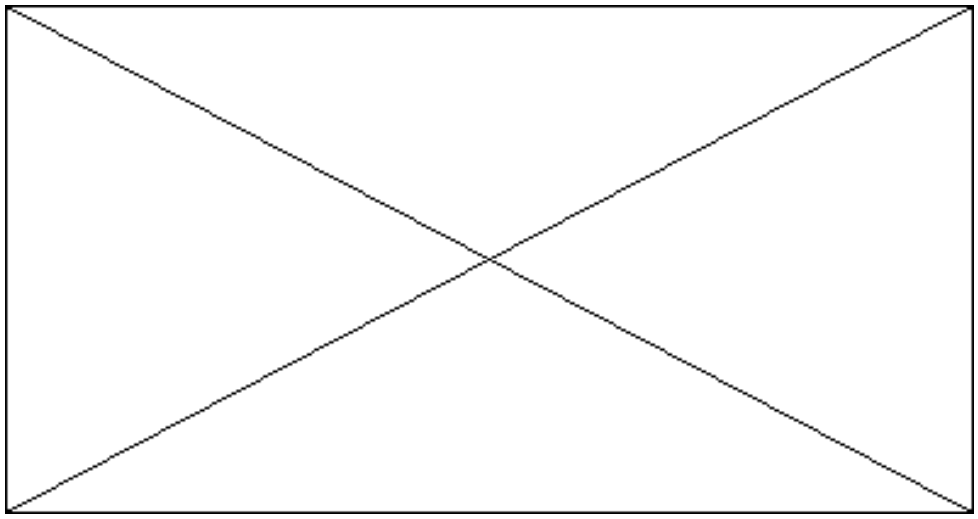


Figure 6.4: TODO: Reprojected image columns.

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

$$\log P(\Phi, \Theta_1, \ldots, \Theta_K \mid M) = \sum_i \sum_{k=0}^{K} \|\boldsymbol{\phi}_i - \bar{\boldsymbol{\theta}}_k(\boldsymbol{p}_i; M)\|_\Sigma + c \tag{6.25}$$

We now write equation (6.25) in payoff form. To this end we leverage the observation made in Section 4.5.4 that the depth of a pixel $\boldsymbol{p}$ under a scene hypothesis M is functionally dependent only on the pair (s_j, o_j) for the column j containing $\boldsymbol{p}$. Let $\bar{d}(\boldsymbol{p}; s_j)$ be the depth of $\boldsymbol{p}$ under hypothesis s_j, as output by Algorithm **??**.

Furthermore, re–projecting a pixel into an auxiliary view requires only the depth d_i of the scene at the pixel together with the camera parameters. Therefore we can identify the correspondences for any pixel $\boldsymbol{p}$ in all auxiliary views from the value s_j alone. Specifically, the re–projection of $\boldsymbol{p}$ into the k^{th} view under the hypothesis s_j is

$$\boldsymbol{q}_k(\boldsymbol{p}; s_j) = H_k(R_k X_i + \boldsymbol{t}_k) \tag{6.26}$$

where

$$X_i = R_0^{-1}\big(\bar{d}(\boldsymbol{p}; s_j) H_0^{-1}\boldsymbol{p} - \boldsymbol{t}_0\big) \,. \tag{6.27}$$

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

$$\bar{\boldsymbol{\theta}}_k(\boldsymbol{p}; s_j) = \boldsymbol{\theta}(\boldsymbol{q}_k(\boldsymbol{p}; s_j)) \,. \tag{6.28}$$

Finally, the payoff function is

$$\pi_{\mathsf{stereo}}(x, y) = \sum_{r=0}^{H} \sum_{k=1}^{K} \|\boldsymbol{\phi}_{xr} - \bar{\boldsymbol{\theta}}_k([x, r]^T; y)\|_\Sigma \tag{6.29}$$

where as in (6.5) we have switched to a separate indexing scheme for rows and columns, so $\boldsymbol{\phi}_{xy}$ is the feature for the pixel at (x, y) and $\bar{\boldsymbol{\theta}}_k(x, r; y)$ the the feature for the reprojection of (x, r) into the k^{th} view under the hypothesis $s_j = y$.

Substituting (6.29) into (6.1),

$$\Pi(M) = \sum_{j=0}^{W} \pi_{\text{stereo}}(j, s_j) \tag{6.30}$$

$$= \sum_{j=0}^{W} \sum_{r=0}^{H} \sum_{k=1}^{K} \|\boldsymbol{\phi}_{jr} - \bar{\boldsymbol{\theta}}_k([j, r]^T; s_j)\|_\Sigma \tag{6.31}$$

$$= \log P(\Phi, \Theta_1, \ldots, \Theta_K \mid M) - c \tag{6.32}$$

6.6.1 An Alternative 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 various pixel–wise smoothness constraints, our prior is (6.7) for disparity maps corresponding to valid indoor Manhattan reconstructions, and zero otherwise. Inference under this model would be intractible 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 (4.59) the problem becomes tractible.
Note that the column–wise decomposition (6.29) neither commits us to optimizing over columns independently, nor to ignoring interactions between columns. Indeed by inspecting the graphical model in Figure **??** one sees immediately that our model assumes no independence between image columns (only conditional independence given M). Such correlations come into effect when we optimize over the full payoff matrix in Chapter 7, and 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 the following.

6.6.2 Occlusions

. We have ignored self–occlusions in (**??**). 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.

6.7 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 $\boldsymbol{p}_i$. Many such measure-

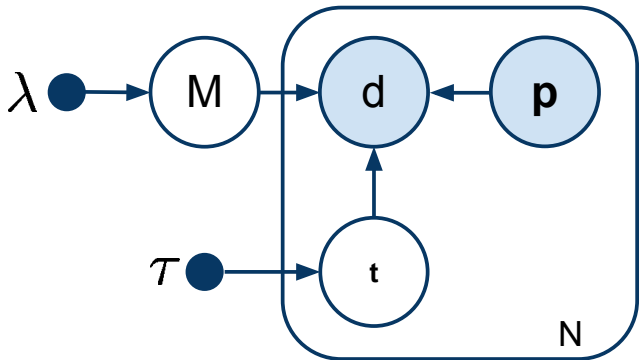


Figure 6.5: The graphical model relating indoor Manhattan models to 3D points. The hidden variable t indicates whether the point is inside, outside, or coincident with the model.

ments 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 following interpretation. If $t_i = \mathsf{ON}$ then d_i corresponds to some surface represented explicitly in M. Otherwise, either $t_i = \mathsf{IN}$, meaning some clutter object within the room was measured, or $t_i = \mathsf{OUT}$, in which case an object outside the room was measured, such as through a window. The likelihoods we use are

$$P(d \mid \boldsymbol{p}, t = \mathsf{IN}, M) = \begin{cases} \alpha, & \text{if } 0 < d < \bar{d}(\boldsymbol{p}; M) \\ 0, & \text{otherwise} \end{cases} \tag{6.33}$$

$$P(d \mid \boldsymbol{p}, t = \mathsf{OUT}, M) = \begin{cases} \beta, & \text{if } \bar{d}(\boldsymbol{p}; M) < d < N_d \\ 0, & \text{otherwise} \end{cases} \tag{6.34}$$

$$P(d \mid \boldsymbol{p}, t = \mathsf{ON}, M) = \mathcal{N}(d\,;\, \bar{d}(\boldsymbol{p}; M), \sigma)\,. \tag{6.35}$$

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

$$P(d \mid \boldsymbol{p}, M, \boldsymbol{\tau}) = \sum_t P(d \mid \boldsymbol{p}, M, t) P(t \mid \boldsymbol{\tau})\,. \tag{6.36}$$

where $P(t \mid \boldsymbol{\tau})$ is a categorical distribution with parameters τ_{IN}, τ_{OUT}, and τ_{ON}. Equation (6.36) can be readily evaluated for any d and $\boldsymbol{p}$ 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

$$P(D \mid \boldsymbol{P}, M) = \prod_i P(d_i \mid \boldsymbol{p}_i, M) \tag{6.37}$$

$$\log P(D \mid \boldsymbol{P}, M) = \sum_i \log P(d_i \mid \boldsymbol{p}_i, M) \tag{6.38}$$

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

$$\pi_{\mathsf{3D}}(x, y) = \sum_{i \in D_x} \log P(d_i \mid \boldsymbol{p}_i, \bar{d}(\boldsymbol{p}_i; y)) \tag{6.39}$$

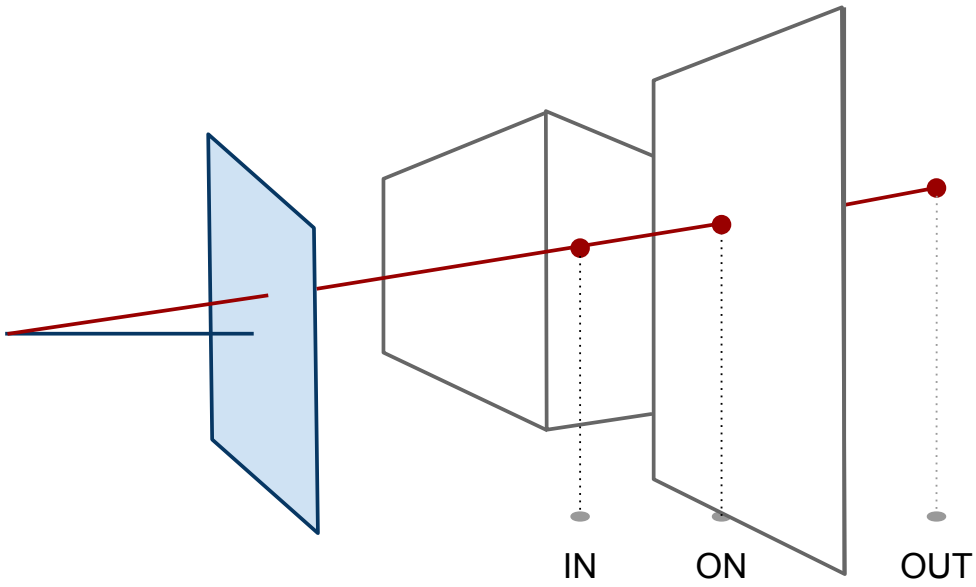


Figure 6.6: Depth measurements d_i might be generated by a surface in our model (represented by $t_i = \mathsf{ON}$) or by an object inside or outside the environment (in which case $t_i = \mathsf{IN}, \mathsf{OUT}$ respectively).

where D_x contains indices for all depth measurements in column x. We verify that (6.39) does in fact correspond to the log–likelihood (6.37) by substituting the above into (6.1), giving

$$\begin{aligned} \Pi(M) &= \sum_{j=0}^{W} \pi_{\mathsf{3D}}(j, s_j) && (6.40) \\ &= \sum_{j=0}^{W} \sum_{i \in D_j} \log P(d_i \mid \boldsymbol{p}_i, \bar{d}(\boldsymbol{p}_i; s_j)\ ) && (6.41) \\ &= \log P(D \mid \boldsymbol{P}, M)\ . && (6.42) \end{aligned}$$

6.8 Joint Model

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

$$P(X_{\mathsf{mono}}, X_{\mathsf{stereo}}, X_{\mathsf{3D}} \mid M) = P(X_{\mathsf{mono}} \mid M) P(X_{\mathsf{stereo}} \mid M) P(X_{\mathsf{3D}} \mid M)\ . \tag{6.43}$$

Taking logarithms leads to summation over payoffs,

$$\log P(X_{\mathsf{mono}}, X_{\mathsf{stereo}}, X_{\mathsf{3D}} \mid M) = \Pi_{\mathsf{joint}}(M) \tag{6.44}$$

where

$$\pi_{\mathsf{joint}}(\boldsymbol{p}) = \pi_{\mathsf{mono}}(\boldsymbol{p}) + \pi_{\mathsf{stereo}}(\boldsymbol{p}) + \pi_{\mathsf{3D}}(\boldsymbol{p})\ . \tag{6.45}$$

Finally, the log posterior is

$$\begin{aligned}\log P(M \mid X_{\text{mono}}, X_{\text{stereo}}, X_{\text{3D}}) &\propto \log P(X_{\text{mono}} \mid M) + \log P(X_{\text{stereo}} \mid M) \\ &\quad + \log P(X_{\text{3D}} \mid M) + \log P(M) \\ &= \Pi_{\text{joint}}(M) - \Gamma(M)\end{aligned} \tag{6.46}$$

We can now cast maximum–likelihood and maximum–aposteriori inference in the form (6.4),

$$\hat{M}_{\texttt{ML}} = \operatorname*{argmax}_{M \in \mathcal{M}} \Pi_{\text{joint}}(M) \tag{6.47}$$

$$\hat{M}_{\texttt{MAP}} = \operatorname*{argmax}_{M \in \mathcal{M}} \Pi_{\text{joint}}(M) - \Gamma(M) \tag{6.48}$$

$$\tag{6.49}$$

6.9 Discussion

Lighting invariants and normalization for stereo
Joint multiple view models - joint gaussian?

6.9.1 Extension: using orientation information for stereo

6.9.2 EM for Occlusion Resolution

6.10 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 monocular reasoning and we have shown that techniques from monocular and stereo vision can be integrated with 3D data in a coherent Bayesian framework.
In future work we intend to use indoor Manhattan models to reason about objects, actions, and scene categories. We also intend to investigate structural SVMs for learning parameters, which may allow us to relax the conditional independence assumptions between sensor modalities.

7

Inference

7.1 Introduction

In this chapter we solve the optimization problem (6.6), which asks for the maximizer M^* of

$$f(M) = \sum_{j=1}^{W} \pi(j, s_j, o_j) - \sum_{i=0}^{K-1} \gamma(i; M) \,, \tag{7.1}$$

where

$$M = (x_1, y_1, o_1,\ x_2, y_2, o_2,\ \ldots,\ x_{n-1}, y_{n-1}, o_{n-1},\ x_n) \tag{7.2}$$

is an indoor Manhattan scene in vertex representation.

We assume that the scene rotation R_w as well as the floor and ceiling position Z have been recovered as discussed in Chapter 5. We further assume that all images are rectified as in (4.17). 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 maximization (6.6). We develop the algorithm conceptually before formalising it.

We wish to constrain hypotheses to those representing valid indoor Manhattan models. In terms of individual pixels this 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 minimisation 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 **??** shows a partial model 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 formalize 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 minimum–cost *partial* model spanning columns $0 \dots x$" for various x. To solve this we enumerate over all possible walls W that terminate at x, then for each we recursively solve sub–problems corresponding to the left extent of the wall. This recursion terminates at the left boundary of the image since each recursion leads to an x' that is strictly less than x. As in all dynamic programming, the solution to each sub–problem is cached to avoid redundant computation, then a simple counting argument shows that the algorithm has polynomial complexity.

The authors have experienced some difficulty giving a clear and concise description of the algorithm in past publications. We proceed therefore with a short intuitive discussion before presenting our algorithm formally.

7.2 Intuition

Consider a path finding problem in which we must are asked to identify the path from left to right through a cost matrix that minimizes the sum of costs along the path. An example cost matrix and the least–cost path is shown in Figure **??**. The solution to this simple problem is a special case of Dijkstra's algorithm and is well known in the literature, but for pedagogical purposes we describe it here.

One view of the solution — slightly non–standard but relevant to the algorithm to come — is as a series of sub–problems as follows. Let $h(\boldsymbol{p})$ be the sub–problem: "what is the minimum energy path from any point on the left edge of the matrix to $\boldsymbol{p}$?" Then we say that any a path $\boldsymbol{Y}$ *satisfies* $h(\boldsymbol{p})$ if the last entry in $\boldsymbol{Y}$ equals $\boldsymbol{p}$, and that $\boldsymbol{Y}$ *solves* $h(\boldsymbol{p})$ if, in addition, $\boldsymbol{Y}$ has minimal cost among all such paths.

These definitions are useful because we can solve each sub–problem in terms of other sub–problems, all the way down to a trivial boundary condition,

$$h(x,y) = \begin{cases} \min\Big(h(x-1,y-1), h(x-1,y), h(x-1,y+1)\Big) + c(x,y), & \text{if } x > 0 \\ 0 & \text{if } x = 0 \end{cases} \tag{7.3}$$

where c is the cost matrix. The algorithm corresponding to (7.3) is simply a recursive function of (x, y) that evaluates (7.3), then the least–cost path is recovered by back–tracking afterwards. The optimization problem that is the subject of this chapter can be thought of as a path–finding problem where the seam representation constitutes a path from left–to–right through the image, and the payoff function constitutes a cost matrix. Unfortunately, however, the penalty term has no counterpart in the above framework, so (6.6) cannot be solved using recurrence relations of the form (7.3). This is because the framework above has no regularization term to penalize complicated paths over those comprising simple straight edges.

7.2.1 Viterbi Decoding

Let us generalize (7.3) by removing the continuity constraint on paths and introducing a penalty for transitions. Let the cost of transitioning from (x, y) to $(x + 1, z)$ be $a(y, z)$. Then the cost of a path $\boldsymbol{Y}$ is

$$\sum_{i=1}^{n} c(i, Y_i) - a(Y_i, Y_{i+1}) \tag{7.4}$$

and the corresponding recurrence relation is

$$h(x, y) = \begin{cases} \min\limits_{1 \le z \le m} h(x-1, z) - a(z, y), & \text{if } x > 0 \\ 0, & \text{if } x = 0\ . \end{cases} \tag{7.5}$$

If the cost and transition matrices correspond to log–likelihoods then this is precisely the Viterbi algorithm for maximum–likelihood inference in hidden Markov models.
Unfortunately, we are still unable to express the penalty term $\Gamma(M)$ in a framework such as (7.5) because Γ introduces a bias for solutions that consists of a few straight edges, but there is no notion of "straightness" in (7.3). Indeed the problem is worsened considerably because we seek a path that corresponds to some 3D reconstruction, but many paths will not correspond to any physically realisable reconstruction.

7.2.2 The Diagonal Route Model

Model where paths can go straight or along either of two diagonals.
The recurrence relation looks like

$$h(x, y, a) = \max\Big[h(x-1, y+a, a), \max_{a'}\big(h(x-1, ya-', a'), T(a', a)\big)\Big] \tag{7.6}$$

We have managed to express a prior on the number of straight sub–segments in a dynamic programming formulation.
There's a lot further to go yet, but the final model will look something like the above.

7.3 Preliminaries

7.3.1 Definitions

Partial Scenes

We begin by generalizing the notion of a scene to a partial scene, which spans the portion of the image from its left edge to some image column x, $0 \le x \le W$. An example of a partial scene in shown in Figure **??**.

Definition 1. *The length of a scene M in vertex representation* (4.59) *is the number of distinct walls in M and is denoted $|M|$.*

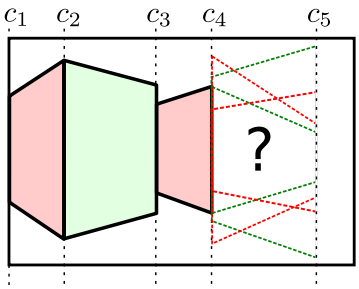


Figure 7.1: A partial model terminating at column c_4 with several feasible (green dashed) and infeasible (red dashed) wall segments.

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

$$S = \{(s_j, o_j)\}_{j=0}^{|M|} \tag{7.7}$$

where the values s_j, o_j are defined as for ordinary seams by equation (4.61).

We now define two operations for manipulating walls. Walls are represented by the 4–tuple (x_a, y_a, o, x_b) as defined by equation (7.22).

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* (4.56).
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.

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

$$M_{-q} = (x_1, y_1, o_1, \ldots,\ x_{|M|-q-1}, y_{|M|-q-1}, o_{|M|-q-1},\ x_{|M|-q}) \,. \tag{7.8}$$

Definition 5. *The concatentation 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*

$$M +\!\!+ W = (x_1, y_1, o_1, \ldots,\ x_{|M|-1}, y_{|M|-1}, o_{|M|-1},\ x_a, y, o,\ x_b) \tag{7.9}$$

Additive Scores

Here we introduce notions of scores for walls and scenes. Our definitions broadly mirror the definition for full scenes in equation (6.1); however, there is in general no direct probabilistic interpretation for the definitions below. This poses no difficulty to the probabilistic consistency of our algorith since in this setting we are simply solving the optimization 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 internals of the probabilistic model in this chapter.

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

$$f(M) = \sum_{j=0}^{x_b} \pi(j, s_j, o_j) - \sum_{i=0}^{|M|} \gamma(i; M) \tag{7.10}$$

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

$$\Pi(W) = \sum_{j=x_a}^{x_b-1} \pi(j, s_j, o_j) \; . \tag{7.11}$$

We note the following additivity property of scores.

Lemma 1. *Let M be a scene and W be a wall and suppose $M +\!\!+ W$ is well–defined. Then the score obtained by $M +\!\!+ W$ is*

$$f(M +\!\!+ W) = f(M) + \Pi(W) - \gamma(M, W) \tag{7.12}$$

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. □

Feasibility

Not all scenes permitted by the vertex representation are physically realisable since some would imply the existence of infinitely thin walls. Figure XXX shows three examples of impossible scenes. Lee *et al.* [DCLK09] 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 [DCLK09], which is functionally dependent on the values*

$$y_b,\ o_i,\ x_{i+1},\ y_{i+1},\ o_{i+1} \tag{7.13}$$

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 [DCLK09] □

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

Proof. Any truncation M_{-q} consists of a subset of the corners in M, all of which are feasible by assumption. □

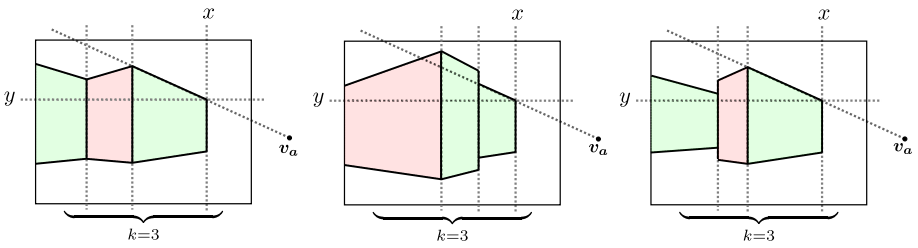


Figure 7.2: Three models satisfying the sub–problem $f_{in}(x, y, a)$.

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

$$y_a,\ o_a,\ x_b,\ y_b,\ o_b, \tag{7.14}$$

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

7.4 Proposed Algorithm

We now present the sub–problems and the recurrence relations comprising the first version of our solution to (6.6).

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 score.

Figure 7.2 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 $1 \leq x \leq W$, $1 \leq y \leq H$, $o \in \{1, 2\}$.*

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 [DCLK09], 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. □

We now turn to the central theorem of this chapter.

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 feasibile 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

$$f(Q') > f(M') \,. \tag{7.15}$$

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

1. Q terminates at (x, y, o) because its right–most wall is W, which terminates at (x, y, o) by assumption.

2. Q is feasible by Lemma 4.

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

$$\begin{aligned} f(M) &= f(M') + \Pi(W) - \gamma(M', W) \\ f(Q) &= f(Q') + \Pi(W) - \gamma(Q'; W) \end{aligned} \tag{7.16}$$

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' terminating at the same state. Therefore

$$\gamma(M', W) = \gamma(Q', W) \,. \tag{7.17}$$

Combining (7.15), (7.16), and (7.17), we have

$$f(Q) > f(M) \,, \tag{7.18}$$

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

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

$$\mathcal{S} = [1, W] \times [1, H] \times \{1, 2\} \,. \tag{7.19}$$

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

$$M = M' +\!\!+ W \tag{7.20}$$

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

Intuitively one can think of the feasible set as the set of states "reachable in one step" from $\boldsymbol{s}$. Our first recurrence relation is a corollary to theorem 1 as follows.

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

$$f_{in}(\boldsymbol{s}) = \max_{\boldsymbol{s}' \in \mathcal{F}(\boldsymbol{s})} \Big(f_{in}(\boldsymbol{s}') + \Pi(W) - \gamma(\boldsymbol{s}', W) \Big) \,, \tag{7.21}$$

where

$$W = (x', y', o, x) \,. \tag{7.22}$$

Further,

$$f_{in}(1, y, o) = 0 \qquad \forall y, o \,. \tag{7.23}$$

Proof. Let M be a solution to $f_{in}(\boldsymbol{s})$. We first show that the left side of (7.21) 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 $f_{in}(\boldsymbol{s}')$ for $\boldsymbol{s}' \in \mathcal{F}(\boldsymbol{s})$. Let W be as defined in (7.22). We have that

1. $M' +\!\!+ W$ is feasible by the definition of the feasible set; and
2. $M' +\!\!+ W$ terminates at $\boldsymbol{s}$ by the definition of W.

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

$$\begin{aligned} f(M) &\geq f(M' +\!\!+ W) && (7.24)\\ &\geq f(M') + \Pi(W) - \gamma(\boldsymbol{s}', W) && (7.25)\\ f_{in}(\boldsymbol{s}) &\geq f_{in}(\boldsymbol{s}') + \Pi(W) - \gamma(\boldsymbol{s}', W) \,, && (7.26) \end{aligned}$$

thus giving the desired inequality.
The equality is obtained by noting that theorem 1 guarantees that there is at least one state $\boldsymbol{s}^* \in \mathcal{F}(\boldsymbol{s})$ such that a solution to $f_{in}(\boldsymbol{s}^*)$ concatenated with W is a solution to $f_{in}(\boldsymbol{s})$.
Finally, the boundary condition (7.23) follows from the fact that a scene that terminates at column 1 does not span any part of the image. □

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

$$f(M^*) = \max_{y,o} f_{in}(W, y, o) \,. \tag{7.27}$$

We compute $f(M^*)$ by recursively evaluating f_{in} according to (7.21) until we reach the boundary condition (7.23). To avoid redundant computation we cache the result of each evaluation together with the state $\boldsymbol{s}'$ corresponding to the maximising term in (7.21). This allows the desired model M^* to be reconstructed by back–tracking once all evaluations are complete. This is formalised in Algorithm **??**.
TODO: expand on how back–tracking is done

Algorithm 4 Solve (6.6) [version 1]

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

```
cache = ∅
ptrs = ∅
f* = −∞
for y = 1 to H do
  for o ∈ {1, 2} do
    f_cur ← Solve(W, y, o)
    if f_cur > f* then
      f* ← f_cur
      s* ← {W, y, o}
    end if
  end for
end for
M* ← (W)
while s* ≠ ∅ do
  M* ← (s*_x, s*_y, s*_o) ++ M*
  s* ← ptrs(M*)
end while
```

Subprocedure Solve:

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

```
if s ∉ cache then
  if x = 0 then
    cache[s] ← 0
    ptrs[s] ← ∅
  else
    for all s' ∈ F(s) do
      W ← (s'_x, s_y, s_o, s_y)
      f_cur ← Solve(s') + Π(W) − γ(s', W)
      if f_cur > cache[s] then
        cache[s] ← f_cur
        ptrs[s] ← s'
      end if
    end for
  end if
end if
return cache[s]
```

Algorithmic Complexity

Due to the caching scheme, (7.21) is evaluated at most once for each unique state. There are $2WH$ possible parameters and the complexity of each evaluation is $O(W^2H)$, since the minimisation in (7.21) 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 $L = \max(W, H)$.

7.4.1 First Refinement: Auxiliary Sub–problems

The basic algorithm described thus far involves minimising over all pixels to the left of $\boldsymbol{s}$ for each state $\boldsymbol{s}$. In the previous section we enforced feasibility by explicitly testing each $\boldsymbol{s}'$ 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 minimization over $O(L^2)$ 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 $\boldsymbol{s} = (x, y, o)$ be a state. Then*

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

We say that a scene solves a sub–problem if no other satisfying scene obtains greater score.

Intuitively, the scenes satisfying f_{up} and f_{down} are those that termate directly above (x, y) and directly below (x, y), respective. 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. Figure TODO shows examples of scene satisfying each of thes above sub–problems.
The purpose of introducing auxiliary sub–problems is to permit a series of more efficient recurrent relations. We begin with f_{up} and f_{down}.

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

$$f_{up}(x, y, o) = \begin{cases} \max\Big(f_{in}(x, y, a), f_{up}(x, y-1, o)\Big), & \text{if } y > 1 \\ f_{in}(x, y, a), & \text{if } y = 1 \end{cases} \tag{7.28}$$

$$f_{down}(x, y, o) = \begin{cases} \max\Big(f_{in}(x, y, o), f_{down}(x, y+1, o)\Big), & \text{if } y < H \\ f_{in}(x, y, o), & \text{if } y = H \end{cases} \tag{7.29}$$

Proof. We prove (7.28) only; the proof of (7.29) is nearly 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. □

Next we prove recurrence relations for f_{out}.

Corollary 3. *Let $\boldsymbol{s} = (x, y, o) \in \mathcal{S}$ be a state. Then TODO: deal with feasibility here TODO: introduce notation for corner categories properly*

$$f_{out}(x, y, o) = \max_{o' \in \{l,r\}} \max\Big(f_{up}(x, y-1, o') - \gamma(x, o', o, -1), \\ f_{in}(x, y, o') - \gamma(x, o', o, 0), \\ f_{down}(x, y+1, o') - \gamma(x, o', o, +1), \Big) \tag{7.30}$$

Proof. TODO □

And finally we re–write (7.21) in terms of the new sub–problems as follows.

Corollary 4. *Let $\boldsymbol{s} = (x, y, o) \in \mathcal{S}$ be a state with $x > 1$. Let $Y_{\boldsymbol{s}}(x')$ be the intersection of column x' with the line through $\boldsymbol{v}_o$ and (x, y) (see figure TODO). Then*

$$f_{in}(x, y, o) = \max_{x' < x} \Big(f_{out}(x', Y_{\boldsymbol{s}}(x'), o) + \Pi(W) \Big) , \tag{7.31}$$

where $\boldsymbol{s}' = (x, Y_{\boldsymbol{s}}(x'), o)$ and $W = (x', Y_{\boldsymbol{s}}(x'), o, x)$ is a wall.

Proof. TODO □

The dependencies between the sub–problems are illustrated as an evaluation graph in Figure **??**.
TODO: mention how this is all implemented.

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 maximization (7.31) is now over $O(L)$ terms (whereas (7.21) consists of $O(L^2)$ terms). The complexity of the algorithm in this section is therefore $O(L^3)$.

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

Evaluating (7.31) remains an $O(W)$ operation due to the minimisation over x'. In this section we to reduce this to $O(1)$.
Consider recurrence relation (7.31). The maximization is over columns $x-1, x-2, \ldots, 1$. Suppose we split this set at some column z, producing two sets, $Z_+ = \{x-1, x-2, \ldots, z\}$ and

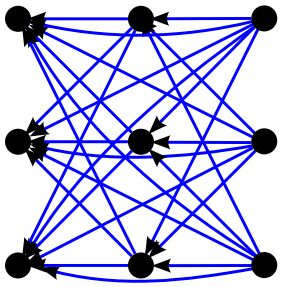


Figure 7.3: The evaluation graph for a 3×3 image under the naive algorithm.

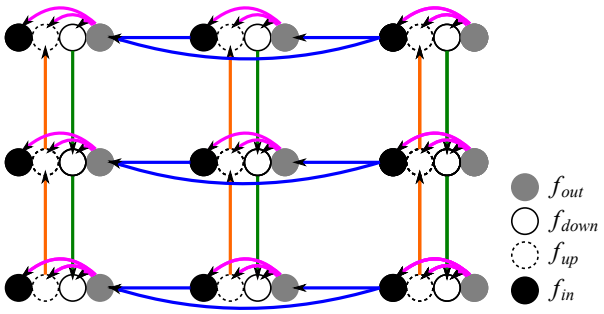


Figure 7.4: The evaluation graph for a 3×3 image introducing auxiliary sub–problems.

$Z_- = \{z-1, z-2, \ldots, 1\}$. Then the maximizing scene at column x could be written

$$f_{in}(x,y,o) = \max\Big\{ \max_{\boldsymbol{s}' \in Z_+} \big(f_{out}(\boldsymbol{s}', o) + \Pi(W) - \gamma(\boldsymbol{s}', W) \big), \\ \max_{\boldsymbol{s}' \in Z_-} \big(f_{out}(\boldsymbol{s}', o) + \Pi(W) - \gamma(\boldsymbol{s}', W) \big) \Big\}. \tag{7.32}$$

Now compare the maximization over Z_- in (7.32) with the sub–problem $f_{in}(z, \tilde{y}, o)$, where $\tilde{y}$ equals $Y_{\boldsymbol{s}}(z)$ rounded to the nearest integer. Both maximizations are over columns $z-1 \ldots 1$. The only difference is that in (7.32) the y–coordinates at each column are $Y_{\boldsymbol{s}}(x)$ whereas in $f_{in}(z, \tilde{y}, o)$ they are $Y_{\boldsymbol{t}}(x)$ where $\boldsymbol{t} = (z, \tilde{y}, o)$. If we accept the latter as an approximation of the former then we could set $z = x - 1$ and replace the $O(W)$ maximization 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 approximation because the quantities $Y_{\boldsymbol{s}}(x)$ and $Y_{\boldsymbol{t}}(x)$ may differ substantially, as illustrated in figure TODO. Furthermore, repeatedly making this approximation in recursive evaluations of f_{in} compounds the errors, as illustrated in figure TODO.

Alternative Description

Consider the sub–problem $f_{in}(x, y, o)$ as formulated in (7.31). Evaluating f_{in} is like walking

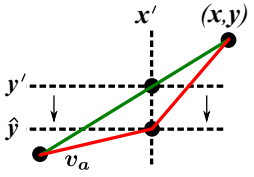


Figure 7.5: The bend introduced by rounding y' to $\hat{y} = \lfloor y' + 0.5 \rfloor$.

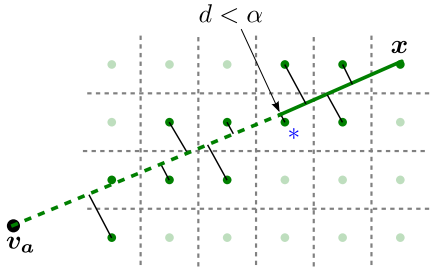


Figure 7.6: A line from $\boldsymbol{x}$ to $\boldsymbol{v_a}$ with the distances d to nearby pixel centres (green dots). The starred pixel is the first that satisfies $d < \epsilon$.

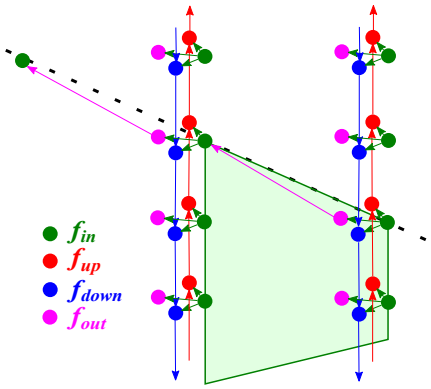


Figure 7.7: A graph in which each node represents a sub–problem and each edge is a dependence relation. Two columns are expanded; other column are omitted for brevity. The green quad is a wall corresponding to a particular pair of nodes in the graph.

along each column $x-1, x-2, ..., 1$ 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 same for f_{in} would produce a bend in the wall as shown in Figure **??**. In (**??**) 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 $\boldsymbol{t} = (z, \tilde{y}, o)$ to replace $\boldsymbol{s} = (x, y, o)$ whenever

$$|y - \tilde{y}| < \epsilon \,. \tag{7.33}$$

To evaluate $f_{in}(x, y, o)$ we find the largest integer $z < x$ satisfying the above. We do this by enumerating in turn 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

$$\begin{aligned} f_{in}(x, y, o) = \max\Bigg\{ & \max_{x' \in [z, x-1]} \Big(f_{out}(x', Y_{\boldsymbol{s}}(x'), o) + \Pi(W) - \gamma(\boldsymbol{s}', W) \Big), \\ & f_{in}(z, Y_{\boldsymbol{s}}(z), o) + \Pi(W) - \gamma(\boldsymbol{s}', W) \Bigg\} \,. \end{aligned} \tag{7.34}$$

where

$$z = \max\Big\{ x' \in \mathbb{Z} : x' < x \wedge \big| Y_{\boldsymbol{s}}(x') - \lfloor Y_{\boldsymbol{s}}(x') + 0.5 \rfloor \big| \leq \epsilon \Big\} \tag{7.35}$$

A bound on number of terms in (7.34) 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 $\boldsymbol{v} = (v_x, v_y) \in \mathbb{R}^2$ there is some $(p, q) \in \mathbb{Z}^2$ with $x - R \leq p < x$ such that*

$$\frac{|(v_y - y)(p - x) - (v_x - x)(q - y)|}{\sqrt{(v_x - x)^2 + (v_y - y)^2}} \leq \epsilon \tag{7.36}$$

Further, the above holds for

$$R = \epsilon\sqrt{2} + \frac{1}{\epsilon} \tag{7.37}$$

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 $b = \frac{v_x - x}{\sqrt{(v_x-x)^2+(v_y-y)^2}}$,

$$|ar - bs| < \epsilon \tag{7.38}$$

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

$$\left| \frac{b}{a} s - r \right| \leq \frac{\epsilon}{a} \leq \epsilon\sqrt{2} \,. \tag{7.39}$$

Dirichlet's theorem (TODO: cite) guarantees that such a pair (r, s) exist with $1 \leq s \leq \frac{1}{\epsilon}$. Rearranging the above we get

$$\begin{aligned} |r| &\leq \epsilon\sqrt{2} + \left| \frac{b}{a} s \right| && (7.40) \\ &\leq \epsilon\sqrt{2} + \frac{1}{\epsilon} && (7.41) \\ & && (7.42) \end{aligned}$$

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

$$-\epsilon\sqrt{2} - \frac{1}{\epsilon} \leq \quad r \quad < 0 \tag{7.43}$$

$$-\epsilon\sqrt{2} - \frac{1}{\epsilon} \leq \quad p - x \quad < 0 \tag{7.44}$$

$$x - R \leq \quad p \quad < x\,. \tag{7.45}$$

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

Algorithmic Complexity

For fixed ϵ, the number of terms in the maximization in (7.34) is bounded by some constant. Hence overall complexity of the algorithm is given by the total number of unique sub–problems, which is $O(L^2)$.

7.4.3 Down–sampling the orientation map

An optional further optimisation is to down–sample the payoff function. In this case the objective function is still equal to the log likelihood, but now only a subset of the original family of models are considered.

7.5 Results

Our data–set consists of 18 manually annotated video sequences of indoor scenes averaging 59 seconds in duration. We sample frames at one second intervals and divide frames into consecutive groups of 3 (one base frame and two auxiliary frames). Our training set consists of 150 such triplets generated from 8 different sequences. Our test set contains 204 triplets from the remaining 10 sequences. No sequence appears in both the training and test sets.

To acquire ground truth data we reconstructed camera trajectories using structure–from–motion software (we use the PTAM system of Klein and Murray [KM07]) 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 point. 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 [DCLK09]. For stereo we use patches of size 5×5.

We compute two error metrics: the labeling accuracy, which is the proportion of all pixels that were labeled with the correct orientation, and the mean relative depth error (**??**). While the latter better captures similarity to the ground truth, not all the systems we compare against have direct 3D interpretations and in such cases we must compare on labeling accuracy.

To the best of our knowledge, there is no previously published work 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.* [BSFC08], 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 randomized 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.* [DCLK09]. 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 7.8. Our system significantly out–performs both others. Even when restricted to monocular features, our system outperforms [BSFC08], which has access to 3D cues. This reflects the utility of global consistency and the indoor Manhattan representation in our approach.

The initialization procedure of [DCLK09] fails for 31% of our training images, so at the bottom of Figure 7.8 we show results for their system after excluding these images. Labeling 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 optimizes for the depth error.

Figure 7.8 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 but can lack precision at corners and boundaries.

Figure **??** 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 optimization.

TODO: compare results with no penalty term, or using a viterbi prior.

7.6 Alternative Approaches

In this section we briefly compare our algorithm to two related approaches.

[1]This row excludes cases for which [DCLK09] was unable to find overlapping lines during initialization.

Algorithm	Mean depth error (%)	Labeling 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 *et al.* [BSFC08]		40.6
Lee *et al.* [DCLK09]	79.8	45.5
excluding failures[1]	34.1	66.2

Figure 7.8: Performance on our data–set. Labeling accuracy is the percentage of correctly labeled pixels over the data–set, and depth error is a per–pixel average of (**??**).

7.6.1 Branch and Bound

Lee *et al.* [DCLK09] 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 (**??**). 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. We give timing comparisons with their approach in Figure **??**.
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 (7.28) and (7.29) 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.

7.6.2 Graph Cuts

Many pixel–labelling problems can been solved using graph cuts. Kolmogorov and Zabih [KZ02a] 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 optimization 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 optimize directly within labelling representation then if some pixel $\boldsymbol{p}$ were assigned label a then $\boldsymbol{q} = H_a\boldsymbol{p}$ 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 — and 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 [KZ02a], which is both approximate and non–deterministic. In contrast, our approach is exact, deterministic, and highly efficient.

7.7 Extensions

In this section we discuss several possible extensions and generalizations to our algorithm.

7.7.1 Non–memoryless Scene Priors

Recall that our prior on scenes (6.7) is

$$P(M \mid \boldsymbol{\lambda}) = \frac{1}{Z}{\lambda_1}^{n_1}{\lambda_2}^{n_2}{\lambda_3}^{n_3} \tag{7.46}$$

This prior is memoryless because it corresponds to the outcome of a series of independent trials. Intuitively, this corresponds to the notion that the marginal probability of an additional corner is independent of the number of corners already added to a model. Formally,

$$P(n_i = k + m \mid n_i \geq k, \boldsymbol{\lambda}) = P(n_i = m \mid \boldsymbol{\lambda}) \; . \tag{7.47}$$

This property is important for the algorithm presented above because the logaritm of a memoryless prior is linear in n_1,n_2,, and n_3, which permitted us to write the prior as a sum over independent penalties for each corner category, as in (6.10). On this basis we defined sub–problems independently of the number of corners in a model and were able to incorporate penalties as additive terms in (7.30). Without a 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 (7.46) 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. If we sampled images of indoor environments and recorded the complexity of each scene manually it seems plausible that scene frequency would increase up to some small number of walls (say, between 2 and 10), and then decrease.
Of course, we may endlessly re-examine our prior information, and the question phrased above could form the topic of a significant empirical study of real–world floorplans and their images. Here we show how to incorporate any prior that can be written as a function of n_1, n_2, n_3, at the cost of introducing extra dimensions to the state space and a corresponding increase in the computational complexity of the algorithm.

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

$$\mathcal{S} = \{(x, y, o, \boldsymbol{n})\} \tag{7.48}$$

$$\mathcal{S} = [1, W] \times [1, H] \times \{1, 2\} \times \mathbb{Z}^3 \tag{7.49}$$

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 lemma **??**, ω is functionally dependent only on the values

$$x_b, o_b, \; o_0, \; \text{Sign}(y_0 - y_b) \tag{7.50}$$

so we write

$$\omega(x, o_1, o_2, s) = \begin{cases} [1\ 0\ 0], & \text{for concave corners} \\ [0\ 1\ 0], & \text{for convex corners} \\ [0\ 0\ 1], & \text{for occluding corners} \end{cases} . \tag{7.51}$$

The only recurrence relation we need to update is (7.30). The relation for f_{out} becomes

$$\begin{aligned} f_{out}(x, y, o, \boldsymbol{n}) = \max_{o' \in \{l, r\}} \max\Big(& f_{up}(x, y-1, o', \boldsymbol{n} - \omega(x, o', o, -1)\ ), \\ & f_{in}(x, y, o', \boldsymbol{n} - \omega(x, o', o, 0)\ ), \\ & f_{down}(x, y+1, o', \boldsymbol{n} - \omega(x, o', o, +1)\ )\ \Big) \end{aligned} \tag{7.52}$$

Comparison with (7.30) shows that we have replaced the penalty term with a transition from $\boldsymbol{n}$ to $Counts' = \boldsymbol{n} - \omega(\cdot)$. This means our sub–problems no longer incorporate penalties at all
TODO: finish

7.7.2 Upper–bounds

Can upper-bound the payoff for any model with ¿= k walls by the maximum payoff for each column minus k*(the minimum penalty for any wall category). By enumerating k=1... this upper bound will eventually be less than the best model found so far, so can stop.

7.7.3 Computing expectations

Can compute expectations and max–expectations over functions that decompose column–wise.

7.7.4 Computing more general expectations

Sampling from $P(M \mid X)$ using perturb–and–map.

7.7.5 No constraints

Experiment: remove physical realisability constraint, evaluate on complete dataset.
Experiment: remove all manhattan constraints, just do Viterbi decoding, evaluate on dataset.

8

Learning

8.1 Introduction

This chapter shows how to learn to reconstruct indoor Manhattan models from training examples. We are given a series of input images together with ground truth reconstructions provided by hand, and the goal is to discover and extrapolate this data in order to recover the correct reconstruction for future images.

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 [HEH05, SSN09, DCLK09, **?**]. 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 preceeding 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 structural support vector machine [**?**]. The use of these tools is part part of a long trend within computer vision towards statistically rigorous, well–understood convex optimization techniques. Recent successfully applications of the structured SVM include detection [**?**], segmentation [**?**], and scene classification [**?**]. In the domain of reconstruction, structured prediction ideas have been applied to several simple models classes such as stereo disparities [**?**] and cuboids [HHF09].

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 image coordinates. These constraints are context–dependent, 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 optimization; (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.

8.2 Background

The approach to learning adopted in this chapter has roots in the structural risk minimization framework developed by Vapnik and Chervonenkis [**?**]. 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 minimizing the expected loss

$$\mathbb{E}\big[\Delta(f(\boldsymbol{x}), \boldsymbol{y})\big] = \int \Delta\big(f(\boldsymbol{x}), \boldsymbol{y}\big)\, \mathrm{d}P(\boldsymbol{x}, \boldsymbol{y}) \tag{8.1}$$

with respect to the predictor f, where Δ is a loss function. Given training samples of the form $(\boldsymbol{x}_i, \boldsymbol{y}_i) \in \mathcal{X} \times \mathcal{Y}$ the integral above is approximated by a summation

$$\mathcal{R}(f) + \sum_i \Delta\big(f(\boldsymbol{x}_i), \boldsymbol{y}_i\big) \tag{8.2}$$

where $\mathcal{R}(f)$ is a regularizer on f. Structural risk minimization is very general and 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 minimization for particular predictor classes and loss functions. For example, the classic support vector machine (SVM) [**?**] is defined for Hilbert input spaces, binary outputs, the class of linear predictors, and the Hinge loss function. In this case equation (8.2) becomes

$$\|\boldsymbol{w}\|^2 + \sum h(\boldsymbol{y}_i, \boldsymbol{w} \cdot \boldsymbol{x}_i)) \tag{8.3}$$

where h is the hinge loss, and learning consists of minimization with respect to $\boldsymbol{w}$. Rewriting the hinge loss terms as constraints leads to the well–known quadratic programming formulation of the SVM [**?**].

Extending binary classification to more general output spaces is a major research programme within the machine learning community; an excellent introduction to recent advances is given

by Bakir *et al.* [**?**]. One recently successful approach is the structured support vector machine proposed by Tsochantaridis *et al.* [**?**], which reduces learning in non–binary ("structured") output spaces to a ranking problem. The structured SVM makes use of a joint feature space $\Psi(\boldsymbol{x}, \boldsymbol{y})$ mapping input/output pairs to real–valued vectors. The structured SVM considers predictors of the form

$$f(\boldsymbol{x}; \boldsymbol{w}) = \underset{\boldsymbol{y}}{\operatorname{argmax}} \big\langle \boldsymbol{w}, \Psi(\boldsymbol{x}, \boldsymbol{y}) \big\rangle \tag{8.4}$$

where $\boldsymbol{w}$ 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,

$$\|\boldsymbol{w}\|^2 + \sum h(\boldsymbol{y}_i, f(\boldsymbol{x}_i)) \; . \tag{8.5}$$

In order to optimize with respect to $\boldsymbol{w}$, Toschantaridis *et al.* showed how to write (8.5) as a constrained optimization problem parallelling the classic SVM:

$$\begin{aligned} &\min_{\boldsymbol{w}, \boldsymbol{\xi}} \; \frac{1}{2}\|\boldsymbol{w}\|^2 + C \sum_{i=1}^{n} \xi_i \\ &s.t. \; \forall k, \boldsymbol{y} \neq \boldsymbol{y}_i : \; \big\langle \boldsymbol{w}, \Psi(\boldsymbol{x}_i, \boldsymbol{y}_i) \big\rangle - \big\langle \boldsymbol{w}, \Psi(\boldsymbol{x}_i, \boldsymbol{y}) \big\rangle \geq \Delta(\boldsymbol{y}, \boldsymbol{y}_i) - \xi_i \; . \end{aligned} \tag{8.6}$$

Unfortunately the number of constraints in (8.6) is in general very large since there is a constraint for every element of the output space. Tsochantaridis *et al.* [**?**] 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.

8.3 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).

8.3.1 Hypothesis Class

This chapter is concerned with the output space consisting of indoor Manhattan reconstructions as described in Chapter 4. For consistency with preceeding 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 $\{(\Phi_i, S_i)\}$.

8.3.2 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 7, we would like each predictor to implement MAP inference for some value of the hyper–parameters, *i.e.* we are interested in predictors of the form

$$f(\Phi) = \operatorname*{argmax}_S P(S \mid \Phi, \boldsymbol{w}) \tag{8.7}$$

for various $\boldsymbol{w} \in \mathbb{R}^n$. In other words, we would like to optimise over the set

$$\left\{ f \;\middle|\; f : \Phi \longmapsto \operatorname*{argmax}_S P(S \mid \Phi, \boldsymbol{w}), \boldsymbol{w} \in \mathbb{R}^n \right\} \tag{8.8}$$

consisting of MAP inference predictors for all possible hyper–parameter values, which we have grouped into the parameter $\boldsymbol{w}$. In order to employ the structured SVM we need to write f in the form (8.4), or equivalently, we need to write the log–posterior $\log P(S \mid \Phi, \boldsymbol{w})$ in the form

$$\log P(S \mid \Phi, \boldsymbol{w}) = \left\langle \boldsymbol{w}, \Psi(\Phi, S) \right\rangle . \tag{8.9}$$

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 log–posterior (6.43), our goal being to show that each is linear in the relevant hyper–parameters.

Prior

Recall the log–prior (6.9), which is repeated below.

$$\log P(S \mid \boldsymbol{\lambda}) = -\log Z + n_1 \log \lambda_1 + n_2 \log \lambda_2 + n_3 \log \lambda_3 . \tag{8.10}$$

Defining $\boldsymbol{n} = \begin{bmatrix} n_1 & n_2 & n_3 \end{bmatrix}$ we have

$$\log P(S \mid \boldsymbol{\lambda}) = \left\langle \boldsymbol{\lambda}, \boldsymbol{n} \right\rangle \; + O(1) . \tag{8.11}$$

Photometric Likelihood

Combining (6.16) and (6.12), the log likelihood for photometric features that was described in Chapter 6 is

$$\log P(\Phi \mid S) = \sum_i \boldsymbol{\omega}_a \cdot \boldsymbol{\phi} \; + O(1) . \tag{8.12}$$

Next we define

$$\boldsymbol{\omega} = \begin{bmatrix} \boldsymbol{\omega}_1 \\ \boldsymbol{\omega}_2 \\ \boldsymbol{\omega}_3 \end{bmatrix} \qquad \Psi_{\mathsf{mono}}(\Phi, S) = \begin{bmatrix} \sum_{i:a_i^*=1} \boldsymbol{\phi}_i \\ \sum_{i:a_i^*=2} \boldsymbol{\phi}_i \\ \sum_{i:a_i^*=3} \boldsymbol{\phi}_i \end{bmatrix} . \tag{8.13}$$

Finally we can write

$$\log P(\Phi \mid S) = \left\langle \boldsymbol{\omega}, \Psi_{\mathsf{mono}}(\Phi, S) \right\rangle \; + O(1) . \tag{8.14}$$

Photoconsistency Likelihood

The log likelihood for photoconsistency features given by equation (6.25) is

$$\log P(\Phi, \Theta_1, \ldots, \Theta_K \mid S) = \sum_i \sum_{k=0}^{K} \|\boldsymbol{\phi}_i - \bar{\boldsymbol{\theta}}_k(\boldsymbol{p}_i; S)\|_\Sigma \,. \tag{8.15}$$

Here we assume that $\boldsymbol{\sigma}$ has the diagonal form

$$\Sigma^{-1} = \mathrm{diag}(\sigma_1, \ldots, \sigma_2) \,. \tag{8.16}$$

It follows that by defining

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_1 \\ \vdots \\ \sigma_m \end{bmatrix} \qquad \Psi_{\mathsf{stereo}}(\Phi, S) = \begin{bmatrix} \sum_i \sum_{k=1}^{K} \big((\boldsymbol{\phi}_i)_1 - (\bar{\boldsymbol{\theta}}_k(\boldsymbol{p}_i; S))_1\big)^2 \\ \vdots \\ \sum_i \sum_{k=1}^{K} \big((\boldsymbol{\phi}_i)_m - (\bar{\boldsymbol{\theta}}_k(\boldsymbol{p}_i; S))_m\big)^2 \end{bmatrix} \,. \tag{8.17}$$

we can write the log likelihood (8.15) in the form

$$\log P(\Phi, \Theta_1, \ldots, \Theta_K) = \big\langle \boldsymbol{\sigma}, \Psi_{\mathsf{stereo}}(\Phi, S) \big\rangle \;\; + O(1) \,. \tag{8.18}$$

8.3.3 Point Cloud Likelihood

The log–likelihood for point cloud features (6.36) is non–linear so for the purpose of learning we approximate it by a first order Taylor expansion about $\boldsymbol{\tau_0}$ as follows.

$$\log P(d \mid S, \boldsymbol{\tau}) \;\; \approx \;\; \frac{\sum_t P(d \mid S, t) P(t \mid \boldsymbol{\tau})}{\sum_t P(d \mid S, t) P(t \mid \boldsymbol{\tau_0})} + O(1) \tag{8.19}$$

$$\log P(D \mid S, \boldsymbol{\tau}) \;\; \approx \;\; \sum_i \sum_t \frac{P(d_i \mid S, t) P(t \mid \boldsymbol{\tau})}{\sum_t P(d_i \mid S, t_i) P(t \mid \boldsymbol{\tau_0})} + O(1) \,. \tag{8.20}$$

It follows that by defining

$$\Psi_{\mathsf{3D}}(D, S) = \frac{1}{\sum_t P(d_i \mid S, t) P(t \mid \boldsymbol{\tau_0})} \begin{bmatrix} \sum_i P(d_i \mid S, t = \mathsf{IN}) P(t = \mathsf{IN} \mid \boldsymbol{\tau}) \\ \sum_i P(d_i \mid S, t = \mathsf{OUT}) P(t = \mathsf{OUT} \mid \boldsymbol{\tau}) \\ \sum_i P(d_i \mid S, t = \mathsf{ON}) P(t = \mathsf{ON} \mid \boldsymbol{\tau}) \end{bmatrix} \,. \tag{8.21}$$

we can write

$$\log P(D \mid S, \boldsymbol{\tau}) = \big\langle \boldsymbol{\tau}, \Psi_{\mathsf{3D}}(D, S) \big\rangle \;\; + O(1) \,. \tag{8.22}$$

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 (**??**), (**??**), (**??**), and (**??**) we have the following linear form for the posterior,

$$P(S \mid X, \boldsymbol{w}) = \big\langle \boldsymbol{w}, \Psi(X, S) \big\rangle \tag{8.23}$$

Feature	Dimensionality	Multi–view?	Single–view?	Reference
Stereo photo–consistency	4	yes	no	Flint *et al.* [**?**]
Point cloud	2	yes	no	Flint *et al.* [**?**]
Line sweeps	1	yes	yes	Lee *et al.* [DCLK09]
RGB+HSV	6	no	yes	
Gabor responses[1]	12	no	yes	

Figure 8.1: The composition of our single– and multiple–view feature space. We omit color and Gabor features from the multiple–view feature space for training efficiency.

where X denotes all observations from all sensor modalitities and we have defined

$$\boldsymbol{w} = \begin{bmatrix} \boldsymbol{\lambda} \\ \boldsymbol{\omega} \\ \boldsymbol{\sigma} \\ \boldsymbol{\tau} \end{bmatrix} \qquad \Psi(X, S) = \begin{bmatrix} \boldsymbol{n} \\ \Psi_{\text{mono}} \\ \Psi_{\text{stereo}} \\ \Psi_{\text{3D}} \end{bmatrix} . \tag{8.24}$$

Discussion

We have shown in the derivations above that the posterior from chapter Section **??** is log–linear (or can be approximated as such) in the hyper–parameters, so substituting (**??**) into (8.7) we are now interested in predictors of the form

$$f(\Phi) = \underset{S}{\operatorname{argmax}} \big\langle \boldsymbol{w}, \Psi(X, S) \big\rangle \tag{8.25}$$

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 aribtrary 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 (8.25), 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 $\boldsymbol{w}$ with respect to an expected loss over a training set $\{(\Phi_i, S_i)\}$.

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 summarized in Figure 8.1.

[1] 4 orientations, 3 scales

8.3.4 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 optimization, and choosing a loss that measures the quantity that one "really" cares about. For example, Hoiem *et al.* [HEH05] learn a per–segment orientation classifier, then pass this as input to a separate 3D reconstruction system [**?**]. However, what one "really" cares about is some loss defined on the output of the entire system rather than the output of individual components, since some segment–level mistakes are insignificant to the overall reconstruction quality, while others are catastrophic. This is not a criticism of the authors' choice, but an illustration of the trade–off faced when choosing a loss function. In this paper 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 [HZ04], and measures the average deviation between reconstructed and ground truth depths. In our notation,

$$\Delta_{\mathsf{depth}}(S, \hat{S}) = \frac{1}{N} \sum_{\boldsymbol{p}} \frac{|d(\boldsymbol{p}; \hat{S}) - d(\boldsymbol{p}; S)|}{d(\boldsymbol{p}; S)} \,, \tag{8.26}$$

where N is the number of pixels. Another reasonable choice is the labelling error, used widely within the semantic segmentation literature,

$$\Delta_{\mathsf{labelling}}(\hat{S}, S) = \frac{1}{N} \sum_{\boldsymbol{p}} \big[\, a(\boldsymbol{p}; \hat{S}) \neq a(\boldsymbol{p}; S) \,\big] \,, \tag{8.27}$$

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 optimized exactly*. The algorithmic details are left to Section 8.4; the key result we establish here is that Δ_{depth} and $\Delta_{\mathsf{labelling}}$ can be written in a form resembling the payoff formulation (**??**) for the feature likelihoods.

First we invoke the independence established in (**??**):

$$\Delta_{\mathsf{depth}}(\hat{S}, S) = \frac{1}{N} \sum_{x=1}^{W} \sum_{y=1}^{H} \frac{|\tilde{d}(x, y; \hat{s}_x) - d(x, y; S)|}{d(x, y; S)} \,. \tag{8.28}$$

Defining a real matrix δ_S,

$$\delta_S(x, j) = \sum_{y=1}^{H} \frac{|\tilde{d}(x, y; j) - d(x, y; S)|}{d(x, y; S)} \,, \tag{8.29}$$

we see that we can write Δ_{depth} in the form

$$\Delta_{\mathsf{depth}}(\hat{S}, S) = \frac{1}{N} \sum_{x=1}^{W} \delta_S(x, \hat{s}_x) \,. \tag{8.30}$$

There is a similar form for $\Delta_{\mathsf{labelling}}$, which we omit here due to space constraints.

Choosing a Loss Function

Neither of the above losses is unequivocally the "correct" loss; the choice will depend on the application. One might expect a strong correlation between the losses, and indeed one can show analytically that

$$\Delta_{\mathsf{depth}}(\hat{S}, S) = 0 \iff \Delta_{\mathsf{labelling}}(\hat{S}, S) = 0 \,. \tag{8.31}$$

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 8.4 shows a significant number of outliers that score very well on Δ_{depth} but poorly on $\Delta_{\mathsf{labelling}}$, and vice versa.

8.4 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 minimizer

$$f^* = \mathrm{argmin}_f \, \mathbb{E}\Big[\Delta\big(f(\Phi), S\big)\Big] \,, \tag{8.32}$$

which we approximate in the framework of structural risk minimization as

$$f^* = \mathrm{argmin}_f \, \mathcal{R}(f) \sum_k \Delta\big(f(\Phi_k), S_k\big) \,, \tag{8.33}$$

where k indexes a training set and $\mathcal{R}$ is a regularizer. To perform this optimization we turn to the tools of structured prediction [**?**], and in particular the structured SVM [**?**]. Following the standard approach [**?**] we cast the learning problem as a constrained optimisation problem,

$$\begin{aligned} &\min_{\boldsymbol{w},\boldsymbol{\xi}} \;\; \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{k=1}^{n} \xi_k \\ &s.t. \;\; \forall k, S \neq S_k : \; \big\langle \boldsymbol{w}, \Psi(\Phi_k, S_k)\big\rangle - \big\langle \boldsymbol{w}, \Psi(\Phi_k, S)\big\rangle \geq \Delta(S, S_k) - \xi_k \,. \end{aligned} \tag{8.34}$$

where C is a user–specified parameter that trades off between prediction accuracy and model complexity. Tsochantaridis *et al.* [**?**] 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 maximization described in (8.25).

2. *Separation.* The algorithm described in [**?**] requires a user–supplied procedure to find the "most–violated constraint" at each iteration. That is,

$$\mathrm{argmax}_S \big\langle \boldsymbol{w}, \Psi(\Phi_k, S)\big\rangle + \Delta(S, S_k) \,. \tag{8.35}$$

Our solutions to both of the above build on the dynamic programming algorithm presented in Chapter 7, which solves problems of the form

$$\mathrm{argmax}_S \sum_x \pi(x, s_x) - \sum_j \gamma(j; S) \,. \tag{8.36}$$

8.4.1 Inference (Prediction)

We showed in Section 8.3 that (8.36) can be written in the form (8.25), so the prediction problem is a straightforward application of the inference algorithm of Chapter 7. This is as expected since, as we have already remarked, (8.25) is equivalent to MAP inference on indoor Manhattan reconstructions, which was precisely the subject of Chapter 7.

8.4.2 Loss–Augmented Inference (Separation)

The separation problem, it turns out, can also be solved using the dynamic programming algorithm from Chapter 7, as the following proposition shows.

Proposition 1. *Let (Φ_k, S_k) be a training instance with payoff matrices $\{\pi_i\}$ as defined in* (**??**)*. Let*

$$\pi_{\mathsf{aug}} = \delta_{S_k} + \sum_i \pi_i \; . \tag{8.37}$$

Then the solution to (8.36) *with $\pi = \pi_{\mathsf{aug}}$ is identical to the solution to* (8.35)*.*

Proof. Direct equivalence of the expressions to be maximized. First substitute (**??**) and (8.30) into (8.36):

$$\log P(S \mid \Phi) + \sum_{x=1}^{W} \delta_{S_k}(x, s_x) \tag{8.38}$$

Further substituting (**??**) and defining γ as in [**?**] gives

$$\sum_i \sum_{x=1}^{W} \pi_i(x, s_x) - \sum_j \gamma(j; S) + \sum_{x=1}^{W} \delta_{S_k}(x, s_x) \; . \tag{8.39}$$

Finally we see that substituting (8.37) gives

$$\sum_{x=1}^{W} \pi_{\mathsf{aug}}(x, s_x) - \sum_j \gamma(j; S) \tag{8.40}$$

□

8.5 Multiple View Results

We evaluated our approach on the data–set proposed in [**?**], 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 [**?**]. Our metrics differ from theirs in two ways. Firstly, they compute 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

	Depth Error (%)		Labelling Error (%)	
Sequence	This Paper[2]	Flint *et al.*	This Paper[3]	Flint *et al.*
`ground`	4.9	66.6	2.9	10.4
`foyer1`	6.1	6.6	3.1	3.1
`foyer2`	4.3	5.4	3.7	4.0
`corridor`	14.6	52.9	9.5	19.2
`mcr`	34.0	67.6	15.	16.2
`kitchen`	16.8	23.6	5.2	6.1
Average	**13.4**	37.1	**6.7**	9.8

Figure 8.2: Multiple–view reconstruction performance on held–out data, comparing with Flint *et al.* [**?**]. For unavoidable reasons we use slightly different metrics so our figures differ from those published in [**?**]. See main text for explanation.

	Depth Error (%)		Labelling Error (%)	
Sequence	This Paper[2]	Flint *et al.*	This Paper[3]	Flint *et al.*
`ground`	17.3	24.5	7.8	12.2
`foyer1`	25.1	31.0	15.1	22.2
`foyer2`	29.1	30.1	15.9	18.6
`corridor`	31.7	33.6	19.3	24.8
`mcr`	70.1	45.9	26.7	20.8
`kitchen`	25.1	26.2	7.7	11.9
Average	33.1	**31.9**	**15.4**	18.4

Figure 8.3: Single–view reconstruction performance on held–out data, comparing with Flint *et al.* [**?**].

denominator. These metrics are separated by at most a monotonic transform but the latter is more convenient to represent in our framework. Secondly, when we compute labelling error we differentiate vertical and horizontal surfaces only, whereas they 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.

The performance of these two algorithms are summarized in Figure 8.2. Our approach significantly out–performs the bootstrapping algorithm. 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 are shown in Figure 8.6; many more are included in additional material.

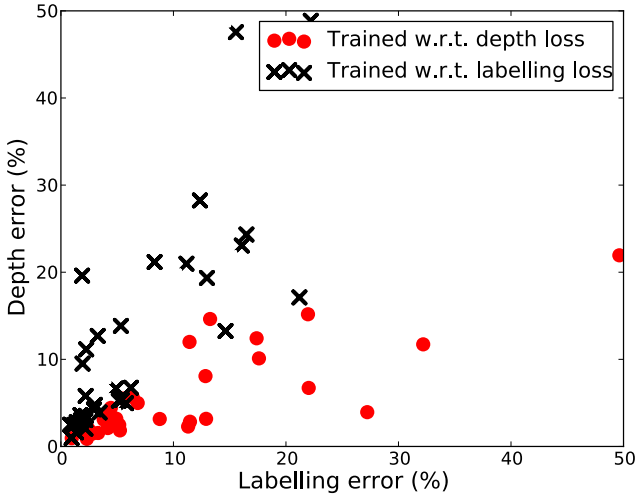


Figure 8.4: The effect of the loss function on training. We train two predictors, one with respect to Δ_{depth} and one with respect to $\Delta_{\text{labelling}}$, then evaluate both on all held–out instances. Each data point shows the error obtained by one predictor on held–out instance. The differing distribution of errors shows that the two predictors trade off errors as expected.

8.6 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 summarized in Figure 8.1. We compared our approach to the single–view approach of Flint *et al.* [**?**], which uses the same dynamic programming algorithm that we rely upon, but uses hand–tuned features.

Performance for each algorithm is summarized in Figure 8.2. When measured by labelling error, our approach out–performs the hand–tuned weights, 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

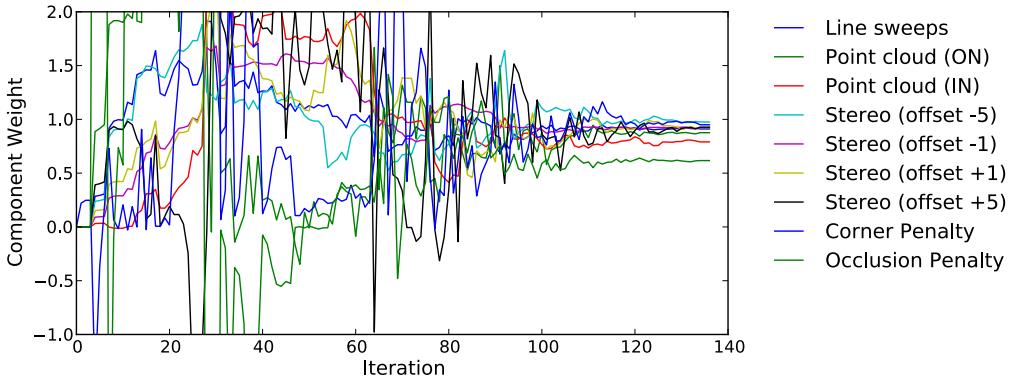


Figure 8.5: Evolution of $\boldsymbol{w}$ during training. Each series shows the value of one component of $\boldsymbol{w}$. After an exploration phase the model converges.

[**?**]. 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 further salient information and reduce the error rate.

8.7 Discussion

The hypothesis class considered in this paper 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 learnt that may be of value to other practitioners.

Condition in the joint feature space, not the input feature space.

gi 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 that should be conditioned:

$$\Psi' = \frac{\Psi - \mu}{\sigma^2} \,. \tag{8.41}$$

Ideally one would sample from the joint feature space to determine the conditioning transformation, but the distribution of inputs and outputs is generally unknown in an empirical risk minimization setting. Instead, we use the training set as a proxy. We compute the empirical mean and variance of $\{\Psi(\Phi_k, S_k)\}_{k=1}^n$ at the outset, then apply the transformation (8.41) after each feature computation.

Condition the loss terms.

For any $\eta > 0$, the minimization problem (8.34) is equivalent (under the substitution $\boldsymbol{w}' = \eta\boldsymbol{w}$, $\boldsymbol{\xi}' = \eta\boldsymbol{\xi}$) to:

$$\begin{aligned} &\min_{\boldsymbol{w}',\boldsymbol{\xi}'} \ \frac{1}{2}\|\boldsymbol{w}'\|^2 + \eta C \sum_{k=1}^{n} \xi'_k \\ &s.t. \ \forall k, S \neq S_k : \ \left\langle \boldsymbol{w}', \Psi(\Phi_k, S_k) \right\rangle - \left\langle \boldsymbol{w}', \Psi(\Phi_k, S) \right\rangle \geq \eta\Delta(S, S_k) - \xi'_k \,. \end{aligned} \tag{8.42}$$

Although any $\eta > 0$ preserves the correctness of the optimization algorithm, we found that choosing $\eta = \mathrm{Var}(\Delta)$ improved numerical stability, since this means the loss terms will have roughly unit variance. Unfortunately, we cannot use the training set to estimate $\mathrm{Var}(\Delta)$ since the loss for the ground truth reconstruction 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.

[2]This column represents the predictor trained with respect to Δ_{depth}.
[3]This column represents the predictor trained with respect to $\Delta_{\mathsf{labelling}}$.

Check that the hypothesis class contains the ground truth.

The algorithm described in [**?**] implicitly assumes that the hypothesis class $\mathcal{Y}$ contains the ground truth labels S_k. This means that if S^+ is the maximizer of (8.35) then

$$\left\langle \boldsymbol{w}, \Psi(\Phi_k, S_k) \right\rangle - \left\langle \boldsymbol{w}, \Psi(\Phi_k, S^+) \right\rangle - \Delta(S^+, S_k) \leq 0 \,, \tag{8.43}$$

since otherwise we would have

$$\left\langle \boldsymbol{w}, \Psi(\Phi_k, S_k) \right\rangle + \Delta(S_k, S_k) > \left\langle \boldsymbol{w}, \Psi(\Phi_k, S^+) \right\rangle + \Delta(S^+, S_k) \,, \tag{8.44}$$

contradicting S^+ as the maximizer of (8.35). 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 (8.43). Our workaround here is simply to check the condition (8.43) each time we solve the separation problem and, if violated, substitute S_k for S^+. This is justified by the observation that if (8.43) 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 $\mathcal{Y}$. This is not an ideal solution but we found it to work well in practice. Unfortunately this patch has the side–effect of hiding bugs in the inference algorithm, so care is warranted.

8.8 Conclusion

We have presented a unified learning framework for reconstructing polygonal models from single and multiple views of a scene. We have chosen to work with the indoor Manhattan class of models in order to leverage the parametrisation and inference algorithm recently proposed for 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 state–of–the–art for multiple–view reconstruction (by a large margin) and on one metric for single–view reconstruction.

In future work will extend this approach to learn geometry together with scene classifiers and context–aware object detectors, optimising with respect to a single joint loss function.

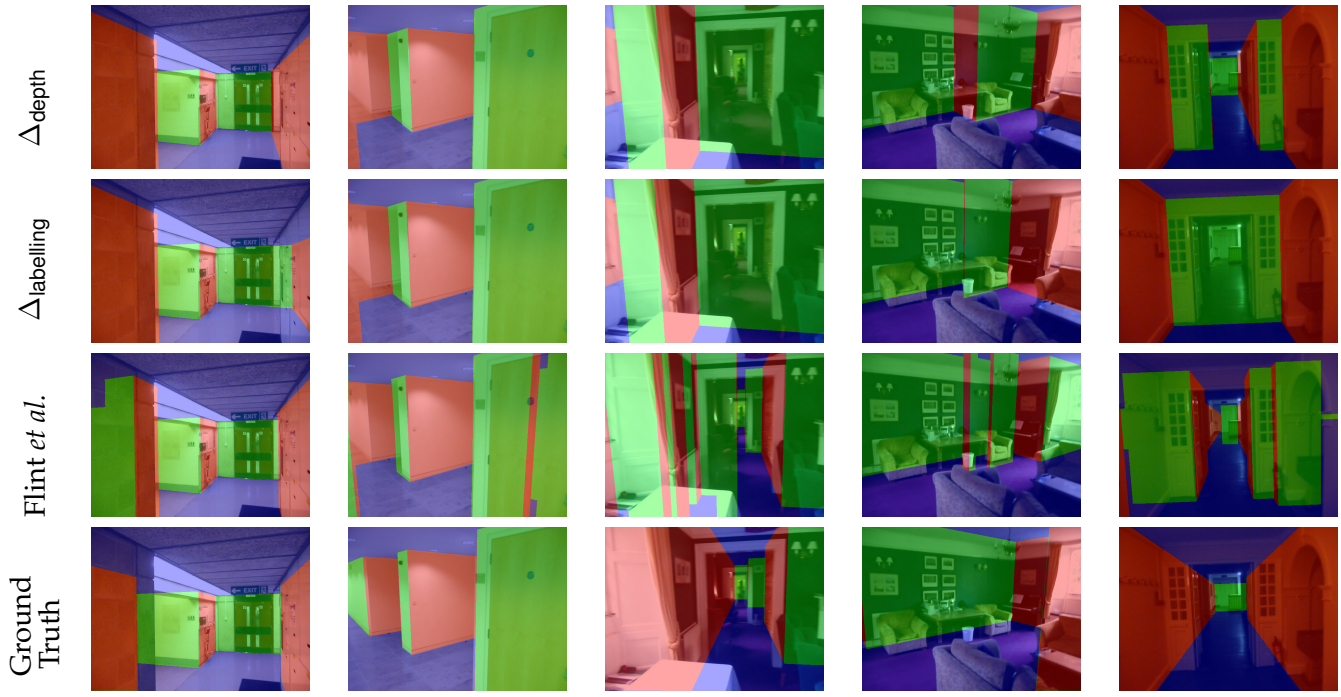


Figure 8.6: Multiple–view reconstructions predicted by our system (held–out samples). The first two rows represent the predictors trained on Δ_{depth} and $\Delta_{\text{labelling}}$ respectively, the third row is from **[?]**, and the fourth row is ground truth.

9

Conclusion

- summarise chapters? - re-iterate take-home messages from introduction - AR is more than just SLAM - multi-view scene understanding is worthwhile - geometry is helpful for scene understanding - indoor Manhattan models are a good choice because - simple - concise - general - efficient inference possible
- key results - textons are a simple and flexible basis for scene context - vanishing points should be estimated from multiple views if available - there is a simple and efficient inference algorithm for indoor Manhattan models - single and multiple views under one framework - discriminative training of indoor Manhattan models is possible
In this thesis I have presented a series of ideas for high–level inference in the context of a moving observed. 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.

9.1 Key Results

Chapter 3 presented
In Chapter 6 we presented a probabilistic model relating indoor Manhattan models to three sensor modalities. We showed that for all three sensor modalities, MAP inference can be reduced to optimisation over a payoff matrix. We see this formulation as the key result of this chapter; it suggests a general strategy by which further sensor modalities can be integrated with our model without modifying the inference algorithm.
One contribution of this thesis is the dynamic programming algorithm that we presented in Chapter 7. The geometric prior that we employed prevents our problem from fitting into common frameworks suited to the Viterbi algorithm. Nevertheless, we showed that a decomposition into sub–problems permitted an efficient and exact dynamic programming solution
A theme of this thesis has been the use of geometry for scene understanding purposes. We have

shown how to extract semantically meaningful reconstructions using ideas drawn from both the geometry literature and the scene understanding literature. It was ideas from geometry that allowed us to decompose our reconstructions into a representation amenable to dynamic programming, but ideas from single–view scene understanding and probability theory that allowed us to connect our hypothesis class to observed image features.

9.2 Future Work

9.2.1 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 look different from that of a camera located close to the floor, even within one scene category. Photoemtric 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 geometry and camera position to compute viewpoint–invariant features. 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 [Tor03].

The second application to which I intended to apply indoor Manhattan models was 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 than just augmenting a feature vector. 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 "traversible" 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 traversibility of each. The occupancy grid estimation might benefit substantially from knowing the shape of the environment about each cell.

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. There are also interesting connections here to the active search problem we considered in relation to our texton work in Chapter 3.

- application areas - use indoor Manhattan models for the purpose they were originally intended for - scene categorization - contextual priming for object detection - other uses - navigation - active search, info-based path planning

9.2.2 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 7, 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 ccould 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 re-

spect 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 penalization of highly complex models. A different approach would be to permit paramtric 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 maximization in (7.32).
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 modifcation to the algorithm presented in Chapter 7.

9.2.3 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 recoery could be extended. In Chapter 8 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 plausability under the model presented in Chapter 6. In this case we would see the reconstruction as a latent variable, so the Bayesian gold standard is always to marginalise. In many cases such a marginalisation is intractible, but in the case of indoor Manhattan models there is a promising approach by which the decomposition of Chapter 7 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 compu-

tational effort invested at previous time steps. Recursive estimators maintain a distribution over an underlying hypothesis class that can be efficiently updated from new evidence. The Kalman filter, for example, uses a vector space as its hypothesis class and maintains a joint Gaussian distribution over hypotheses, represented by a mean and covariance. 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 7. 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 artifacts that integrating information over long periods would introduce.
- Bayesian training - learning from architectural databases - distributions of manhattan worlds - perturb-and-MAP style inference for scene categorisation etc - recursive estimation of indoor Manhattan models

9.2.4 Next Steps For Semantic SLAM

We now delve into some more speculative ideas for future research of particular relevance to the ideas presented in this thesis.
TODO
- next steps for multi-view scene understanding - connecting with textons (other information too?) - occlusion reasoning - unsupervised learning of common building structures - explicitly modelling lighting -

9.3 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 maturatity of both SLAM and scene understanding as separate areas of study.

Bibliography

[APDR77] N. M. Laird A. P. Dempster and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

[BKY+08] O. Barinova, V. Konushin, A. Yakubenko, K. Lee, H. Lim, and A. Konushin. Fast automatic single-view 3-d reconstruction of urban scenes. In *ECCV*, pages 100–113, 2008.

[BS02] P. Buschka and A. Saffiotti. A virtual sensor for room detection. In *Intelligent Robots and System, 2002. IEEE/RSJ International Conference on*, volume 1, pages 637–642 vol.1, 2002.

[BSFC08] Gabriel J. Brostow, Jamie Shotton, Julien Fauqueur, and Roberto Cipolla. Segmentation and recognition using structure from motion point clouds. pages 44–57, Berlin, Heidelberg, 2008. Springer-Verlag.

[CN08] Mark Cummins and Paul Newman. Fab-map: Probabilistic localization and mapping in the space of appearance. *Int. J. Rob. Res.*, 27(6):647–665, 2008.

[Cri01] Antonio Criminisi. *Accurate visual metrology from single and multiple uncalibrated images*. Springer-Verlag New York, Inc., New York, NY, USA, 2001.

[CY99] J.M. Coughlan and A.L. Yuille. Manhattan world: compass direction from a single image by bayesian inference. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 941–947 vol.2, 1999.

[DCLK09] Martial Hebert David Changsoo Lee and Takeo Kanade. Geometric reasoning for single image structure recovery. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2009.

[EVGW+] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2009 (VOC2009) Results. http://www.pascal-network.org/challenges/VOC/voc2009/workshop/index.html.

[FCSS09] Y. Furukawa, B. Curless, S.M. Seitz, and R. Szeliski. Manhattan-world stereo. *CVPR*, 0:1422–1429, 2009.

[Fei05] Li Fei-Fei. A Bayesian hierarchical model for learning natural scene categories. pages 524–531, 2005.

[FGMR10] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. 2010.

[FH04] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59:2004, 2004.

[FP02] D.A. Forsyth and J Pnce. *Computer vision: A modern approach*. Prentice Hall, 2002.

[GEH10] Abhinav Gupta, Alexei Efros, and Martial Hebert. Blocks world revisited: Image understanding using qualitative geometry and mechanics. volume 6314, pages 482–496. Springer Berlin / Heidelberg, 2010.

[HEH05] Derek Hoiem, Alexei A. Efros, and Martial Hébert. Geometric context from a single image. pages 654–661, 2005.

[HEH06] Derek Hoiem, Alexei A. Efros, and Martial Hébert. Putting objects in perspective. pages 2137–2144, 2006.

[HHF09] Varsha Hedau, Derek Hoiem, and David Forsyth. Recovering the spatial layout of cluttered rooms. In *ICCV*, volume 2, 2009.

[HK08] Geremy Heitz and Daphne Koller. Learning spatial context: Using stuff to find things. pages 30–43, 2008.

[HZ04] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[Jeb03] Tony Jebara. Images as bags of pixels. pages 265–272, 2003.

[Jul81] B. Julesz. Textons, the elements of texture perception, and their interactions. *Nature*, 290:91–97, March 1981.

[Kal60] R. E. Kalman. A new approach to linear filtering and prediction problems. In *Journal of Basic Engineering*, November 1960.

[KM07] Georg Klein and David Murray. 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.

[KZ02a] Vladimir Kolmogorov and Ramin Zabih. What energy functions can be minimized via graph cuts? *PAMI*, 2002.

[KZ02b] Jana Koseckà and Wei Zhang. Video compass. volume 2353 of *Lecture Notes in Computer Science*, pages 4: 476–490. Springer, 2002.

[Lin93] Tony Lindeberg. *Scale-Space Theory in Computer Vision*. Kluwer Academic Publishers, Norwell, MA, USA, 1993.

[Low99] David Lowe. Object recognition from local scale-invariant features. pages 1150–1157, 1999.

[MBSL99] J. Malik, S. Belongie, J. Shi, and T. Leung. Textons, contours and regions: cue integration in image segmentation. volume 2, pages 918–925, 1999.

[MSB05] O.M. Mozos, C. Stachniss, and W. Burgard. Supervised learning of places from range data using adaboost. pages 1730–1735, 2005.

[MTKW02] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, Edmonton, Canada, 2002. AAAI.

[Par62] Emanuel Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):1065–1076, 1962.

[PSN08] Ingmar Posner, Derik Schroeter, and Paul Newman. Online generation of scene descriptions in urban environments. *Robot. Auton. Syst.*, 56(11):901–914, 2008.

[RESL98] Peter Bartlett Robert E. Schapire, Yoav Freund and Wee Sun Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, 1998.

[RM03] X. Ren and J. Malik. Learning a classification model for segmentation. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 10–17 vol.1, Oct. 2003.

[Rob65] L. Roberts. *Machine perception of 3–d solids*. PhD Thesis, 1965.

[SC87] Randall C. Smith and Peter Cheeseman. On the representation and estimation of spatial uncertainly. *Int. J. Rob. Res.*, 5(4):56–68, 1987.

[Shu99] J.A. Shufelt. Performance evaluation and analysis of vanishing point detection techniques. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(3):282–288, Mar 1999.

[SMmRB05] C. Stachniss, O. Martnez-mozos, A. Rottmann, and W. Burgard. Semantic labeling of places. In *in Proceedings of the International Symposium on Robotics Research*, 2005.

[SS01] D. Scharstein and R. Szeleski. A taxonomy and evaluation of dense two–frame stereo correspondence algorithms. In *IJCV*, 2001.

[SSN09] Ashutosh Saxena, Min Sun, and Andrew Y. Ng. Make3d: Learning 3d scene structure from a single still image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):824–840, 2009.

[Tar09] Jean-Philippe Tardif. Non-iterative approach for fast and accurate vanishing point detection. volume 2, 2009.

[Tor03] Antonio Torralba. Contextual priming for object detection. 53(2):169–191, 2003.

[Vap95] Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer–Verlag, New York, 1995.

[VGNS07] Shrihari Vasudevan, Stefan Gächter, Viet Nguyen, and Roland Siegwart. Cognitive maps for mobile robots-an object based approach. *Robot. Auton. Syst.*, 55(5):359–371, 2007.

[VZ05] Manik Varma and Andrew Zisserman. A statistical approach to texture classification from single images. 62(1-2):61–81, 2005.

[ZGWX05] S.-C. Zhu, C.-E. Guo, Y. Wang, and Z. Xu. What are textons? *IJCV*, pages 121–143, 2005.