TOYOTA TECHNOLOGICAL INSTITUTE AT CHICAGO

A MACHINE LEARNING APPROACH TO RECOVERY OF SCENE GEOMETRY FROM IMAGES

A DISSERTATION SUBMITTED TO THE THESIS COMMITTEE
IN CANDIDACY FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE

BY

HOANG TRINH

CHICAGO, ILLINOIS

MAY 2010

A MACHINE LEARNING APPROACH TO RECOVERY OF SCENE GEOMETRY FROM IMAGES

A thesis presented

by

HOANG TRINH

in partial fulfillments of the requirements for the degree of

Doctor of Philosophy in Computer Science.

Toyota Technological Institute at Chicago

Chicago, Illinois

MAY 2010

— Thesis Committee —

Committee member (print)	Signature	Date
Committee member (print)	Signature	Date
Committee member (print)	Signature	Date
Thesis/Research Advisor (print)	Signature	Date
Chief Academic Officer (print)	Signature	Date

ABSTRACT

Recovering the 3D structure of the scene from images yields useful information for tasks such as shape and scene recognition, object detection, or motion planning and object grasping in robotics.

In this thesis, we introduce a general machine learning approach called unsupervised CRF learning based on maximizing the conditional likelihood. We describe the application of our machine learning approach to computer vision systems that recover the 3-D scene geometry from images. We focus on recovering 3D geometry from single images, stereo pairs and video sequences. Building these systems requires algorithms for doing inference as well as learning the parameters of conditional Markov random fields (MRF). Unlike previous work, our system is trained unsupervisedly without using ground-truth labeled data.

We employ a slanted-plane stereo vision model in which we use a fixed over-segmentation to segment the left image into coherent regions called superpixels. We then assign a disparity plane for each superpixel. We formulate the problem of inferring plane parameters as an MRF labelling problem, which can be solved by an energy minimization method. The MRF is a graphical model in which superpixels define nodes and the adjacency between superpixels define edges. Our stereo energy function balances between a data matching term and a smoothness term.

For systems with continuous valued variables, or discrete-valued variables with very large state space, it is impossible to directly use a standard discrete MRF inference techniques such as Loopy BP, graph cuts or tree-reweighted message passing. For such systems, we propose to use a generic Particle-based Belief Propagation (PBP) algorithm closely related to previous work, which we then formulate specifically for our MRF labeling problems. Although we only describe a specific use of this generic PBP algorithm, we believe it can be used as an approximate inference scheme for a wide variety of problems that can be formulated by a probabilistic graphical model, especially those containing many random variables with very large or continuous domains.

We demonstrate the use of our unsupervised CRF learning algorithm for a parameterized slanted-plane stereo vision model involving shape from texture cues. This unsupervised learning algorithm implicitly trains shape from texture monocular surface orientation

cues. We exhibit that training monocular cues from stereo pair data improves stereo depth estimation. Our stereo model with texture cues, only by unsupervised training, outperform the results in related work on the same stereo dataset.

Our unsupervised learning method is also implemented for the monocular depth estimation (MDE) problem. The MDE model, learned using stereo pairs only, demonstrates a modest improvement after a few training steps, and achieve performance comparable to previous work on the same dataset. The use of MDE in combination with the dense stereo model also introduces a small boost in depth estimation over the initial stereo model.

In this thesis, we also address the use of stereo video sequences. We formulate structure and motion estimation as an energy minimization problem, in which the model is an extension of our slanted-plane stereo vision model that also handles surface velocity. Surface estimation is done using our own slanted-plane stereo algorithm. Velocity estimation is achieved by solving an MRF labeling problem using Loopy BP. Performance analysis is done using our novel evaluation metrics based on the notion of view prediction error. Experiments on road-driving stereo sequences show encouraging results.

ACKNOWLEDGMENTS

It has been an honor for me to be advised by Prof. David McAllester, who has been giving me his outstanding mentorship, his patience and unconditional support from the first day to the last day of my graduate program at TTI-C. Through him, I also got to understand more about how to think and act as a true scientist. I would never have come this far without his guidance and encouragement.

Many thanks go to my thesis committee members: Greg Shakhnarovich, Raquel Urtasun, Ronen Basri for their valuable and constructive advices and feedbacks.

During the last few years, I have received a lot of help and support from people in TTI-C, CS Department and elsewhere that I want to give thanks to: Pedro Felzenszwalb for his inspiration that fired up my passion in Computer Vision; Alex Ihler, Ronen Basri, Deva Ramanan, Xiaofeng Ren, Cristian Sminchisescu, Yali Amit, John Langford for helping me find answers to lots of my questions and puzzles; Wonseok Chae, Andy Cotter and Karthik Sridharan for being my friend and making me feel less alone as a TTI-C student; Gary Hamburg, Carole Flemming, Christina Novak, Adam Bohlander, Julia MacGlashan, Dawn Gardner and others, who have created a family-like environment and atmosphere at TTI-C, where I consider my second home away from home during the last few years; Nathan Srebro and Umut Acar who gave me advices as Academic advisors; Seiichi Mita and TTI-J Nagoya for welcoming me and giving an warm experience in Nagoya; Mr. Kien Pham and Adam Kalai, through whom I got to know about TTI-C and then became one of the first students here; Robinson, KC Lee, Katharine, Paul King, Dan Prochaska for their support and guidance during my summer internship. Special thanks go to my friends and my family. Last but not least, I want to thank my parents and my beloved $Nhung$.

This thesis was partly funded by a grant from the Vietnam Education Foundation (VEF).

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

One of the most important characteristic of natural scenes is their underlying 3D geometric structure. Recovering the 3D structure of the scene from images yields a great amount of useful information for other computer vision tasks such as shape and scene recognition, object detection, or motion planning and object grasping in robotics. In this thesis, we focus mainly on the problem of scene geometry recovery and motion estimation. Concretely we want to solve the problems of monocular depth estimation, stereo vision with monocular cues, and structure and motion estimation from stereo video. We attempt to solve these problems at different levels, using different forms of input image data and different types of image features.

Markov random fields (MRFs) are a very powerful tool used for modeling a wide range of Vision problems. Throughout this thesis, we will demonstrate that all the problems we are interested in can be modeled using MRFs or conditional random fields (CRFs), a variant of MRFs. A major part of this thesis consists of formulating general learning and inference algorithms for MRFs. The probabilistic inference and learning techniques we develop to solve our specific Vision problems can also be generalized to other problems modeled by MRFs and CRFs.

1.1 Markov Random Fields and Related Formalisms

1.1.1 MRF

An MRF is considered a particular type of graphical model that allows us to efficiently compute marginal distributions. One representation of an MRF is a factor graph, or hypergraph, in which each hyperedge can connect more than one (regular) node. MRFs have been widely used for decades as statistical models in a variety of application areas (Hammersley and Clifford 1971; Kindermann and Snell 1980; Li 1995). An MRF defines a probability distribution on the joint assignments to (configurations of) a set of random variables.

Let $Y = (y_1, \ldots, y_N)$ be a set of N random variables. Let $Y_1, \ldots, Y_M$ be M subsets of Y. Let $F = (f_1, \ldots, f_M)$ be factors where f_α is a function on assignments of values to the variables in Y_α.

$$p_\beta(y) = \frac{1}{Z_\beta} \prod_{\alpha=1}^{M} f_{\beta,\alpha}(y_\alpha) \tag{1.1}$$

where

$$Z_\beta = \sum_{y} \prod_{\alpha=1}^{M} f_{\beta,\alpha}(y_\alpha) \tag{1.2}$$

Here y is an assignment of values to all random variables in Y, each y_α is the projection of y onto the variables in the subset $Y_\alpha \subseteq (y_1, \ldots, y_N)$. β is the mode parameter. Each function f_α is called a factor or sometimes called a local function, and is defined only on the subset Y_α. The set of variables Y and the set of factors F define the MRF.

For inference in MRFs, the goal is to find the assignment of values to all variables maximizing the joint probability:

$$y^* = \underset{y}{\operatorname{argmax}}(p_\beta(y)) \tag{1.3}$$

For training, given the i.i.d sampled training data: $y^1, \ldots, y^T$, the objective is to maximize the joint probability over the whole training corpus:

$$\beta^* = \underset{\beta}{\operatorname{argmax}} \prod_{t=1}^{T} (p_\beta(y^t)) \tag{1.4}$$

The most popular version of MRFs in computer vision is the pairwise MRF model, in which each factor is defined over at most two random variables.

1.1.2 CRF

One notable variant of an MRF is a conditional random field (CRF), introduced by Lafferty et al. in (Lafferty et al. 2001). While MRFs model joint probabilities, CRFs model conditional probabilities. Similar to an MRF, a CRF is an undirected graphical model representing random variables and dependencies between random variables. In a CRF the variables are divided into two groups — exogenous and dependent. A CRF defines a conditional probability of the dependent variables given the exogenous variables and (importantly) does not model the distribution of the exogenous variables. The advantage is that CRFs can estimate the distribution of interest more accurately than MRFs, since there is no danger of corrupting the model by modeling the exogenous variables poorly. For instance,

in the case of stereo vision one might expect that it is easier to model the probability distribution of the right image given the left image than to model a probability distribution over images.

Let $x = (x_1, \dots x_N)$ represent a set of exogenous variables. In a CRF, we want to compute the conditional probability distribution of the dependent variables y with respect to the exogenous variables x.

$$p_\beta(y|x) = \frac{1}{Z_\beta(x)} \prod_{\alpha=1}^{M} f_{\beta,\alpha}(y_\alpha, x_\alpha) \tag{1.5}$$

where the partition function $Z_\beta(x)$ is defined in a similar way to (1.2).

The CRF inference equation:

$$y^* = \underset{y}{\operatorname{argmax}}\, p_\beta(y|x) \tag{1.6}$$

The CRF training equation:

$$\beta^* = \underset{\beta}{\operatorname{argmax}} \prod_{t=1}^{T} (p_\beta(y^t|x^t)) \tag{1.7}$$

1.1.3 Hidden CRF

A variation of CRFs is the hidden CRFs (HCRF) proposed by Quattoni et al. in (Quattoni et al. 2007). In HCRFs, additional latent variables $Z = (z_1, \dots z_K)$ are introduced, which are not observed in the training data, but are part of the model.

$$p_\beta(y, z|x) = \frac{1}{Z_\beta(x)} \prod_{\alpha=1}^{M} f_{\beta,\alpha}(y_\alpha, x_\alpha, z_\alpha) \tag{1.8}$$

where the partition function $Z_\beta(x)$ is defined in a similar way to (1.2).

For inference, HCRFs predict the dependent variable y by computing $p(y|x)$ by marginalizing the hiden variable z. The HCRF inference equation:

$$y^* = \operatorname*{argmax}_y p_\beta(y|x) \tag{1.9}$$

$$= \operatorname*{argmax}_y \sum_z p_\beta(y, z|x) \tag{1.10}$$

In training, given the training data: $(x^1, y^1), \ldots, (x^T, y^T)$ where x the exogenous variable, and y is the dependent variable, the objective of HCRF is to maximize the following conditional probability:

$$\beta^* = \operatorname*{argmax}_\beta \prod_{t=1}^{T} p_\beta(y^t|x^t) \tag{1.11}$$

where:

$$p_\beta(y|x) = \sum_z p(y, z|x) \tag{1.12}$$

1.1.4 Unsupervised CRF

We introduce another variant of the CRF, which we call unsupervised CRF (UCRF). In unsupervised learning one usually formulates a parameterized probability model and seeks parameter values maximizing the likelihood of the unlabeled training data. Our approach to unsupervised learning is based on maximizing the conditional likelihood.

The defining equation of UCRF is similar to HCRF up to some changes in variable naming. Here x and u denote the exogenous variables and y is the dependent variable. In the stereo vision example, x and u correspond to the left and right images of the stereo pair, while y is the hidden-state variable corresponding to the depth map of the left image.

$$p_\beta(u, y|x) = \frac{1}{Z_\beta(x)} \prod_{\alpha=1}^{M} f_{\beta,\alpha}(u_\alpha, y_\alpha, x_\alpha) \tag{1.13}$$

where the partition function $Z_\beta(x)$ is defined in a similar way to (1.2).

For inference, UCRF aims at predicting y by computing the following:

$$y^* = \operatorname*{argmax}_y p_\beta(y|x, u) \tag{1.14}$$

or

$$y^* = \operatorname*{argmax}_y p_\beta(y|x) \tag{1.15}$$

Note that (1.14) looks at both x and u, while (1.15) only looks at x, which in this case u can be considered latent variable. Later we will show that (1.14) is applied to our stereo

inference algorithm, while (1.15) is applied to inference in our monocular depth estimator.

In training, given the training data: $(x^1, u^1), \ldots (x^T, u^T)$, the objective is to find the set of parameters maximizing the conditional probability of arbitrary random variables, both are observed by the training data.

$$\beta^* = \underset{\beta}{\operatorname{argmax}} \prod_{t=1}^{T} p_\beta(u^t|x^t) \tag{1.16}$$

where:

$$p_\beta(u|x) = \sum_y p(u, y|x) \tag{1.17}$$

Note that in HCRF, the test time dependent variable y is part of the training data, while the latent variable z is not. In UCRF, the latent variable u is part of the training data, however the test time dependent variable y, which needs to be predicted during inference, is not included in the training data.

We will then introduce a hard EM learning algorithm for maximizing the conditional likelihood, where we use our Particle-based Belief Propagation inference algorithm to perform the hard E step, and contrastive divergence to update model parameters in the hard M step. Details are covered in Section 1.4.

Throughout the thesis, we emphasize the employment of unsupervised CRFs learning for our vision models. Section 1.4 will succeed this discussion with more details about the components of our learning approach.

A main portion of the thesis involves applying our unsupervised CRF model for the scene geometry estimation problem, in particular the stereo vision problem. Here stereo

vision provides a simple setting to investigate unsupervised learning and hence seems a good place to start. However we believe that our approach to unsupervised learning based on maximizing conditional likelihood can be generalized to other, maybe much more sophisticated models.

1.2 MRFs and CRFs for Our Vision Problems

Many problems in computer vision can be interpreted as pixel-labeling problems: the objective is to assign labels to pixels or groups of pixels. For instance, in two-view stereo vision the labels can be disparity values, in segmentation the labels are indices of different groups that the pixels belong to, etc.

Pixel-labeling problems are usually solved by minimizing an energy function having two terms: a data term penalizing assignments that are inconsistent with the observed data, and a smoothness term enforcing local smoothness (spatial coherence). (Geman and Geman 1984) have pointed out that optimizing an energy function of this form is equivalent to estimating the maximum a posteriori probability of an MRF.

In this thesis we mainly focus on the problem of recovery of 3D scene structure. Scene structure can be recovered from different forms of input image data, ranging from a single image to a multiple view video sequence. The list of most commonly used forms of data is shown in Figure 1.1. All of these image data forms are investigated in this thesis, with main focus on single images, stereo image pairs and stereo image pair sequences. Since images are 2-D representations created by perspective projection of 3-D scenes, there is a natural intrinsic ambiguity for recovering 3-D information from local image features. Increasing the amount of input image data can reduce the difficulty of the problem.

In vision systems, depending on the form of image data used, different types of energy terms may arise. For the vision systems addressed in this thesis, the following types of

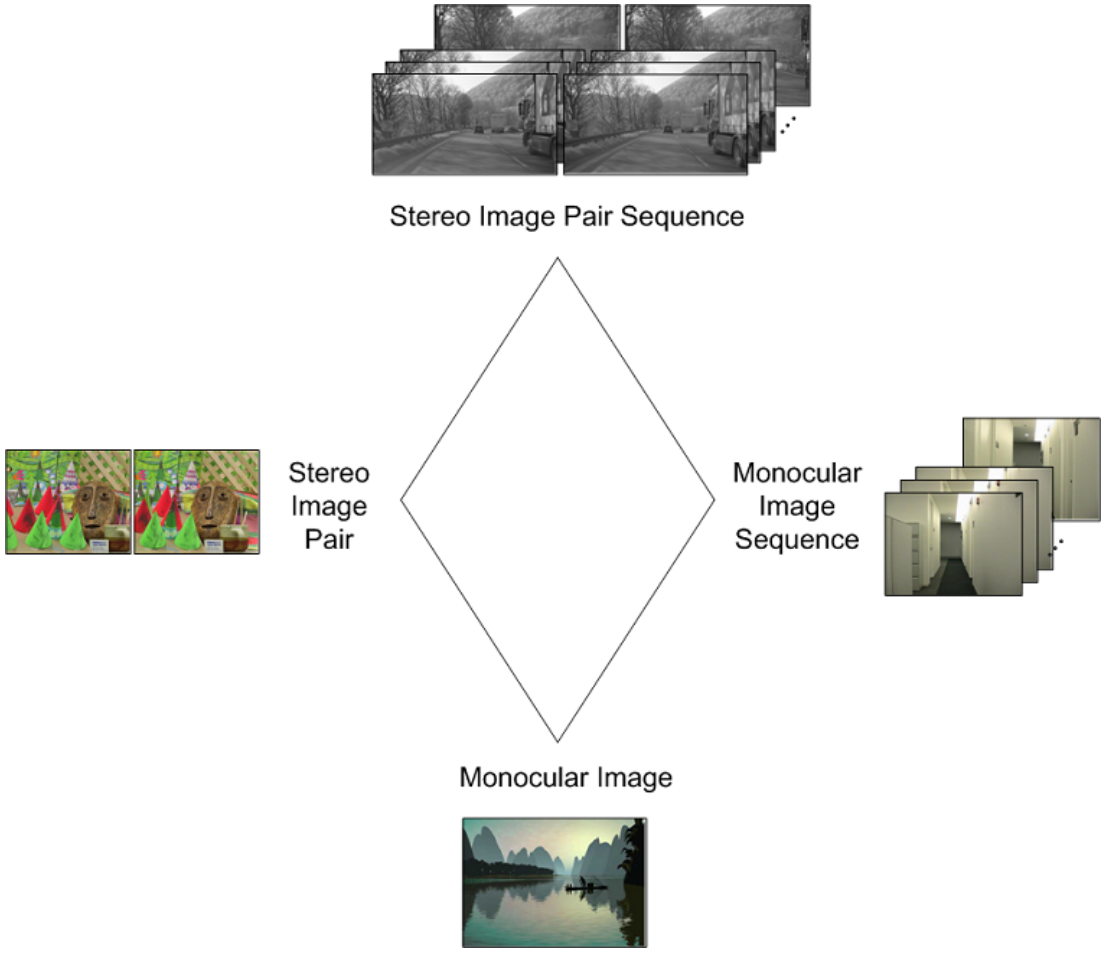


Figure 1.1: Different forms of input image data for recovering scene geometry.

energy terms are exploited:

- Monocular terms: depending on a single image, (e.g: intensity, gradient, SIFT, HOG, etc.)
- Stereo terms: involving a stereo pair of images.
- Motion terms: involving two consecutive video frames.
- Smoothness terms: involving only the output pixel-labeling variables.

As we mentioned earlier, many computer vision problems can be interpreted as pixel-labeling problems. The types of the output labels (dependent variables) are defined by what the vision problem is trying to solve specifically, i.e. the quantities of interest that we want to infer about the underlying scene. An image denoising algorithm would want to assign an intensity value or a color to each pixel, while a stereo algorithm would want to assign a disparity value. In this thesis, our output label types of interest are:

- 3D point locations: for sparse Structure from Motion.
- Disparity values: for monocular depth estimation and dense stereo depth estimation.
- Disparity plane parameter vectors: for slanted-plane stereo model with monocular surface orientation cues.
- Disparity plane parameter vectors and motion vectors: for structure and motion estimation from video sequences.

Any vision problem we address in this thesis can be formulated by a combination of the subsets of each of the three elements above (input image data, types of energy terms, types of output labels). However, in any case, the problem can still be modeled by an MRF.

Application	**Data**	x	y	u
Stereo	stereo pairs	left	depth	right
Stereo+Mono	stereo pairs	left	depth	right
Mono	single image	image	depth	
Motion	stereo video	1st frame	depth+motion	2nd and 3rd frames

Table 1.1: The type of visual data and the notations of variables in our vison applications.

1.3 Our CRF Formulations for Scene Geometry Recovery Problems

1.3.1 Unsupervised CRF Formulation for Stereo Vision

A major part of the thesis focuses on the problem of recovering scene structure from stereo image pairs, with emphasis on monocular surface orientation cues - evidence of surface orientation based on a single image. For the current discussion, let's assume we are not using the monocular cues.

A simple version of the scene structure recovery problem is dense two-view stereo vision, in which the depth of each pixel is recovered from a stereo image pair taken from a stereo camera. The depth of each pixel can be related to the horizontal disparities between the two stereo images. A dense output is returned - a disparity is assigned to each pixel in the reference image. In this problem, the exogenous variable x is the left image, the dependent variable y is the disparity value assignment to all pixels in x, and the exogenous variable u is the right image. A thorough, although not very recent, survey of state of the art dense stereo vision approaches was described in (Scharstein and Szeliski 2002). An online evaluation of the best two-image stereo matching algorithms, along with test images, ground truth, and software can be found in (Scharstein et al. 2001).

The shift from the dense stereo model to slanted-plane stereo model started with the work by BirchField and Tomasi in (Birchfield and Tomasi 1999). Since then, quite a few other stereo algorithms have been applying this slanted-plane model, among which are state-of-the-art algorithms in the current Middlebury benchmark (Klaus et al. 2006; Wang and Zheng 2008; Q. Yang and Nistr 2008). The model uses the assumption that the scene is composed of a set of planar surfaces. The algorithms using this model start by segmenting the image into homogeneous regions called superpixels, using image segmentation. The superpixels found by the segmentation method are coherent groupings of pixels hav-

ing similar properties (color, texture, etc). Therefore it is just natural to assume that each superpixel corresponds to a planar surface in the scene, which from now on we will call disparity plane. Next, the goal of the stereo algorithm is to infer the location and orientation (or surface normal) of each of these planes - which are fully defined by the 3 plane parameters. After that, the disparity of each pixel can be directly computed from the disparity map equation, with sub-pixel precision.

Our stereo vision model is a slanted plane model in which we use a fixed over-segmentation of the left image into superpixels. The goal of inference is to assign a disparity plane for each superpixel. In other plane-based stereo algorithms, the authors mostly use a local matching step to extract reliable correspondences, followed by robust plane fitting to fit a disparity plane into each superpixel separately. The process is then repeated for iterative plane refinement. Our stereo inference algorithm also infers a disparity plane for each superpixel. However we use the Particle BP inference algorithm to simultaneously find the optimal joint assignment of all plane parameters to all superpixels. The optimal assignment is found by minimizing a high-dimensional global energy function. In this Particle BP inference algorithm, each particle, which represents a proposal disparity plane for a specific superpixel, gets updated over time, instead of being fixed after the plane fitting step, as in other slanted-plane stereo algorithms. Chapter 2 will discuss this algorithm in details.

We formulate the problem of inferring plane parameters as an UCRF. The UCRF is a graphical model in which superpixels define nodes and the adjacency between superpixels define edges. In this case, the input image data are stereo image pairs; the energy terms involved are stereo and smoothness terms, and the hidden labels here are the parameters of disparity planes corresponding to superpixels.

In this problem, the exogenous variable x is the left image, the dependent variable y is the disparity plane assignment to all superpixels in x, and the exogenous variable u is the

right image. x and u are in the training data, while y is not.

1.3.2 Unsupervised CRF Formulation for Stereo Vision Models with Monocular Cues

Our stereo vision model is a slanted plane model involving monocular shape from texture cues. The introduction of an additional monocular term into the slanted-plane stereo model is a novel idea that has not been investigated.

One interesting point is that monocular cues can easily be incorporated into a stereo model and can be used as an additional monocular term. The monocular depth cues predicting pixel disparity values directly can be added into a standard dense stereo model, without breaking the UCRF formulation of the model. For this UCRF, x and u form the stereo pair, and y is the assignment of disparity values to all pixels in x. Analogously, the surface orientation cues predicting the surface normal of superpixels can be integrated into our existing slanted-plane stereo model as an additional energy term. For this UCRF, x and u is the stereo pair, y is the assignment of disparity planes to all superpixels in x.

Our UCRF models now involve three terms: a data matching energy (stereo term) measuring the degree to which the left and right images agree under the induced correspondence, a smoothness energy (smoothness term) measuring the local smoothness of the assigned scene structure, and a texture energy (monocular term) measuring the degree to which the assigned labels agree with the labels predicted by the monocular predictor.

1.3.3 Unsupervised CRF Formulation for Depth Estimation from a Single Image

Recently, there has been more interest in the problem of estimating 3-D structure from one single image. While this is a very interesting problem, it remains extremely more challenging than the stereo vision problem, due to the high ambiguity between local image features and the 3-D point locations, due to perspective projection. Some methods such as Shape from Shading (R. Zhang and Shah. 1999) make specific assumptions about the scene and the formation of the image, and rely mostly on photometric cues such as lighting and shading to recover shapes of surfaces in the scene. Several more efforts by Hoiem (Derek Hoiem 2005) and Saxena (Ashutosh Saxena and Ng 2007b) have demonstrated quite encouraging results. Both of these approaches focus on supervised learning to capture the relationship between the local image features and geometric properties (orientation, location) of regions of the scene, as well as the relationship between different regions. In this thesis we use UCRF learning to train a monocular depth estimator. For MDE, we model the disparity of a pixel as a linear function of the monocular feature vector and a parameter vector. The objective of the learning algorithm is to train this monocular feature vector. A monocular energy term is introduced, penalizing the difference between the assigned disparity and the monocularly predicted disparity. An energy function formulated by combining this monocular energy term with a smoothness term itself defines an UCRF. In this UCRF, the exogenous variable x is the left image, or its feature-based representation, the dependent variable y is the disparity map for all pixels in x, and there is no latent variable u.

For inference, we use equation (1.14), i.e, we want to infer the depth map y only by looking at the left image x. Training is formulated by incorporating the monocular energy term into a standard pixel-based UCRF stereo model. At each training step, we do infer-

ence on this joint model to ontain the new depth map, then we use this updated depth map to retrain the MDE. We then implement our UCRF learning algorithm for monocular depth estimation for learning the monocular parameter vector using stereo image pairs only. The experimental results show that the monocular model is able to provide good depth predictions from different scenes in general, although there were still image parts which were not generalized well, i.e, it tends to perform more poorly in predicting depth for image patches that were too different from those in the training data.

We also discuss in this thesis the possibility of using histograms of oriented gradients (HOG) feature as monocular surface orientation cues. We derive a formal relationship between a variant of HOG features and surface orientation. Although our observation that there should be a statistical relationship between HOG features and surface orientation is a simple result in the area of shape from texture, HOG features have only recently gained popularity and to our knowledge the idea of using HOG as a surface orientation cue has not been previously noted.

1.3.4 CRF Formulation for Structure and Motion Estimation from Stereo Videos

Multi-view video sequence is arguably the richest form of image data for scene reconstruction. A subtype in this form that we consider very interesting are stereo sequences. This data form is interesting for two reasons. First, as opposed to a multi-view video sequence, a stereo sequence can easily be captured using only one moving calibrated binocular camera. Second, this data form provides us with enough constraints to conveniently compute depth and motion of scene points simultaneously, since coupling dense stereo matching with motion estimation helps decrease the number of unknowns per image point. More specifically, for stereo sequences, we can formulate structure and motion estimation as an energy mini-

mization problem, in which the model is either an extension of a stereo vision model to also handle scene motions, or an extension of a Structure from Motion or optical flow model under a stereo setup. So far, to our knowledge, this useful data form has hardly been investigated and exploited for the purpose of recovering scene structure and motion. One of the very few research work that has discussed this issue is (Huguet and Devernay 2007), in which the authors presented a variational method for scene flow estimation from a calibrated stereo image sequence. In Chapter 6, we extend the constructed slanted-plane stereo model to handle structure and motion estimation on road-driving stereo sequences. Based on specific assumptions about the motion of the camera and the scene, we can reduced the 2D optical flow problem to a 1D velocity value problem. Our algorithm iteratively and alternately solve for structure (disparity planes) and motion (velocity values). Surface estimation is done using our slanted-plane stereo algorithm. Velocity estimation is achieved by solving a CRF labeling problem using Loopy BP. Experiments on road-driving stereo sequences showed encouraging results. Performance analysis was done using our novel evaluation metric based on the notion of view prediction error.

For this CRF, x is the left frame of the stereo pair at time t, u is composed of the right frame at time t and the left frame at time $t+1$, and y is the assignment of disparity planes and velocity values to all superpixels in the left frame at time t. The dependent variable y is not known in the training video sequences.

1.4 Training Methods for Unsupervised CRFs

As mentioned earlier in Section 1.1.4, our unsupervised CRF learning approach is based on maximizing conditional likelihood. This section covers more details about our learning approach. Here we introduce a hard EM algorithm for maximizing the conditional likelihood, where we use contrastive divergence to update model parameters in the hard M step.

1.4.1 Hard Conditional EM

Depending on the specific use of the model, (1.17) can be further factorized as follows when necessary:

$$P_\beta(u|x) = \sum_y P_\beta(u, y|x) = \sum_y P_{\beta_y}(y|x) P_{\beta_u}(u|x, y) \tag{1.18}$$

The conditional probability at the right hand side of (1.14) can be represented as follows.

$$P_\beta(y|x, u) = \frac{P_\beta(y, u|x)}{P_\beta(u|x)} = \frac{P_\beta(y, u|x)}{\sum_y P_\beta(u, y|x)} \tag{1.19}$$

For soft conditional EM, the goal of the E step is to estimate the probability distribution over the latent variables given the training data - this distribution can be interpreted using equation (1.19), while the goal of the M step is to maximize the model over complete data. Equation (1.19) justifies for our argument that the most likely value of the dependent variables given the model and the observed data, which is the result of the inference step, is also the value maximizing the conditional likelihood. Note that the transformation from (soft) conditional EM to hard conditional EM is done by replacing the sum in equation (1.17) by the max. Hard EM works with the single most likely value of y instead of the distribution over y.

Hard conditional EM is defined to be the process of iterating the following updates:

$$y_i := \operatorname*{argmax}_y P_\beta(u_i, y|x_i) \tag{1.20}$$

$$\beta := \operatorname*{argmax}_\beta \sum_{i=1}^{N} \ln P_\beta(u_i, y_i|x_i) \tag{1.21}$$

We will call (1.20) the hard E step and (1.21) the hard M step. We implement the hard E step using our Particle-based Belief Propagation inference algorithm that we introduce next in Section 1.5. For the M step, we use a learning technique called contrastive divergence, more details of which will follow in Sections 1.4.2 and 4.2. These two steps are the foundation to the hard conditional EM algorithm that we will elaborate later in Chapter 4.

1.4.2 Contrastive Divergence

In order to learn the parameters of our unsupervised CRF model, we use contrastive divergence in the hard M step of our hard EM algorithm. Contrastive divergence first introduced by G. Hinton et al. in (Hinton 2002; Carreira-Perpiñán and Hinton 2005) is a general MRF learning algorithm capable of training large models. Other work that also described learning on MRF using contrastive divergence is the Field of Experts system by Roth and Black in (Roth and Black 2005).

Briefly speaking, contrastive divergence is a tool that allows us to estimate the gradient of a energy function, even though we cannot evaluate the energy function itself. Using contrastive divergence in the hard M step allows us to compute the gradient of the energy function with respect to the model parameters β. We can then update β using gradient descent. By running only a few (one or two) MCMC steps, starting from the current estimation of the latent variable, contrastive divergence reduces significantly both the computation and

the variance compared to running standard MCMC processes. However, since this is an approximate method (with finite number of samples), there is actually no known guarantee for convergence.

1.4.3 Related work

The most closely related work seems to be that of Zhang and Seitz (Zhang and Seitz 2007). They give a method for adapting five parameters of a stereo vision model including the weights for the match and smoothness energies as well as robustness parameters. The five parameters are tuned to each individual input stereo pair, although the method could be used to tune a single parameter setting over a corpus of stereo pairs. The main difference between their work and ours is that we train highly parameterized monocular depth cues. Another difference is that we formulate a general CRF-like model for unsupervised learning based on maximizing conditional likelihood and avoid the need for the independence assumptions used by Zhang and Seitz by using contrastive divergence — a general method for optimizing loopy CRFs (Hinton 2002; Carreira-Perpiñán and Hinton 2005).

There is also related work on learning highly parameterized monocular depth cues by Saxena et al. in (Ashutosh Saxena and Ng 2007b,a), as well as Hoiem et al. in (Derek Hoiem 2005). The main difference between these methods and ours is that we use unsupervised learning while they use ground truth data to train their system. One might argue that stereo pairs constitute supervised training of monocular depth cues. A standard stereo depth algorithm could be used to infer a depth map for each pair which could then be used in a supervised learning mode to train monocular depth cues. However, we demonstrate that training monocular depth cues from stereo pair data improves *stereo* depth estimation. Hence the method can be legitimately viewed as unsupervised learning of a stereo depth. We also demonstrate later that our stereo model with texture cues, only

by unsupervised training, outperformed the results in (Ashutosh Saxena and Ng 2007a).

Other related work includes that of Scharstein and Pal (Scharstein and Pal 2007) and Kong and Tao (Kong and Tao 2004). In these cases somewhat more highly parameterized stereo models are trained using methods developed for general CRFs. However, the training uses ground truth depth data rather than unlabeled stereo pairs.

1.5 Inference using Particle-based Belief Propagation

Inference problems arise from various scientific areas such as AI, statistical physics, computer vision, coding theory. In MRFs,the goal of inference is to find the values of the latent variables achieving the maximum a posteriori of the MRF. Inference is also a vital component of MRFs learning algorithms in general, and of our learning approach in particular. By performing inference, we want to estimate the most likely value of the latent variables given the model and the observed data, which can be formulated by the following equation.

$$y^* \quad := \quad \underset{y}{\operatorname{argmax}}\, p_\beta(y|x,u) \tag{1.22}$$

From equation (1.19), we can see that (1.22) is equivalent to (1.20), which is exactly the hard E step in our learning algorithm described in Chapter 4. In that chapter we will show that the inference step minimizes an energy function defined by our MRF model.

Recent energy minimization techniques such as Loopy Belief Propagation (LBP) (Pearl 1988; J. Yedidia and Weiss 2000) or graph cuts (Y. Boykov and Zabih 2001) have shown superior performance on MRFs than were previously possible. According to the widely used Middlebury stereo benchmarks (Scharstein and Szeliski 2002; Scharstein et al. 2001), almost all the best-performing stereo methods rely on graph cuts or LBP. However, these

methods are known to mainly perform well on systems of many variables, each of which has a relatively small state space (number of possible values), or particularly nice parametric forms (such as jointly Gaussian distributions). For systems with continuous valued variables, or discrete-valued variables with very large domains, it is impossible to directly use a standard discrete MRF inference techniques such as Loopy BP, graph cuts or tree-reweighted message passing. In our slanted plane stereo model, the MRF is a graphical model in which segments define nodes and the adjacency between segments define edges. Since the inference problem is to assign a disparity plane to segments, the state space of each node is a continuous 3-dimensional vector space. Similarly in our MRF model for Structure from Motion, our graphical model consists of two types of nodes, in which each camera node has 6-D state space, and each map node has a 3-D state space. For such systems, we propose to use a generic Particle-based Belief Propagation (PBP) algorithm closely related to previous work in (Koller et al. 1999), which we then formulate specifically for our MRF labeling problems. The popularity of particle filtering for inference in Markov chain models defined over random variables with very large or continuous domains makes it natural to consider sample-based versions of belief propagation (BP) for more general (tree structured or loopy) graphs. The algorithm creates an initial set of values for each node, representing samples from the posterior marginal. It then continues to improve the current marginal estimate by constructing a new sampling distribution and draw new sets of particles. The consistency of PBP has been theoretically justified in (Ihler and McAllester 2009). The authors showed that PBP approaches the true BP messages as the number of samples grows large. They used concentration bounds to analyze the finite-sample behavior, giving a convergence rate of $O(1/\sqrt{n})$ where n is the number of samples, independent of the domain size of the variables. Our empirical results also show that the algorithm is consistent in finite case, i.e. it converges to the optimal values of the message

and belief functions with finite samples. PBP also provides estimates of uncertainty in the form of an approximate posterior density function. Although we only described a specific use of this generic PBP algorithm, we believe it can be used as an approximate inference scheme for a wide variety of problems that can be formulated by a probabilistic graphical model, especially those containing many random variables with very large or continuous domains.

Similar to standard belief propagation, there are also two different versions of implementation for PBP - sum-product and max-product. We use the max-product implementation for the hard E step of our learning algorithm. We have also implemented the sum-product version for some other vision experiments that we describe later in Chapter 2.

1.6 Contributions of the Thesis

The main contributions of the thesis are threefold.

First, we introduce a novel general machine learning approach for unsupervised learning on CRFs, based on maximizing the conditional likelihood. We formulate our learning algorithm using hard EM. The hard E step can be replaced by any inference algorithm based on each specific application. The hard M step uses contrastive diveregence to update the model parameters.

Second, we investigate the relationship between HOG features and the planar surface orientation. We observe that HOG, which describe the edge distribution, are good cues to predict orientation of surfaces. We then succesfully justified our claim, both mathematically and practically.

Third, we succesfully applied our machine learning approach into building several vison systems for 3D scene structure recovery. For example, we used our using unsupervised learning technique to train a plane-based stereo model with surface orientation cues.

1.7 Applications to Scene Geometry Estimation

The vision problems we focus on in this thesis are monocular depth estimation, stereo vision with monocular cues, scene structure and motion estimation. We design and implement our own algorithms solving each of these problems. Quantitative analysis is a crucial step that helps evaluating the performance of our algorithms. In this thesis, we demonstrate our experimental results for each of our following vision systems:

- In Chapter 2 we present our Particle-based Belief Propagation inference algorithm and some of its applications in vision problems.

- In Chapter 4, we demonstrate our application of our unsupervised learning algorithm for stereo vision with monocular cues - the stereo model that we describe in Chapter 3. The experimental results on two different datasets help validating our learning approach, as well as testing the performance of the final trained stereo model.

- In Chapter 5, we describe our unsupervised CRF learning approach to monocular depth estimation.

- In Chapter 6, we propose a novel evaluation metric, based on the notion of view prediction error. This metric can be used to evaluate the performance of structure and motion estimation algorithms on stereo sequences without ground truth data. Experimental results on road-driving stereo sequences demonstrate that our algorithm successfully improves the view prediction error although it was not designed to directly optimize this quantity.

1.8 Thesis Organization

The thesis is organized as follows. In Chapter 2 we describe in details the application of our Particle-based Belief Propagation (PBP) inference algorithm to several Vision problems that we address throughout this thesis. Chapter 3 present our slanted-plane stereo model with monocular cues, and the use of Histogram of Gradients (HOG) monocular feature as surface orientation cues. Chapter 4 explains our unsupervised CRF learning approach using hard conditional EM. Experimental results of both learning and inference on two different stereo datasets are also demonstrated in this chapter. We present our learning approach to the problem of monocular depth estimation (MDE) in Chapter 5. The unified framework solving for depth and motion simultaneously is introduced in Chapter 6. We address conclusions in Chapter 7.

CHAPTER 2

INFERENCE USING PARTICLE-BASED BELIEF PROPAGATION AND ITS APPLICATIONS FOR VISION PROBLEMS

Although the main emphasis of the thesis is on our unsupervised CRF learning algorithm, one key component that takes part in both the training process and the testing process of our learning approach is the inference algorithm. To prepare the audience with necessary technical details in order to have a better understanding of unsupervised CRF learning, we decide to introduce our inference algorithm earlier in the thesis.

In this chapter, we first review some background technical details based on which we construct our PBP inference algorithm. Next we describe in details our PBP inference algorithm that we mentioned earlier in Section 1.5, followed by its applications to the following Vision problems:

- Structure from Motion (SfM).
- The dense stereo vision problem with unknown camera constraints.
- The slanted-plane stereo vision.

By experimental results we show proofs of the convergence for the accuracy of such a generic PBP algorithm with finite samples. In addition to accuracy, the PBP algorithm also allows us to estimate and represent state uncertainty, although at some computational cost. In particular, an experiment with a synthetic structure from motion arrangement shows that its accuracy is comparable with the state-of-the-art Bundle Adjustment (BA) while allowing estimates of state uncertainty in the form of an approximate posterior density function.

2.1 Belief Propagation

2.1.1 Definitions and Notations

Let G be an undirected graph consisting of nodes $V = \{1, \ldots, k\}$ and edges E, and let Γ_s denote the set of neighbors of node s in G, i.e., the set of nodes t such that $\{s, t\}$ is an edge of G. In a probabilistic graphical model, each node $s \in V$ is associated with a random variable X_s taking on values in some domain, $\mathcal{X}_s$. We assume that each node s and edge $\{s, t\}$ are associated with potential functions Ψ_s and $\Psi_{s,t}$ respectively, and given these potential functions we define a probability distribution over assignments of values to nodes as

$$P(\vec{x}) = \frac{1}{Z} \left(\prod_s \Psi_s(\vec{x}_s) \right) \left(\prod_{\{s,t\} \in E} \Psi_{s,t}(\vec{x}_s, \vec{x}_t) \right) \tag{2.1}$$

Note that (2.1) corresponds to the pairwise MRF model. Here $\vec{x}$ is an assignment of values to all k variables, $\vec{x}_s$ is the value assigned to X_s by $\vec{x}$, and Z is a scalar chosen to normalize the distribution P (also called the partition function). We consider the problem of computing marginal probabilities, defined by

$$P_s(x_s) = \sum_{\vec{x} : \vec{x}_s = x_s} P(\vec{x}) \tag{2.2}$$

In the case where G is a tree and the sets $\mathcal{X}_s$ are small, the marginal probabilities can be computed efficiently by belief propagation (Pearl 1988). This is done by computing messages $m_{t \to s}$ each of which is a function on the state space of the target node, $\mathcal{X}_s$.

These messages can be defined recursively as

$$m_{t\to s}(x_s) = \sum_{x_t \in \mathcal{X}_t} \Psi_{t,s}(x_t, x_s)\Psi_t(x_t) \prod_{u \in \Gamma_t \setminus s} m_{u\to t}(x_t) \tag{2.3}$$

When G is a tree this recursion is well founded (loop-free) and the above equation uniquely determines the messages. We will use an unnormalized belief function defined as follows.

$$B_s(x_s) = \Psi_s(x_s) \prod_{t \in \Gamma_s} m_{t\to s}(x_s) \tag{2.4}$$

When G is a tree the belief function is proportional to the marginal probability P_s defined by (2.2). It is sometimes useful to define the "pre-message" $M_{t\to s}$ as follows for $x_t \in \mathcal{X}_t$.

$$M_{t\to s}(x_t) = \Psi_t(x_t) \prod_{u \in \Gamma_t \setminus s} m_{u\to t}(x_t) \tag{2.5}$$

Note that the pre-message $M_{t\to s}$ defines a weighting on the state space of the source node $\mathcal{X}_t$, while the message $m_{t\to s}$ defines a weighting on the state space of the destination, $\mathcal{X}_s$. We can then re-express (2.3)–(2.4) as

$$m_{t\to s}(x_s) = \sum_{x_t \in \mathcal{X}_t} \Psi_{t,s}(x_t, x_s)M_{t\to s}(x_t) \qquad B_t(x_t) = M_{t\to s}(x_t)m_{s\to t}(x_t)$$

Although we develop our results for tree–structured graphs, it is common to apply belief propagation to graphs with cycles as well ("loopy" belief propagation). We note connections to and differences for loopy BP in the text where appropriate.

For reasons of numerical stability, it is common to normalize each message $m_{t\to s}$ so that it has unit sum. However, such normalization of messages has no other effect on the (normalized) belief functions (2.4). Thus for conceptual simplicity in developing and

analyzing particle belief propagation we avoid any explicit normalization of the messages; such normalization can be included in the algorithms in practice.

Additionally, for reasons of computational efficiency it is common to use the alternative expression $m_{t\to s}(x_s) = \sum \Psi_{t,s}(x_t, x_s)B_t(x_t)/m_{s\to t}(x_t)$ when computing the messages. By storing and updating the belief values $B_t(x_t)$ incrementally as incoming messages are re-computed, one can significantly reduce the number of operations required. Although our development of particle belief propagation uses the update form (2.3), this alternative formulation can be applied to improve its efficiency as well.

2.1.2 Sum-Product versus Max-Product Algorithm

There are two different algorithms that can be chosen to implement BP. The sum-product algorithm computes the marginal distribution at each node of the MRF. The max-product algorithm, on the other hand, aims at computing the MAP estimate of the whole MRF. Therefore, max-product belief propagation attempt to solve the same problem as that of Graph Cuts algorithms.

2.1.3 Loopy BP and Message Normalization

Loopy belief propagation maintains a state which stores a numerical value for each message value $m_{t\to s}(x_s)$. In loopy BP one repeatedly updates one message at a time. More specifically, one selects a directed edge $t \to s$ and updates all values $m_{t\to s}(x_s)$ using equation (2.3). Loopy BP often involves a large number of message updates. As the number of updates increases message values typically diverge — usually tending toward zero but possibly toward infinity if the potentials Ψ_s and $\Psi_{t,s}$ are large. This typically results in floating point underflow or overflow. To avoid floating point errors messages can be normalized — the function $m_{t\to s}$ can be scaled by a constant so that it sums to one. We

can normalize an entire state by normalizing each message. It is important to note that normalization commutes with update. More specifically, normalizing a message, updating, and then normalizing the resulting state, is the same as updating (without normalizing) and then normalizing the resulting state. This implies that any normalized state of the system can be viewed as the result of running unnormalized updates and then normalizing the resulting state only at the end. For conceptual simplicity we avoid explicit normalization throughout this paper. But all algorithms described here can be normalized and have the property that message updates commute with normalization in the sense just mentioned.

2.1.4 Efficiency of Updates

Performing a message update using equation (2.3) would seem to involve an inner loop over the node u. This inner loop can be avoided by using the following which is equivalent to (2.3).

$$m_{t\to s}(x_s) = \sum_{x_t \in \mathcal{X}_t} \frac{\Psi_{t,s}(x_t, x_s) B_t(x_t)}{m_{s\to t}(x_t)} \tag{2.6}$$

The belief values B_t can be stored as part of the state and maintained incrementally when updates are made. This efficiency improvement is possible with the particle belief algorithm described in the next section. However, for conceptual simplicity we will consider only the inefficient update analogous to (2.3).

2.2 Markov Chain Monte Carlo (MCMC) Methods

Markov chain Monte Carlo (MCMC) is a class of algorithms for simulating direct draws from complex probability distributions of interest. MCMC has its name from the fact that it uses the previous samples to randomly generate the next samples, constructing a Markov

chain that has the desired distribution as its equilibrium distribution. The state of the chain after a large number of steps is then used as a sample from the desired distribution. The quality of the sample improves as a function of the number of steps.

2.2.1 Markov Process and Markov Chains

Let X_t denote the value of a random variable at time t, and let the state space refer to the range of possible X values. The random variable is a Markov process if the transition probabilities between different values in the state space depend only on the random variables current state, i.e.,

$$Pr(X_{t+1}|X_t, X_{t-1}, ..., X_0) = Pr(X_{t+1}|X_t) \tag{2.7}$$

Thus for a Markov random variable the only information about the past needed to predict the future is the current state of the random variable, knowledge of the values of earlier states do not change the transition probability. A Markov chain refers to a sequence of random variables $(X_0; ...; X_n)$ generated by a Markov process.

2.2.2 Monte Carlo Integration

The Monte Carlo approach was originally used as a method to compute integrals approximately by random number generation. Suppose we wish to compute a complex integral that can be represented as a product of a function $f(x)$ and a probability density function $p(x)$, the integral can then be expressed as an expectation of $f(x)$ over $p(x)$. This expectation in turn can be approximated by drawing a large number of random variables from $p(x)$:

$$\int_a^b g(x)dx = \int_a^b f(x)p(x)dx = E_{p(x)}[f(x)] \approx \frac{1}{n}\sum_{i=1}^{n} f(x_i) \tag{2.8}$$

2.2.3 Metropolis-Hasting Algorithm

The most challenging problem of applying Monte Carlo integration is how to sample from some complex probability distribution $p(x)$. Suppose our goal is to draw samples from some distribution $p(x)$ where $p(x) = \frac{1}{Z}f(x)$, where the normalizing constant Z may not be known, and very difficult to compute.

The Metropolis algorithm (Metropolis and Ulam 1949) generates a sequence of draws from this distribution by first computing

$$\alpha = \min\left(\frac{f(\theta^*)Q(\theta_{t-1}, \theta^*)}{f(\theta_{t-1})Q(\theta^*, \theta_{t-1})}, 1\right) \tag{2.9}$$

at each time step t, and then accepting a candidate point with probability α. If the candidate is rejected, the current value θ_{t-1} is retained. Here $Q(\theta_{t-1}, \theta^*)$ is the transition probability function. The original Metropolis algorithm calls for the transition probability function to be symmetric, i.e. $Q(\theta_{t-1}, \theta^*) = Q(\theta^*, \theta_{t-1})$; Hastings in (Hastings) generalized the Metropolis algorithm by using an arbitrary transition probability function.

2.2.4 Simulated Annealing

Simulated annealing was developed as an approach for locating a good approximation to the global optimum of a complex multimodal function in a large search space. This is a

good solution for cases where standard hill-climbing approaches such as gradient descent may get stuck at a local optimal peak.

The idea is that when we initially start sampling the space, we will accept a reasonable probability of a down-hill move in order to explore the entire space. As the process proceeds, we decrease the probability of such down-hill moves.

$$\alpha_{SA} = \min\left(\left(\frac{f(\theta^*)Q(\theta_{t-1},\theta^*)}{f(\theta_{t-1})Q(\theta^*,\theta_{t-1})}^{\frac{1}{T(t)}}\right), 1\right) \tag{2.10}$$

We can see that simulated annealing is very closely related to Metropolis sampling, differing only in that the probability α of a move also depends on a temperature $T(t)$, often called the cooling schedule. If T is set to 1 then equation (2.10) is exactly (2.9). Simulated annealing usually starts off with a high temperature - high transition probability, and then cool down to a very low temperature ($T \approx 0$) - low transition probability.

2.2.5 Gibbs sampling

Gibbs sampling is considered a special case of Metropolis-Hastings sampling wherein the random value is always accepted (i.e, $\alpha = 1$). The purpose of Gibbs sampling is to generate a sequence of samples from the joint probability distribution of multiple random variables. The purpose of such a sequence is to approximate the joint distribution, or to compute an integral (such as an expected value).

The key to the Gibbs sampler is that one only considers univariate conditional distributions the distribution when all of the random variables but one are assigned fixed values. Such conditional distributions are far easier to simulate than complex joint distributions and usually have simple forms. Thus, one simulates n random variables sequentially from the

n univariate conditionals rather than generating a single n-dimensional vector in a single pass using the full joint distribution.

2.3 Particle Filters

Particle filtering, also known as sequential Monte Carlo methods (SMC), are sophisticated model estimation techniques based on simulation ((Arulampalam et al. 2002; Doucet et al. 2001; Godsill and Clapp 2001; Khan et al. 2005)).

They are usually used to estimate Bayesian models and are the sequential ('on-line') analogue of Markov chain Monte Carlo (MCMC) batch methods and are often similar to importance sampling methods. Well-designed particle filters can often be much faster than MCMC. They are often an alternative to the Extended Kalman filter (EKF) or Unscented Kalman filter (UKF) ((van der Merwe et al. 2001)) with the advantage that, with sufficient samples, they approach the Bayesian optimal estimate, so they can be made more accurate than either the EKF or UKF. However, when the sample size is not sufficiently large, they might suffer from sample impoverishment. Particle filters can also be combined by using a version of the Kalman filter as a proposal distribution for the particle filter.

Particle filtering starts with a large set of random samples called particles and their weights:

$$\left\{ \left(x_t^i, w_t^i \right) : i = 1, .., P \right\}$$

$$\sum_{i=1}^{P} w_t^i = 1$$

Particle filtering then iteratively approximates the probability distribution of interest using the initial set of particles.

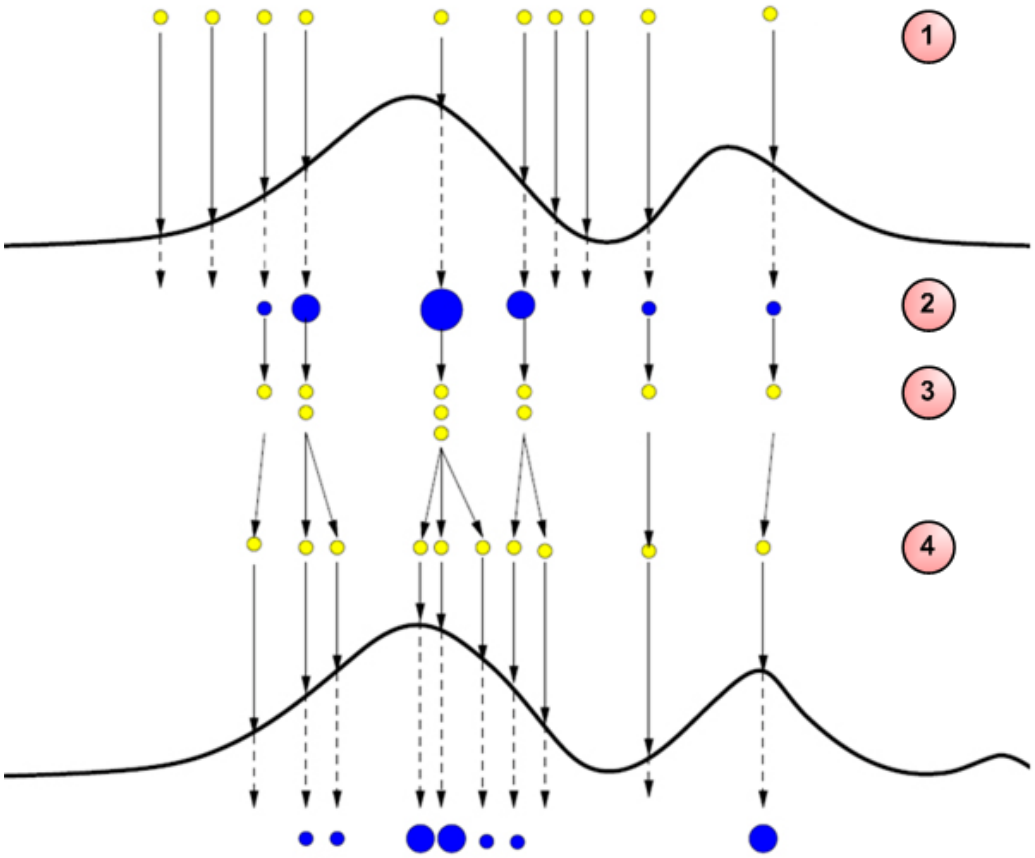


Figure 2.1: Particle filtering approximates the probability distribution of interest using a large set of random samples, named particles.

$$\int f(x_t)p(x_t|y_0,..,y_t)dx_t \approx \sum_{i=1}^{P} w_t^i f(x_t^i)$$

A single iteration of the algorithm (as illustrated in Figure 2.1)is as follows:

1. Draw samples from the proposal distribution.

2. Update the importance weights up to a normalizing constant.

3. Draw P new particles from the current particle set with probabilities proportional to their weights.

4. Set the weight of each particle to be 1/P. Update the proposal distribution using the new set of particles.

2.4 Particle-based Belief Propagation

We now consider the case where $|\mathcal{X}_s|$ is too large to enumerate in practice and define a generic particle (sample) based algorithm. This algorithm essentially corresponds to a non-iterative version of the method described in (Koller et al. 1999).

The procedure samples a set of particles $x_s^{(1)}, \ldots, x_s^{(n)}$ with $x_s^{(i)} \in \mathcal{X}_s$ at each node s of the network[1], drawn from a sampling distribution (or weighting) $W_s(x_s) > 0$ (corresponding to the proposal distribution in particle filtering). First we note that (2.3) can be written as the following importance-sampling corrected expectation.

$$m_{t\to s}(x_s) = \mathrm{E}_{x_t \sim W_t}\left[\frac{\Psi_{s,t}(x_s, x_t)\Psi_t(x_t)\prod_{u\in\Gamma_t\setminus s} m_{u\to t}(x_t)}{W_t(x_t)}\right] \tag{2.11}$$

Given a sample $x_t^{(1)}, \ldots, x_t^{(n)}$ of points drawn from W_t we can estimate $m_{t\to s}(x_s^{(i)})$ as follows where $w_s^{(i)} = W_s(x_s^{(i)})$.

$$\widehat{m}_{t\to s}^{(i)} = \frac{1}{n}\sum_{j=1}^{n} \frac{\Psi_{t,s}(x_t^{(j)}, x_s^{(i)})\Psi_t(x_t^{(j)})\prod_{u\in\Gamma_t\setminus s}\widehat{m}_{u\to t}^{(j)}}{w_t^{(j)}} \tag{2.12}$$

Equation (2.12) represents a finite sample estimate for (2.11). Alternatively, (2.12) defines a belief propagation algorithm where messages are defined on particles rather than the entire

1. It is also possible to sample a set of particles $\{x_{st}^{(i)}\}$ for each *message* $m_{s\to t}$ in the network from potentially different distributions $W_{st}(x_s)$, for which our analysis remains essentially unchanged. However, for notational simplicity and to be able to apply the more computationally efficient message expression described in Section 2.1, we use a single distribution and sample set for each node.

set $\mathcal{X}_s$. As in classical belief propagation, for tree structured graphs and fixed particle locations there is a unique set of messages satisfying (2.12). Equation (2.12) can also be applied for loopy graphs (again observing that message normalization can be conceptually ignored). In this simple version, the sample values $x_s^{(i)}$ and weights $w_s^{(i)}$ remain unchanged as messages are updated.

We now show that equation (2.12) is consistent—it agrees with (2.3) in the limit as $n \to \infty$. For any finite sample, define the particle domain $\widehat{\mathcal{X}}_s$ and the count $c_i(x)$ for $x \in \widehat{\mathcal{X}}_s$ as

$$\widehat{\mathcal{X}}_s = \{x_s \in \mathcal{X}_s : \exists k \; x_s^{(i)} = x_s\} \qquad\qquad c_s(x_s) = |\{k : \; x_s^{(i)} = x_s\}|$$

Equation (2.12) has the property that if $x_s^{(i)} = x_s^{(i')}$ then $m_{t\to s}^{(i)} = m_{t\to s}^{(i')}$; thus we can rewrite (2.12) as

$$\widehat{m}_{t\to s}(x_s) = \frac{1}{n} \sum_{x_t \in \widehat{\mathcal{X}}_t} \frac{c_t(x_t)}{W_t(x_t)} \Psi_{t,s}(x_t, x_s) \Psi_t(x_t) \prod_{u \in \Gamma_t \setminus s} \widehat{m}_{u\to t}(x_t) \quad x_s \in \widehat{\mathcal{X}}_s \qquad (2.13)$$

Since we have assumed $W_s(x_s) > 0$, in the limit of an infinite sample $\widehat{\mathcal{X}}_t$ becomes all of $\mathcal{X}_t$ and the ratio $(c_t(x_t)/n)$ converges to $W_t(x_t)$. So for sufficiently large samples (2.13) approaches the true message (2.3). Fundamentally, we are interested in particle-based approximations to belief propagation for their finite–sample behavior, i.e., we hope that a relatively small collection of samples will give rise to an accurate estimate of the beliefs - the true marginal $P_t(x_t)$. At any stage of BP we can use our current marginal estimate to construct a new sampling distribution for node t and draw a new set of particles $\{x_t^{(i)}\}$. This leads to an iterative algorithm which continues to improve its estimates as the sampling distributions become more accurately targeted. Unfortunately, such iterative resampling processes often require more work to analyze; see e.g. (Neal et al. 2003).

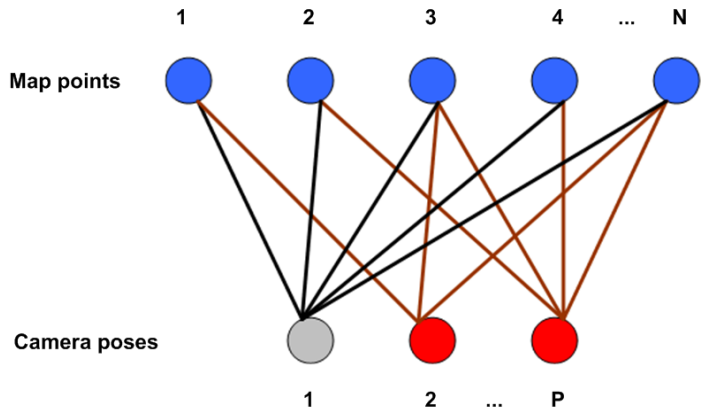


Figure 2.2: The representation of the Structure from Motion problem as a bipartite graphical model.

In (Koller et al. 1999), the sampling distributions were constructed using a density estimation step (fitting mixtures of Gaussians). However, the fact that the belief estimate $\widetilde{M}_t(x_t)$ can be computed at any value of x_t allows us to use another approach, which has also been applied to particle filters (Godsill and Clapp 2001; Khan et al. 2005) with success. By running a short MCMC simulation such as the Metropolis-Hastings algorithm, one can attempt to draw samples directly from the weighting $\widetilde{M}_t$. This manages to avoid any distributional assumptions inherent in density estimation methods, but has the disadvantage that it can be difficult to assess convergence during MCMC sampling.

2.5 Applications of PBP to vision problems

In this section we describe the applications of the PBP algorithm to several Vision problems related to scene geometry recovery.

2.5.1 Particle Belief Propagation for Structure from Motion

In structure from motion(SfM), given a set of 2D images of the same scene, the objective is to simultaneously recover both the camera trajectories and the 3D structure of the scene.

For this problem, it is commonly accepted that Bundle Adjustment (BA) (Triggs et al. 1999) is the Gold standard. There has been related work on this problem (Hartley and Zisserman 2004; Kaucic et al. 2001; Snavely et al. 2006), mostly using geometric based methods. All of these works demonstrated that their result using BA as the last step is much better than without BA. With each camera pose and each map object 3D location being represented as a finite set of particles, we show that PBP can successfully estimate their true states by estimating their posterior distributions over the state space given the image observations. The algorithm was tested on both real and synthetic data. An experiment on synthetic data allows us to compare the performance of our method with BA.

First, we represent the problem of SfM in the context of PBP. Each observed image is converted to a sparse set of image points. These points can be high-level image features, obtained by a feature detector (such as corner detector or SIFT detector (Lowe 2004)). The correspondences of image points between images are automatically obtained using a feature matching method. This also defines the correspondences between the image points and the map points. The scene is represented as a sparse map of 3D points. Each camera pose is also a 3D point, combined with 3 angles of rotation, which define the rotation of the camera about 3 axes. Our method is based on the assumption that given a set of image observations and the estimate of all map points, there exists a probability distribution for each camera pose over its state space. The true state of map points and camera poses are hidden variables that we want to estimate. Let $P_i \in \Re^6$ denote the random variable for the i^{th} camera pose. Let $Y_j \in \Re^3$ denote the random variable for the j^{th} map point. The observed data are image points and their correspondences. We denote $x_{ij} \in \Re^2$ the image point variable associated with map point Y_j as seen from camera P_i.

Our graphical model $G(V, E)$ consists of two types of nodes. Each camera node i is associated with a camera variable P_i, each map node j is associated with a map variable

Y_j. For simplicity of notation we name each node after its variable. For each camera and a map point it observes, we add an edge connecting two respective nodes into the graph. No edge connects any two camera nodes, or any two map nodes. If each map point is seen by all cameras, we have a complete bipartite graph. For each edge in the graph, there is a binary potential function associated to it, denoted $\Psi_{i,j}(P_i, Y_j)$. More specifically, we have:

$$\Psi_{i,j}(P_i, Y_j) = e^{\frac{-\left\|Q(P_i,Y_j)-x_{ij}\right\|^2}{2*\sigma^2}} \tag{2.14}$$

where $Q(P_i, Y_j)$ is the reprojection function that takes a camera pose and a map point and returns the reprojected image point, x_{ij} is the image observation of map point y_j as seen from camera p_i, σ^2 is the variance of the Gaussian image noise. This is equivalent to assuming a normal distribution of the true observation around the given observation. This allows us to represent the uncertainty of the observations.

At this point the message function and belief function on G are well defined using (2.3) and (2.4). However, the state space of each hidden variable in G is too large to do inference with BP. In order to use PBP, we discretize the state space into a relatively small number of states, each of which is represented by a particle. These states can be sampled from a normal distribution with some initial mean and variance. Now we have at each camera node P_i, a set of M particles: $P_i^{(1)}$, ..., $P_i^{(M)}$, at each map node Y_j, a set of N particles: $y_j^{(1)}$, ..., $y_j^{(N)}$. If we assume no unary potential at each node, we can define the message going from camera node i to map node j in G by (2.12) as follows:

$$\widehat{m}_{i\to j}^{(k)} = \frac{1}{M}\sum_{h=1}^{M} w_t^{(h)} \Psi_{i,j}(P_i^{(h)}, Y_j^{(k)}) \prod_{u\in\Gamma_i\setminus j} \widehat{m}_{u\to i}^{(h)} \tag{2.15}$$

At the beginning, all particles in a node are assigned uniform weights and no message has been computed, equation (2.15) becomes:

$$\widehat{m}_{i\to j}^{(k)} = \frac{1}{M}\sum_{h=1}^{M}\Psi_{i,j}(P_i^{(h)}, Y_j^{(k)}) \tag{2.16}$$

This can be interpreted as the marginal probability $\sum_{h=1}^{M} P(Y_j = Y_j^{(k)}|x_{ij}, P_i = P_i^{(h)})$. It follows that the belief of a map node becomes the marginal probability over all assignments of its neighboring camera nodes, conditioned on relevant observations:

$$\widehat{B}_j(Y_j^{(k)}) = \prod_{i\in\Gamma_j}\sum_{h=1}^{M} P(Y_j = Y_j^{(k)}|x, P_i = P_i^{(h)}) \quad = \sum_{\vec{p}} P(Y_j = Y_j^{(k)}|x, \vec{P}) \tag{2.17}$$

where $\vec{P}$ is an assignment of values to all neighboring pose nodes. The message from a map node to a pose node and the belief of a pose node can be interpreted similarly. As BP proceeds, the information from one node is sent to all other nodes in the graph. This allows nodes of the same type to communicate with each other. Then (2.17) will become exactly (2.2), which is what we want to compute. As shown in (Ihler and McAllester 2009), the posterior marginal estimated by PBP is in most cases guaranteed to converge to the true posterior marginal distribution estimated by BP. In addition to PBP, the samples at each node are iteratively updated by Gibbs sampling from the estimated posterior marginal distribution at that node. Gibbs sampling allows particles to freely explore the state space and thus compensates the inadequacy of representing a large state space by a small number of samples. After a sufficient number of alternative PBP and Gibbs sampling iterations, the estimated posterior at each node converges to the true posterior distribution, which gives us

the answer for the final state of each camera pose and map point.

The objective of this experiment is to compare PBP with the well known method BA in terms of reconstruction accuracy, by measuring the deviation of their reconstruction estimates from the ground truth. The model for bundle adjustment is a pairwise graphical model over poses p_i and object positions y_j given observed locations x_{ij} in the images:

$$P(\{p_i\},\{y_j\}) = \prod \psi_{i,j}(p_i, y_j) \qquad \Psi_{i,j}(p_i, y_j) = e^{\frac{-\left\|Q(p_i,y_j)-x_{ij}\right\|^2}{2*\sigma^2}}$$

Bundle adjustment then searches for a local maximum (i.e., a mode) of the posterior distribution over the $\{p_i, y_j\}$. However, if the intent is to minimize the expected squared error loss, we should prefer to estimate the mean of the posterior distribution rather than its mode. These can be very different in problems where the posterior distribution is very skewed or multi-modal; in such cases, it may be advantageous to estimate the posterior distribution and its mean using methods such as belief propagation. Additionally, an explicit representation of uncertainty can be used to assess confidence in the estimate.

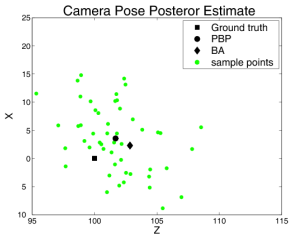
We show a comparison between estimates given by BA[2] and PBP on synthetic structure from motion data in Figure 2.3. For the data, we generate a fixed set of map points and a few camera poses. Each point's observations are obtained by projecting the point on each camera's image planes and adding Gaussian noise. We assume that the camera calibration parameters and the image correspondences are known, and initialize the estimates for both algorithms to ground truth, then run each to convergence and compare the resulting error.

Figure 2.3(a) shows the estimated mean and standard deviation of reconstruction errors of BA and PBP from 200 different runs (units are synthetic coordinates). This shows that,

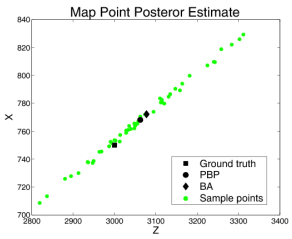
2. We use the *sba* package in (Lourakis and Argyros 2004).

Method	Error
BA (Map)	185.81 ± 66.43
PBP (Map)	194.00 ± 63.02
BA (Pose)	11.88 ± 4.40
PBP (Pose)	11.56 ± 4.39

(a)



(b)



(c)

Figure 2.3: Comparing bundle adjustment to PBP in structure from motion. (a) Estimated mean and standard deviation of reconstruction errors for camera pose and map positions; (b) example posterior for one camera pose and (c) for one map point using PBP.

at the same level of image noise and from the same initial conditions, PBP produces essentially the same accuracy as BA for both camera and map points. However, we expect that in less idealized cases (including, for example, incorrect feature associations or outliers measurements), PBP may perform much better. In addition, PBP provides an estimate of state uncertainty (although at some computational cost). Figure 2.3(b)–(c) shows the estimated posterior distributions given by PBP for a camera pose and map point, respectively, along with the mean found via PBP (circle), mode found via BA (diamond), and true position (square).

2.5.2 PBP for Dense Stereo Vision with Uncalibrated Cameras

Classical dense two-frame Stereo algorithms compute a dense disparity map for each pixel given a pair of images under known camera constraints (i.e, the orientation of the 2 cameras and the distance between them are known) (Sun et al. 2003; Felzenszwalb and Huttenlocher 2006; Scharstein et al. 2001). In this experiment, given a pair of stereo images with unknown camera constraints, we use PBP to simultaneously compute the dense depth map for each pixel, and the configuration of the second camera relative to the first.

The formulation of the graphical model in this case is quite similar to the previous problem. However there are only 2 camera nodes, and the number of map nodes equals the

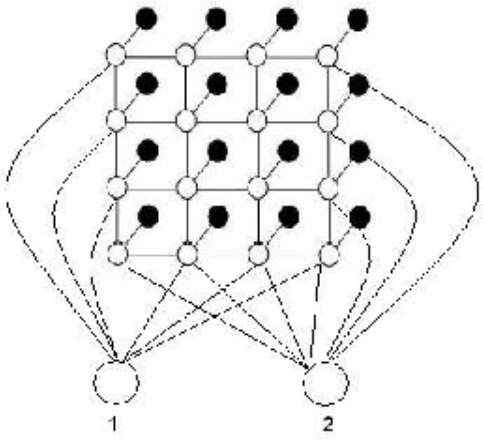
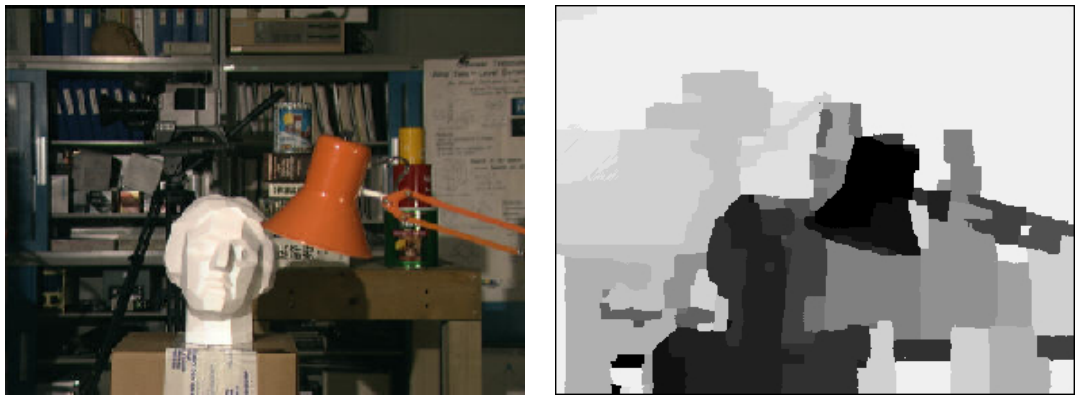


Figure 2.4: The representation of the Dense Stereo Vision problem as a graphical model.



(a) (b)

Figure 2.5: (a) Ground truth; (b) estimated depth map after a few iterations of PBP.

number of pixels in the first image. An edge is added between every pixel and every camera. (see Figure 2.4) Figure 4.2 shows the estimated depth map after a few iterations. We do not show evaluation results of our output depth map in comparison with ground truth and other stereo algorithm (using for example the Middlebury stereo benchmark (Scharstein et al. 2001)), because such a comparison will not be very meaningful, as the labels we use are the true 3D depths of each pixel instead of their disparities. However, in Figure 2.6, we plot the estimated posterior distribution of the second camera pose at each iteration, thus show that the distribution gradually approaches the true state over time.

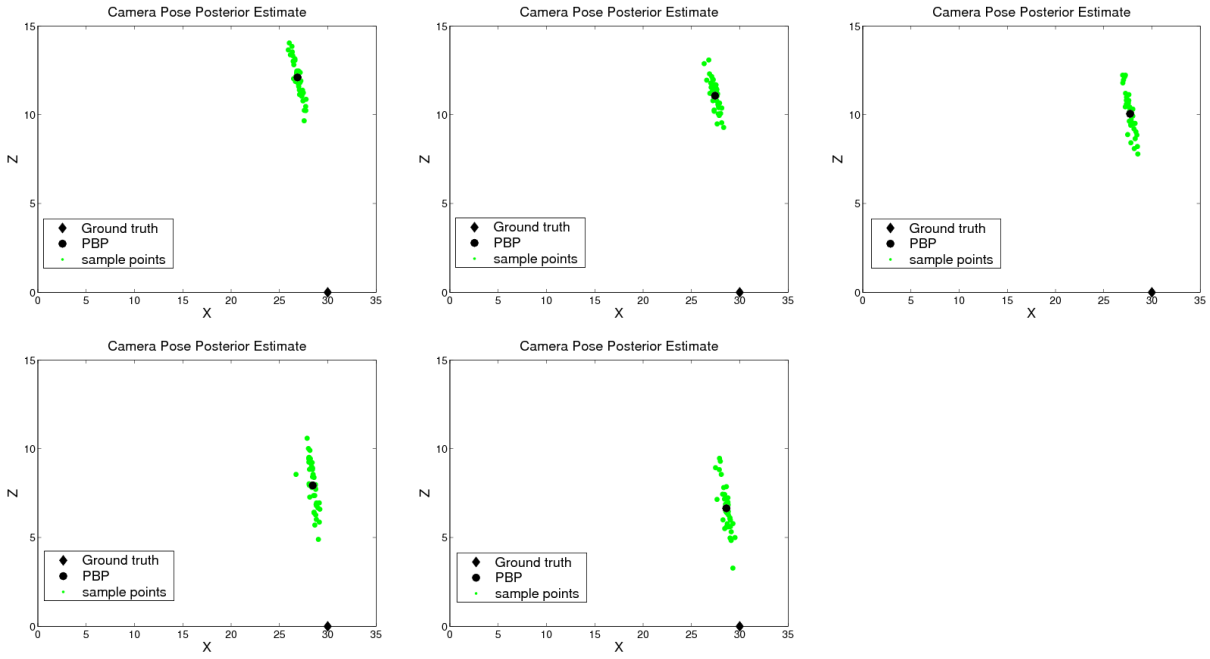


Figure 2.6: Estimated posterior distribution of the camera pose over time

2.6 PBP for Slanted-Plane Stereo Estimation

For this task, given a pair of images (X, Y), and a given segmentation of X, and a given setting of the model parameters, the inference problem is to find an assignment Z of plane parameters to superpixels so as to minimize the total energy in (3.2). The energy defines a Markov random field. More specifically, the match energy defines the unary potential on each superpixel independently and the smoothness energy defines the binary potential on pairs of adjacent superpixels.

The complete inference algorithm is described in Section 4.1.

CHAPTER 3

SLANTED PLANE MODEL WITH SURFACE ORIENTATION CUES FOR STEREO DEPTH ESTIMATION

In this chapter, we present our slanted-plane stereo vision model with monocular cues for surface orientation estimation. We also explain our idea of using HOG features as surface orientation cues, and present an analytical justification for this idea.

3.1 The Slanted Plane Stereo Model

As previously mentioned in our introduction, the two-view stereo vision problem is commonly treated as a pixel-labeling problem, where labels are disparity values, usually quantized to be integers. The disparity is equivalent to the horizontal displacement of the pixel between the left and the right image, or the inverse depth of that pixel. Specifically, the corresponding pixel in the right image of a pixel $p = (x_p, y_p)$ in the left image is given by $p' = p - d(p) = (x_p - d(p), y_p)$. The labeling is then resolved by using an energy minimization technique. The standard form of the global energy function is formulated in equation (3.1).

$$E(d) = E_M(d) + \lambda E_S(d) \tag{3.1}$$

where E_M is the match energy measuring how well the disparity assignment d agrees with the input image pair, E_S is the smoothness energy enforcing the smoothness assump-

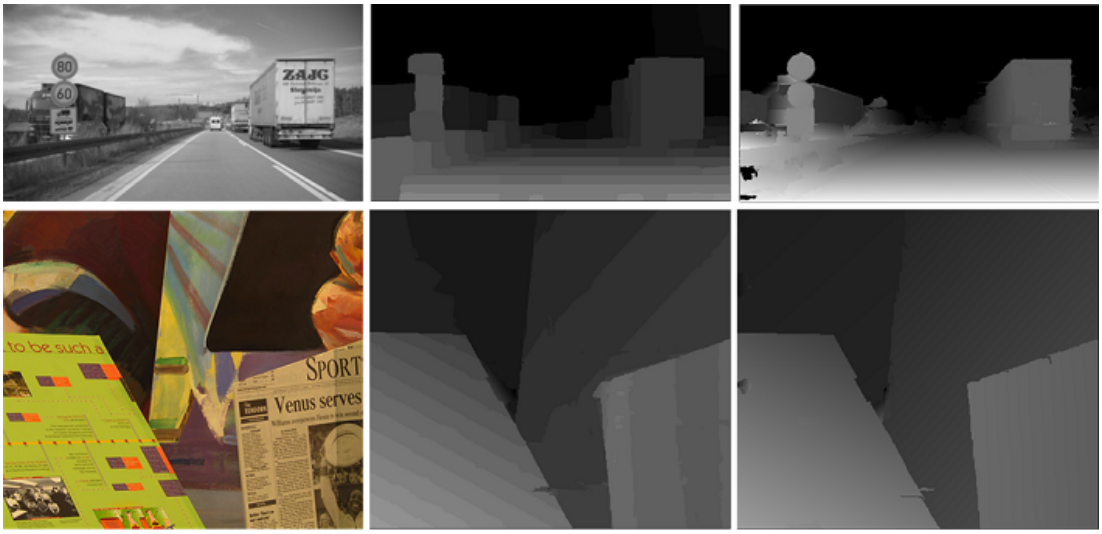


Figure 3.1: The scene geometry is much more accurately captured by the slanted-plane model than by the pixel-based stereo model.

tion. Stereo methods aiming at finding d that optimizes the global energy function are called global methods. The best-performing algorithms in the Middlebury benchmark (Scharstein et al. 2001) are all global methods.

The quantization of disparity values $d(p)$ helps reduce considerably the state space for searching. All techniques discussed in (Scharstein and Szeliski 2002) follow a common framework - producing the quantized disparity map as output. However, this output has the effect that the scene seems to be split into a series of fronto-parallel surfaces, as illustrated in Figure 3.1. In other words, piecewise continuity of the scene is replaced by piecewise constancy. This is a poor explanation of the scene geometry, especially when the scene contains slanted surfaces.

The use of slanted-plane model for stereo vision was first proposed by BirchField and Tomasi in (Birchfield and Tomasi 1999) as a special case of the affine model. Since then the model has been widely applied in many other stereo algorithms, including most of

state-of-the-art algorithms in the current Middlebury benchmark. (Klaus et al. 2006; Wang and Zheng 2008; Q. Yang and Nistr 2008). The model assumes that the scene structure is locally planar - the scene is composed of a set of planar regions. This is a reasonable assumption, based on the fact that most surfaces found in a natural environment can be approximated by a plane or a set of planes. These algorithms start by segmenting the image into homogeneous regions called superpixels, using an image segmentation method. The superpixels found by the segmentation method are coherent groupings of pixels having similar properties (e.g: color, texture). Therefore it is just natural to assume that each superpixel corresponds to a disparity plane. Next, the goal of the stereo algorithm is to infer the location and orientation (or surface normal) of each of these planes - which are fully defined by the 3 plane parameters. After that, the disparity of each pixel can be directly computed from the disparity map equation, with sub-pixel precision.

Our stereo vision model is a slanted plane model involving monocular shape from texture cues. The introduction of an additional monocular term into the slanted-plane stereo model is a novel idea that has not been investigated. Another difference of our work is in the inference algorithm. Other related work that also employ the plane-based stereo model use different approach in the inference step. One of the best methods in the Middlebury benchmark is (Klaus et al. 2006). The authors used a local matching step to extract reliable correspondences, followed by robust plane fitting to fit a disparity plane into each superpixel separately. The process were repeated for iterative plane refinement. For plane fitting, instead of trying to estimate all plane parameters jointly, the authors proposed a decomposition method to solve each parameter, again separately. The estimated planes are fixed after this step. Other slanted-plane stereo algorithms as in (Wang and Zheng 2008; Q. Yang and Nistr 2008) also follow a very similar approach for plane estimation.

Our stereo inference algorithm also infers a disparity plane for each superpixel. How-

ever we use the Particle BP inference algorithm to simultaneously find the optimal joint assignment of all plane parameters to all superpixels. The optimal assignment is found by minimizing a high-dimensional global energy function. With this PBP inference algorithm, each particle, which represents a proposal disparity plane for a specific superpixel, gets updated over time, instead of being fixed after the plane fitting step, as in previously mentioned methods.

The energy function involves three terms — a correspondence energy measuring the degree to which the left and right images agree under the induced disparity map, a smoothness energy measuring the smoothness of the induced depth map, and a texture energy measuring the degree to which the surface orientation at each point agrees with a certain (monocular) texture based surface orientation cue. Specifically, the global energy in equation (3.1) is replaced by the following:

$$E(Z) = E_M(Z) + E_S(Z) + E_T(Z) \tag{3.2}$$

where Z is the assignment of a disparity plane to each superpixel in the left image. We denote X to be the left image, Y to be the right image. For each superpixel i of X we have that Z specifies three plane parameters A_i B_i, and C_i. Given an assignment Z of plane parameters to superpixels we define the disparity $d(p)$ for any pixel p as follows (where $i(p)$ is the superpixel containing p and x_p and y_p are the image coordinates of p).

$$d(p) = A_{i(p)}x_p + B_{i(p)}y_p + C_{i(p)} \tag{3.3}$$

So by equation (3.3) we have that Z assigns a real valued disparity to each pixel.

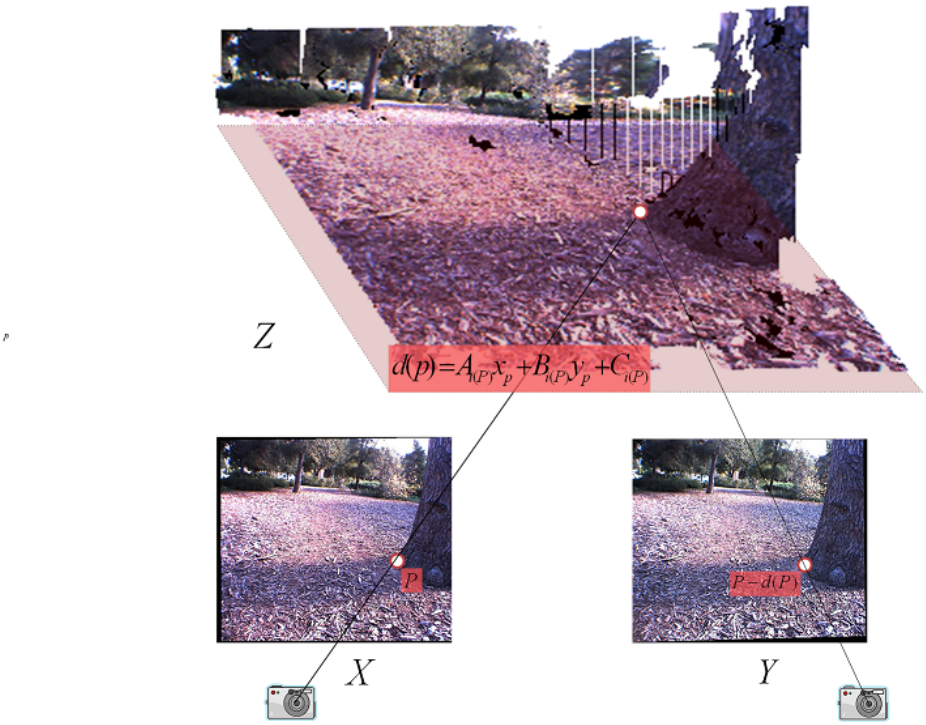


Figure 3.2: The assigned disparity plane defines a disparity value for any pixel in the superpixel.

To define the smoothness energy we write $(p, q) \in B_{i,j}$ if p is a pixel in superpixel i, q is a pixel in superpixel j, and p and q are adjacent pixels (p is directly above, below, left or right of q). The smoothness energy is defined as follows (where τ_S and λ_S are parameters of the energy, i ranges over all superpixels, $N(i)$ is the set of superpixels adjacent to i, and z is the overall assignment of three dimensional vectors of plane parameters (A_i, B_i, C_i) for all superpixels i):

$$E_S(Z) = \sum_{i,j \in N(i)} \min \left(\tau_S, \sum_{(p,q) \in B_{i,j}} \lambda_S |d(p) - d(q)| \right) \tag{3.4}$$

Intuitively the minimization with τ_S corresponds to interpreting the entire boundary between i and j as either an occlusion boundary or as a joining of two planes on the same

object.

Next we consider the match energy. We write $p - d(p)$ for the pixel in Y that corresponds to the pixel p in image X under the disparity $d(p)$. For color images we construct a nine dimensional feature vector $\Phi^X(p)$ and $\Phi^Y(p)$ for the pixel p in the images X and Y respectively. The vector $\Phi^X(p)$ consists of three (bias gain corrected) color values plus a six dimensional color gradient vector and similarly for Φ^Y_p. We write $\Phi^X_k(p)$ for the kth component of the vector $\Phi^X(p)$. The match energy is defined as follows where λ_k are nine scalar parameters of the match energy.

$$E_M(Z) = \sum_{p \in X} \sum_k \lambda_k (\Phi^x_k(p) - \Phi^y_k(p + d(p)) \,)^2 \tag{3.5}$$

Finally we consider the texture energy. At each pixel p we also compute a HOG vector $H(p)$ which is a 24 dimensional feature vector consisting of three 8 dimensional normalized edge orientation histograms — an 8 dimensional orientation histogram is computed at three different scales. The texture energy is defined as follows where $i(p)$ is the superpixel containing pixel p and where the scalars τ_T, λ_A, λ_B, and the vectors β_A and β_B are parameters of the energy. The form of this energy is justified in Chapter 3.2.

$$E_T = \sum_p \min \left(\begin{array}{rcl} \tau_T, & & \lambda_A \, \left(d(p)(\beta_A \cdot H(p) \,) - A_{i(p)} \right)^2 \\ & + & \lambda_B \, \left(d(p)(\beta_B \cdot H(p) \,) - B_{i(p)} \right)^2 \end{array} \right) \tag{3.6}$$

These energy terms can also be interpreted in a probabilistic way as follows. The combination of the smoothness energy term and the texture energy term can be interpreted as an energy $E_Z(X, Z, \beta_Z)$, which determines the probability $P(Z|X, \beta_Z)$. The match energy term can be interpreted as an energy $E_Y(X, Y, Z, \beta_Y)$, which determines $P(Y|X, Z, \beta_Y)$. The general conditional probability formulation of our model will be discussed in details

in Chapter 4.

3.2 HOG as Surface Orientation Cues

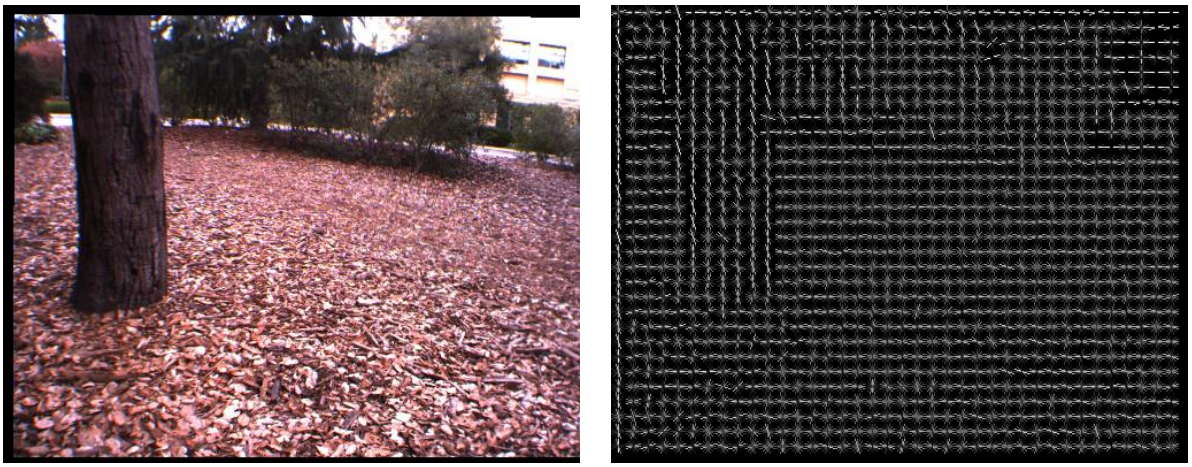


Figure 3.3: HOG features for an example image.

3.3 Relationship between HOG and Surface Orientation

In this chapter we discuss the possibility of using a specific type of monocular cues - the HOG feature (Histogram of Oriented Gradients). This feature has recently been extensively applied in computer vision and image processing for the purpose of object detection ((Dalal and Triggs 2005b; Felzenszwalb et al. 2008)). The technique counts occurrences of gradient orientation in localized portions of an image. This method is similar to that of edge orientation histograms, scale-invariant feature transform descriptors (or SIFT (Lowe 2004)), and shape contexts (S. Belongie and Puzicha 2002), but differs in that it computes on a dense grid of uniformly spaced cells and uses overlapping local contrast normalization for improved performance.

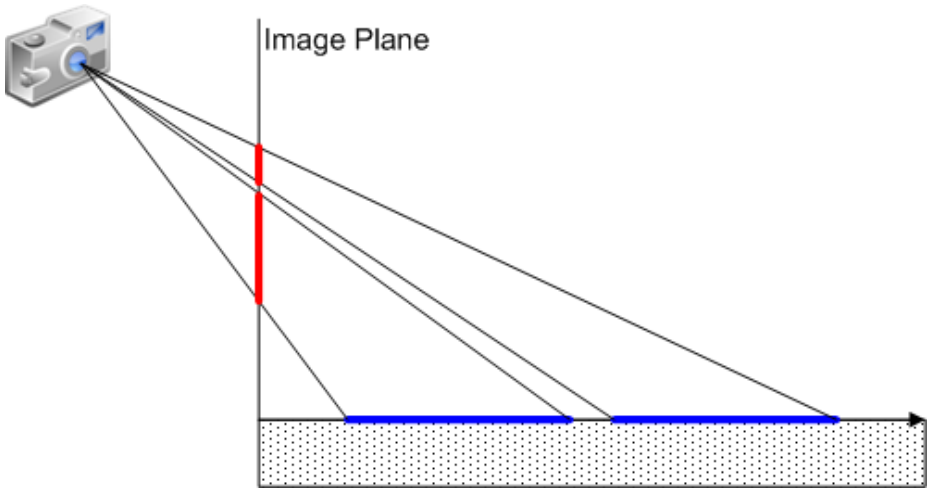


Figure 3.4: As a surface is tilted away from the camera the edges in the direction of the tilt become foreshortened while the edges orthogonal to the tilt are not.

First we observe that HOG, which describe the edge distribution, can be related to the orientation of surfaces. We formulate the statistical relationship between HOG features and surface normal. Based on this derivation, we then argue that HOG features can be used as surface orientation cues, and thus can be incorporated in our existing slanted-plane stereo model (as described in Section 3.1).

Here we aim at describing a justification for the idea of using HOG features as surface orientation cues, as mentioned in Chapter 3. The basic intuition behind HOG as an orientation cue is that as a surface is tilted away from the camera the edges in the direction of the tilt become foreshortened while the edges orthogonal to the tilt are not. This changes the edge orientation distribution and therefore the edge orientation distribution can be used as a cue for surface orientation. This effect is shown in figure 3.5 where the average HOG feature is shown for various regions of tree trunk, forest floor, grass lawn, and patio tile. The cylindrical shape of the tree trunk is clearly indicated by the warping of the HOG feature as a function of position on the trunk.

We consider the case of a surface with an isotropic edge distribution — the amount of

edge at any two edge orientations is equal. We do not expect this assumption to hold in any particular small surface patch, but we do expect a statistical relationship between HOG features and orientation. The assumption of isotropic surface textures provides a departure point for analysis. We are interested in the expected HOG feature for the edges projected onto the image plane when the surface is tilted relative to the image plane.

We consider edges to be line segments on the image plane. Each edge has a length r and an orientation Θ. The HOG feature is a histogram of edge orientations. We will normalize the HOG histogram so that it is a probability distribution on orientations. If P is a probability distribution (or probability density) on edges then the HOG feature $H(\Theta)$ is a probability distribution (or density) on Θ. More specifically, H is a marginal of P onto Θ in which edges are weighed by length.

$$H(\Theta) \quad = \quad \frac{1}{\mathrm{E}\left[r\right]} \; P(\Theta)\mathrm{E}\left[r|\Theta\right] \tag{3.7}$$

We consider a surface patch imaged by a perspective camera. A perspective camera induces the following map from three dimensional coordinates to image plane coordinates.

$$x' = (fx)/z \quad y' = (fy)/z \tag{3.8}$$

We assume a coordinate system on the surface patch such that each point on the surface patch has coordinates x_s, y_s. The image plane and surface coordinates can be selected so that we have the following map from surface coordinates to three dimensional coordinates where Ψ is the angle between surface normal and the image plane normal.

$$x = x_s \quad y = y_s \cos\Psi \quad z = z_0 + y_s \sin\Psi \tag{3.9}$$

Combining (3.8) and (3.9), and differentiating with respect to x_s and y_s gives the following with $z = z_0 + y_s \sin\Psi$.

$$\begin{aligned} dx' &= \left(\frac{f}{z}\right) dx_s \quad - \left(\frac{fx\sin\Psi}{z^2}\right) dy_s \\ dy' &= \left(\frac{f\cos\Psi}{z}\right) dy_s \quad - \left(\frac{fy\cos\Psi\sin\Psi}{z^2}\right) dy_s \end{aligned} \tag{3.10}$$

We first consider the region near the center of the image. At the center of the image we have $x = y = 0$ and (3.10) becomes the following.

$$dx' = \left(\frac{f}{z}\right) dx_s \quad dy' = \left(\frac{f\cos\Psi}{z}\right) dy_s \tag{3.11}$$

We now have the following map from the length R and orientation Γ on the surface to length r and orientation Θ on the image plane.

$$\Theta = \tan^{-1}\left(\sin\Gamma\cos\Psi,\ \cos\Gamma\right)$$

$$r = \frac{fR}{z}\sqrt{\sin^2\Gamma\cos^2\Psi + \cos^2\Gamma}$$

For $cos\Psi \neq 0$ we have a bijection between Γ and Θ. Furthermore, for fixed Θ and Γ we have a linear relationship between r and R. This gives the following.

$$H(\Theta) \propto \quad Q(\Gamma(\Theta)\,)\frac{d\Gamma}{d\Theta}\mathrm{E}\left[R|\Gamma(\Theta)\right]\frac{dr}{dR}$$

We assume that Q is isotropic, i.e., that $Q(\Gamma)\mathrm{E}\left[R|\Gamma\right]$ is a constant independent of Γ. We

then have the following.

$$H(\Theta) \propto \frac{d\Gamma}{d\Theta}\sqrt{\cos^2\Psi\sin^2\Gamma+\cos^2\Gamma}$$

$$\frac{d\Gamma}{d\Theta} = \frac{1+\tan^2\Theta}{\cos\Psi\left(1+\frac{\tan^2\Theta}{\cos^2\Psi}\right)}$$

We have no simple formula for $H(\Theta)$. However, we can see that $H(\Theta)$ is a continuous function of Θ satisfying the following.

$$H(0) = H(\pi) = \tfrac{1}{Z}\left(\tfrac{1}{\cos\Psi}\right)$$

$$H(\pi/2) = H(3\pi/2) = \tfrac{1}{Z}\cos^2\Psi$$

So we get the following where $H_{\min}$ is the minimum of $H(\Theta)$ and $H_{\max}$ is the maximum of $H(\Theta)$.

$$H_{\min}/H_{\max} = \cos^3\Psi \tag{3.12}$$

To justify the form of the orientation energy (3.6) we first note that in the same coordinate system as (3.9) the equation for the surface plane can be written as follows.

$$z = z_0 + y\tan\Psi \tag{3.13}$$

If we let b be the distance between the foci of the two cameras we have that the disparity d

equals bf/z. Multiplying (3.13) by $bf/(zz_0)$ gives the following.

$$\frac{bf}{z_0} = \frac{bf}{z} + \left(\frac{b\tan\Psi}{z_0}\right)\left(\frac{fy}{z}\right) \tag{3.14}$$

$$d_0 = d + \left(\frac{d_0\tan\Psi}{f}\right)y' \tag{3.15}$$

$$d = d_0 - \left(\frac{d_0\tan\Psi}{f}\right)y' \tag{3.16}$$

$$B = -\frac{d_0\tan\Psi}{f} \tag{3.17}$$

For the pixel p at the center of the image we have $d(p) = d_0$. We handle pixels outside of the center of the image by considering panning the camera to bring the desired point to the center and approximating panning the camera by translating the image. This gives the following general relation between the disparity plane parameter and the angle Ψ between the ray form the camera and the surface normal.

$$B = -\frac{d(p)\tan\Psi}{f} \tag{3.18}$$

In the orientation energy (3.6) we interpret $\beta_B \cdot H(p)$ as a predictor of $-(\tan\Psi)/f$ and we multiply by $d(p)$ to get a predictor of B. We do not currently exploit (3.12). Doing so should result in a more refined texture cue.

3.4 HOG Computation and Representation

In order to use HOG features as a probability distribution on orientations, we want to compute HOG as a normalized histogram of edge orientations through the whole image.

Our HOG descriptor at each pixel is a 24-dimensional vector which describes the edge orientation distribution over the local image region around that pixel. (We compute a 8-dimensional HOG feature at each image scale, for 3 different scales to get a 24-dimensional feature vector at each pixel). The HOG computation includes the following 3 main steps:

- **Gradient Computation:**

  The first step of calculation is the computation of the gradient values for all pixels in the left image. This is to simply apply the 1-D centered, point discrete derivative mask in one of or both the horizontal and vertical directions: $[-1, 0, 1]$ and $[-1, 0, 1]^T$.

- **Orientation Binning** The second step of calculation involves creating the cell histograms. The cells themselves can either be square or radial in shape, and the histogram channels are evenly spread over 0 to 180 degrees or 0 to 360 degrees, depending on whether the gradient is unsigned or signed. Here we use square cells, unsigned gradients in conjunction with 8 histogram channels. Each pixel within the cell casts a weighted vote for an orientation-based histogram channel based on the values found in the gradient computation. As for the vote weight, pixel contribution can either be the gradient magnitude itself, or some function of the magnitude; here we use the gradient magnitude. Specifically, we quantize the gradient orientation of pixels in a cell into 8 bins (corresponding to 8 equal ranges from 0 to 180 degrees). In each bin we sum the magnitude of gradients in that bin. Then we normalize the magnitude of the resulting 8-dimensional vector of the bin.

- **Pyramidal HOG Descriptors** We generate the HOG feature pyramid by computing HOG features of 3 different scales of the image pyramid (see Figure), corresponding to different sizes of the cell. The sizes of the cell at each scale are 8x8, 16x16 and

32x32 respectively. In this step we used the integral image trick for the purpose of computational efficiency.

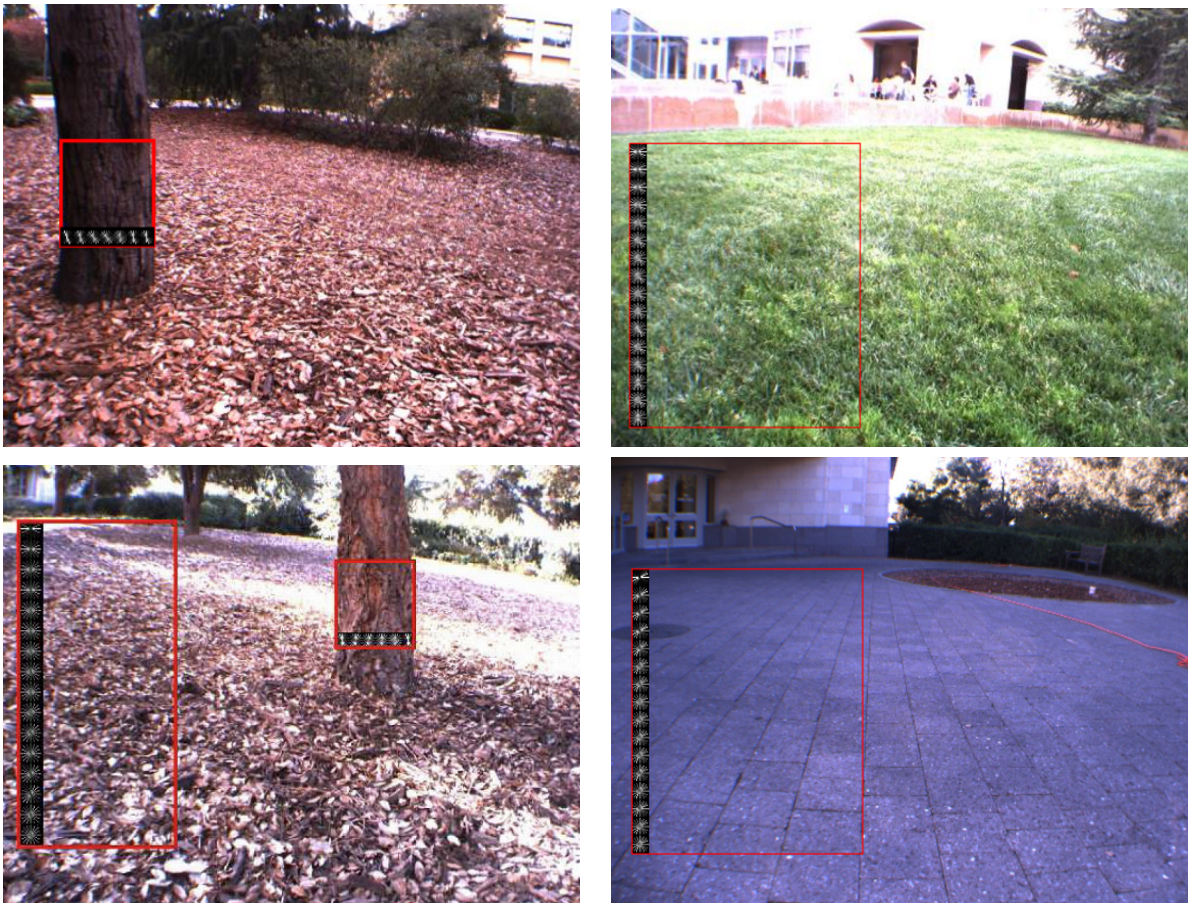


Figure 3.5: HOG features for image regions. The amount of edge as a function of angle, a HOG feature, is averaged over different vertical and horizontal regions on various images. The different surface orientations of these regions affect the HOG features. We can see the cylindrical structure of tree trunks and the fact that the ground plane becomes more tilted as the distance increases.

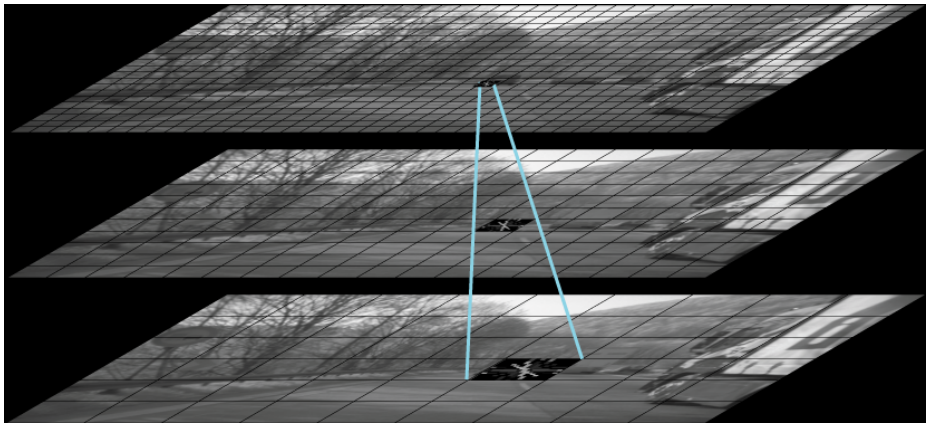


Figure 3.6: We compute HOG features at each image scale for three different scales to obtain a 24-dimensional feature vector at each pixel.

CHAPTER 4

UNSUPERVISED CRF LEARNING FOR STEREO VISION WITH MONOCULAR CUES

In this chapter, we apply the unsupervised CFR model we introduced in Section 1.4 to the problem of unsupervised learning of a highly parameterized stereo vision model involving the monocular shape from texture cues - training the model parameters from unlabeled stereo pair training data(Trinh and McAllester 2009). First, we describe in more details our PBP inference algorithm for this CRF stereo model. We then give a background review on contrastive divergence, a machine learning technique that plays a key component in our unsupervised CRF learning algorithm. Next, we present our unsupervised CRF learning of the slanted-plane stereo vision model. We conclude the chapter by reporting experimental results both for learning and inference on two different stereo datasets.

4.1 Particle Belief Propagation for Plane Estimation

Given a pair of images (X, Y), and a given segmentation of X, and a given setting of the model parameters, the inference problem is to find an assignment Z of plane parameters to superpixels so as to minimize the total energy in (3.2). The energy defines a Markov random field. More specifically, the match energy defines the unary potential on each superpixel independently and the smoothness energy defines the binary potential on pairs of adjacent superpixels.

The complete inference algorithm is illustrated in Figure 4.1. The steps of the algorithm include:

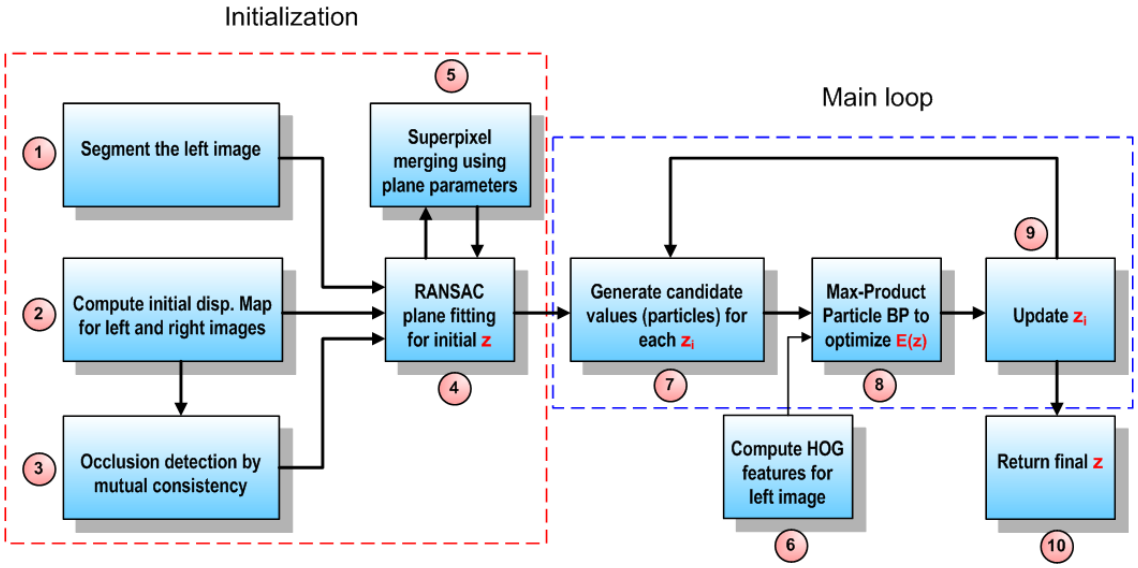


Figure 4.1: Disparity plane inference algorithm using Particle-Belief Propagation.

1. We compute a segmentation using the Felzenszwalb-Huttenlocher segmentation algorithm (Felzenszwalb and Huttenlocher 2004). Although this segmentation algorithm has a parameter governing the number of superpixels, we do not attempt to tune this parameter with our training algorithm. A more principled approach would include the segmentation itself in the energy functional and tune the segmentation parameter along with the other parameters of the model. However by holding the segmentation parameter fixed, we can treat the segmentation as part of the input data.

2. We run the Felzenszwalb-Huttenlocher's efficient loopy BP stereo algorithm ((Felzenszwalb and Huttenlocher 2006)) to compute the disparity maps for both images X and Y.

3. Mutual consistency check requires that for a particular pixel in the left image, the disparity values between the left and right disparity maps are consistent, i.e:

$$d_L(p) = d_R(p - d_L(p)) \tag{4.1}$$

The pixel is considered occluded if (4.1) does not hold, and considered non-occluded otherwise.

4. The plane fitting is performed in the disparity space, and is applied per superpixel to obtain an initial disparity plane. This is done robustly using RANSAC (Fischler and Bolles 1981) on the disparity values of the non-occluded pixels only. (similar to (Q. Yang and Nistr 2008))

5. This step is optional. In this step we implement a superpixel grouping algorithm similar to the one used in step 1. However each time we consider merging two adjacent superpixels, we also look at the difference in their disparity planes computed from step 4. Intuitively we prefer merging adjacent superpixels having very similar disparity planes. Note that if used, this step may change the initial segmentation generated by step 1. Then the output is iteratively fed back into the plane fitting in step 4.

6. At each pixel p in the left image X, we compute a HOG vector $H(p)$ which is a 24 dimensional feature vector consisting of three 8 dimensional normalized edge orientation histograms an 8 dimensional orientation histogram is computed at three different scales.

7. Let C_i be a set of candidate values for z_i derived by repeatedly adding random noise to the current value of z_i.

8. Run discrete max-product BP with the finite value set C_i for each node i.

9. Set z_i to be the best value for i found in step 8 and repeat.

10. The iteration can be stopped after a fixed number of iterations or when the energy is no longer reduced.

Note that the main loop of the algorithm, including steps 7, 8, 9, is very similar to the Particle-based Belief Propagation algorithm described in Section 2.4, which is closely related to previous work in (Ihler and McAllester 2009) and (Koller et al. 1999). This can be viewed as the max-product PBP algorithm. In Chapter 2, we describe in details two other applications of the sum-product PBP inference algorithm to two inference problems in Computer Vision.

4.2 Background Review on Contrastive Divergence

4.2.1 Introduction to Contrastive Divergence

Assume that our goal is to model the probability of the observed data x using a function of the form $f(x, \beta)$, where β is the model parameter vector. This probability can be defined as follows.

$$P(x, \beta) = \frac{1}{Z(\beta)} f(x, \beta) \tag{4.2}$$

where $Z(\beta)$, commonly known as the partition function, is defined as

$$Z(\beta) = \int f(x, \beta) dx \tag{4.3}$$

Given a training dataset $X = (x_1, x_2, ..., x_N)$, we want to learn the model parameter vector β so as to maximize the probability of the training data.

$$P(X, \beta) = \prod_{i=1}^{N} \frac{1}{Z(\beta)} f(x_i, \beta) \tag{4.4}$$

Taking the negative log of (4.4) will give us the following:

$$E(X, \beta) = \log Z(\beta) - \frac{1}{N} \sum_{i=1}^{N} \log f(x_i, \beta) \tag{4.5}$$

We denote $E(X, \beta)$, which is the negative log of $P(X, \beta)$, to be the energy function.

The partition function $Z(\beta)$ in equation (4.3) is, for many cases, a very complicated function which can be very hard to compute. In cases where this function is intractable, this leads to a situation where we are trying to minimize an energy function that we cannot evaluate.

Contrastive divergence offers a solution to this problem. Briefly speaking, contrastive divergence is a tool that allows us to estimate the gradient of the energy function, even though we cannot evaluate the energy function itself.

4.2.2 Formulation of Contrastive Divergence

As previously mentioned, contrastive divergence provides us a way to estimate the gradient of our energy function $E(X, \beta)$ with respect to the model parameters β, given the training dataset X. Taking the partial derivative of equation (4.5) with respect to β gives us the following derivation.

$$\frac{\partial E(X,\beta)}{\partial \beta} = \frac{\partial \log Z(\beta)}{\partial \beta} - \frac{1}{N}\sum_{i=1}^{N}\frac{\partial \log f(x_i,\beta)}{\partial \beta} \tag{4.6}$$

$$= \frac{\partial \log Z(\beta)}{\partial \beta} - \left\langle \frac{\partial \log f(x,\beta)}{\partial \beta} \right\rangle_X \tag{4.7}$$

where $\langle . \rangle_X$ denotes the average over the training data X. In other words, it is the expectation of . given the training data distribution X.

The first term on the right hand side of (4.6) involves the partition function $Z(\beta)$. Substituting equation (4.3) into this term, we have:

$$\frac{\partial \log Z(\beta)}{\partial \beta} = \frac{1}{Z(\beta)}\frac{\partial Z(\beta)}{\partial \beta} \tag{4.8}$$

$$= \frac{1}{Z(\beta)}\frac{\partial}{\partial \beta}\int f(x,\beta)dx \tag{4.9}$$

$$= \frac{1}{Z(\beta)}\int \frac{\partial f(x,\beta)}{\partial \beta}dx \tag{4.10}$$

$$= \frac{1}{Z(\beta)}\int f(x,\beta)\frac{\partial \log f(x,\beta)}{\partial \beta}dx \tag{4.11}$$

$$= \int P(x,\beta)\frac{\partial \log f(x,\beta)}{\partial \beta}dx \tag{4.12}$$

$$= \left\langle \frac{\partial \log f(x,\beta)}{\partial \beta} \right\rangle_{P(x,\beta)} \tag{4.13}$$

From equation (4.13), it is not hard to see that, although the integration in (4.12) is

algebraically intractable, it can be numerically approximated by drawing samples from the proposal distribution $P(x, \beta)$. The next problem is that the distribution $P(x, \beta)$ is actually unknown, as we do not know the value of the partition function. In order to resolve this, we can use a Markov Chain Monte Carlo (MCMC) sampling method to simulate draws from the proposal distribution by drawing directly from the target distribution (the training data distribution), then transforming these samples to the proposal distribution. The transformation is done by generating a Markov chain starting from the drawn sample at the target distribution to the proposal distribution. Please recall that in the Markov chain, each state x^{t+1} only depends on the previous state x^t. At each time step t, a new value x' is accepted for the next state x^{t+1} if a random number drawn from $U(0, 1)$ satisfies:

$$\alpha < \min\left\{\frac{P(x', \beta)Q(x^t, x')}{P(x^t, \beta)Q(x', x^t)}, 1\right\}$$

This involves computing the ratio $\frac{P(x',\beta)}{P(x^t,\beta)}$, which is possible since the partition function cancels out. The approximated version of equation (4.6) can be written as follows:

$$\frac{\partial E(X, \beta)}{\partial \beta} = \left\langle \frac{\partial \log f(x, \beta)}{\partial \beta} \right\rangle_{X^n} - \left\langle \frac{\partial \log f(x, \beta)}{\partial \beta} \right\rangle_{X} \tag{4.14}$$

where X^n represents the set of samples drawn from the training data after n steps of MCMC.

The use of MCMC sampling strategy, however, can be very inefficient and not practical, since it may take a very long time for the Markov chain to mix and approximately converges. Hinton in (Hinton 2002) asserts that only a few MCMC steps would be sufficient to calculate an approximate gradient. The intuition behind this is that after a few iterations the data will have moved from the target distribution (i.e. that of the training data) towards

the proposed distribution, and so give an idea in which direction the proposed distribution should move to better model the training data. Empirically, Hinton has found that even 1 step of MCMC is often sufficient for the algorithm to converge.

That said, as we use gradient descent in order to minimize our energy function, the contrastive divergence parameter update equation should look like the following:

$$\beta_{t+1} = \beta_t - \gamma \left(\left\langle \frac{\partial \log f(x,\beta)}{\partial \beta} \right\rangle_{X^1} - \left\langle \frac{\partial \log f(x,\beta)}{\partial \beta} \right\rangle_{X} \right) \tag{4.15}$$

where γ is the step size value which is allowed to change at every iteration, based on convergence time and stability.

4.3 Our Unsupervised CRF Learning Algorithm

As described earlier in Section 3.1, our stereo model involves three terms — a correspondence energy measuring the degree to which the left and right images agree under the induced disparity map, a smoothness energy measuring the smoothness of the induced depth map, and an texture energy measuring the degree to which the surface orientation at each point agrees with a certain (monocular) texture based surface orientation cue. For surface orientation cue we use histograms of oriented gradients (HOG) (Dalal and Triggs 2005a). The stereo model itself and the monocular surface cues (as part of the model) are both viewed as the objects being trained. Our energy functional includes in total 62 parameters - 10 correspondence parameters, 2 smoothness parameters, and 50 texture parameters. Among these parameters, we train 50 of them are trained using gradient descent. The gradient of the energy function w.r.t to each parameter is obtained using contrastive divergence, the method described in the previous section. The other 12 parameters, including

10 matching parameters (λ_k in (3.5)) and 2 texture parameters (β_A and β_B in (3.6)), are computed by least squares regression.

Our approach to unsupervised learning is based on maximizing conditional likelihood. In particular, we consider a general conditional probability model $P_\beta(u|x)$ over arbitrary variables x and u and defined in terms of a parameter vector β. Analogously, CRFs model (Lafferty et al. 2001) defines a conditional probability of the dependent variables u given the exogenous variables x and (importantly) does not model the distribution of the exogenous variables. In the case of stereo vision one might expect that it is easier to model the probability distribution of the right image given the left image than to model a probability distribution over images.

In a closely related earlier work by Zhang and Seitz (Zhang and Seitz 2007), the authors also used a similar CRF model for the classical pixel-based stereo algorithm. Although the training was also based on maximizing conditional likelihood, they worked with a much lower-dimensional model (5 parameters) and tuned these parameters separately to each single stereo pair as in (4.16).

$$\beta_i^* = \underset{\beta}{\operatorname{argmax}} \quad P_{\beta_i}(u_i|x_i) \tag{4.16}$$

The five parameters are tuned to each individual input stereo pair, although the method could be used to tune a single parameter setting over a corpus of stereo pairs. The main difference between their work and ours is that we train highly parameterized monocular depth cues. Another difference is that we formulate a general CRF-like model for unsupervised learning based on maximizing conditional likelihood and avoid the need for the independence assumptions used by Zhang and Seitz by using contrastive divergence — a

general method for optimizing loopy CRFs (Hinton 2002; Carreira-Perpiñán and Hinton 2005).

There is also related work on learning highly parameterized monocular depth cues by Saxena et al. in (Ashutosh Saxena and Ng 2007b,a), as well as Hoiem et al. in (Derek Hoiem 2005). The main difference between these methods and ours is that we use unsupervised learning while they use ground truth data to train their system. In Chapter 4.4 we will show that our algorithm with unsupervised training outperformed the results in (Ashutosh Saxena and Ng 2007a). One might argue that stereo pairs constitute supervised training of monocular depth cues. A standard stereo depth algorithm could be used to infer a depth map for each pair which could then be used in a supervised learning mode to train monocular depth cues. However, we demonstrate that training monocular depth cues from stereo pair data improves *stereo* depth estimation. Hence the method can be legitimately viewed as unsupervised learning of a stereo depth. Also the general formulation of unsupervised learning by maximizing conditional likelihood, like the shift from MRFs to CRFs, may have significance beyond computer vision.

Other related work includes that of Scharstein and Pal (Scharstein and Pal 2007) and Kong and Tao (Kong and Tao 2004). In these cases somewhat more highly parameterized stereo models are trained using methods developed for general CRFs. However, the training uses ground truth depth data rather than unlabeled stereo pairs.

We learn MRF parameters using contrastive divergence (Hinton 2002; Carreira-Perpiñán and Hinton 2005), a general MRF learning algorithm capable of training large models. Another work that also described learning on MRF using contrastive divergence is the Field of Experts system by Roth and Black in (Roth and Black 2005). In (Roth and Black 2005), the authors describe an extension of the traditional MRF model by learning potential functions over extended pixel neighborhoods, which they then implemented with two vision

applications: image denoising and image impainting. To training the model parameters, they also used contrastive divergence update, aiming at minimizing the KL-divergence between the model distribution and the data distribution. The MCMC sampling strategy was used to approximate the expectation over model distribution. The data distribution is easy to compute by computing the average over training data. Similar to other aforementioned related work, they also used training data with ground truth label.

Our training approach is actually more related to the hidden CRF model (Quattoni et al. 2007) than the standard CRF. Our model includes one more variable, the latent variable y which is not observed in the training data. Specifically in our slanted plane stereo model, x denoted segmented left image, u denoted a right image, and y was an assignment of a plane to each superpixel of x. y is not in the training data, but it is part of the model. However, in this chapter we aim at describing our model in a more general level rather than a model specifically applied for stereo vision, as defined by (4.17). The conditional probability model $P_\beta(u|x)$ now is defined not only over x and u but also by an arbitrary latent variable y.

$$P_\beta(u|x) = \sum_y P_\beta(u, y|x) \tag{4.17}$$

Given a set of stereo image pairs as training data $(x_1, u_1), \ldots (x_N, u_N)$ conditional EM is an algorithm for locally optimizing the parameter vector β so as to maximize the probability of the u values given the x values in the training data, as formulated in equation (4.18) (This is exactly equation (1.16) from Section 1.4).

$$\beta^* = \operatorname*{argmax}_\beta \sum_{i=1}^{N} \ln P_\beta(u_i|x_i) \tag{4.18}$$

Conditional EM is a straightforward modification of EM and is defined by the following

two updates where β is initialized with domain specific heuristics.

$$P_i(y) := P_\beta(y|x_i, u_i) \tag{4.19}$$

$$\beta := \underset{\beta}{\operatorname{argmax}} \sum_{i=1}^{N} \mathrm{E}_{y\sim P_i} \left[\ln P_\beta(u_i, y_i, |x_i)\right] \tag{4.20}$$

Update (4.19) is called the E step and update (4.20) is called the M step. Hard EM, also known as Viterbi training, works with the single most likely (hard) value of y rather than the (soft) distribution P_i defined by (4.19). Hard conditional EM locally optimizes the following version of (4.18).

$$\beta^* = \underset{\beta}{\operatorname{argmax}} \sum_{i=1}^{N} \max_{y} \ln P_\beta(u_i, y|x_i) \tag{4.21}$$

Hard conditional EM is defined to be the process of iterating the updates (4.22) and (4.23) below which can be interpreted as hard versions of (4.19) and (4.20).

$$y_i := \underset{y}{\operatorname{argmax}} P_\beta(u_i, y|x_i) \tag{4.22}$$

$$\beta := \underset{\beta}{\operatorname{argmax}} \sum_{i=1}^{N} \ln P_\beta(u_i, y_i|x_i) \tag{4.23}$$

We will call (4.22) the hard E step and (4.23) the hard M step. Updates (4.22) and (4.23) are both coordinate ascent steps for the objective defined by (4.21). However, we refer to (4.22) and (4.23) as hard conditional EM rather than simply "coordinate ascent" because of the clear analogy between (4.21), (4.22), (4.23) and (4.18), (4.19), (4.20).

The parameter vector β is composed of two components $\beta = (\beta_y, \ \beta_u)$ where β_y parameterizes $P(y|x)$ and β_u parameterizes $P(u|x,y)$. Note that the probability model on the right hand side of (4.22) can be further expanded as follows:

$$\begin{aligned} y_i &:= \operatorname*{argmax}_y P_{\beta_y}(y|x_i) P_{\beta_u}(u_i|x_i,y) \\ &:= \operatorname*{argmax}_y \ln\Big(P_{\beta_y}(y|x_i) P_{\beta_u}(u_i|x_i,y)\Big) \\ &:= \operatorname*{argmin}_y E_{\beta_y}(x_i,y) + E_{\beta_u}(x_i,u_i,y) + \ln Z_{\beta_u}(x,y) + \ln Z_{\beta_y}(x) \end{aligned} \tag{4.24}$$

We have that:

$$Z_{\beta_y}(x) = \sum_y e^{(-E_y(x,y,\beta_y)\ )} \tag{4.25}$$

$$Z_{\beta_u}(x,y) = \sum_u e^{(-E_u(x,u,y,\beta_u)\ )} \tag{4.26}$$

where equation (4.25) corresponds to the sum of (3.4) and (3.6), equation (4.26) corresponds to (3.5). Since (4.25) does not depend on y, we can eliminate it from (4.24). It is less obvious to show that (4.26) does not depend on y.

$$\begin{aligned} Z_{\beta_u}(x,y) &= \sum_u e^{(-E_u(x,u,y,\beta_u)\ )} \\ &= \sum_u e^{-(\sum_{p\in X}(\Phi^x(p)-\Phi^u(p+d(p,y))\ )^2)} \end{aligned} \tag{4.27}$$

We assume that the mapping from x to u in (4.27) is a bijection, i.e. each pixel p in x only maps to one unique pixel in u (since x and u have the same size, this condition is sufficient to guarantee bijection). We can consider the matching cost from equation (3.5) as a distance function between the left image x and one of its prediction $\hat{x}$, where $\hat{x}$ is a function of a pair (u, y). Note that (4.26) is a sum over all possible values of u, and pairing all possible u with y can generate all possible values of $\hat{x}$. We can rewrite (4.26) as follows:

$$\begin{aligned} Z_{\beta_u}(x, y) &= \sum_{u} e^{(-E_u(x,u,y,\beta_u)\ )} \\ &= \sum_{\hat{x}} e^{(-E_{\hat{x}}(x,\hat{x},\beta_u)\ )} \end{aligned} \tag{4.28}$$

It follows that (4.26) does not depend on y either, and therefore can also be eliminated from (4.24). We can rewrite (4.24) exactly as:

$$y_i := \operatorname*{argmin}_{y} E_{\beta_y}(x_i, y) + E_{\beta_u}(x_i, u_i, y) \tag{4.29}$$

Each of the terms in the right hand side of (4.29) is a CRF. In the case of the slanted plane stereo model, the hard E step (4.24) is implemented using a stereo inference algorithm which computes y_i by minimizing the corresponding energy functionals. The inference algorithm is described in Section 4.1.

Our implementation of the hard M step relies on a factorization of the probability model into two conditional probability models each of which is defined by an energy functional. Unlike CRFs, we do not require the energy functional to be linear in the model parameters.

$$P_{\beta_u,\beta_y}(u,y|x) = P_{\beta_y}(y|x)P_{\beta_u}(u|x,y) \tag{4.30}$$

$$P_{\beta_y}(y|x) = \frac{e^{(-E_y(x,y,\beta_y)\ )}}{Z_y(x,\beta_y)} \tag{4.31}$$

$$Z_y(x,\beta_y) = \sum_y e^{(-E_y(x,y,\beta_y)\ )}$$

$$P_{\beta_u}(u|x,y) = \frac{e^{(-E_u(x,u,y,\beta_u)\ )}}{Z_u(x,y,\beta_u)} \tag{4.32}$$

$$Z_u(x,y,\beta_u) = \sum_u e^{(-E_u(x,u,y,\beta_u)\ )}$$

Given this factorization of the model, the hard M step (4.23) can be written as the following pair of updates.

$$\begin{aligned}
\beta_y &:= \operatorname*{argmax}_{\beta_y} \prod_i P_{\beta_y}(y_i|x_i) \\
&:= \operatorname*{argmax}_{\beta_y} \sum_i \ln P_{\beta_y}(y_i|x_i) \\
&:= \operatorname*{argmin}_{\beta_y} \sum_i E_y(x_i,y_i,\beta_y) \\
\beta_u &:= \operatorname*{argmax}_{\beta_u} \prod_i P_{\beta_u}(u_i|x_i,y_i) \\
&:= \operatorname*{argmax}_{\beta_u} \sum_i \ln P_{\beta_u}(u_i|x_i,y_i) \\
&:= \operatorname*{argmin}_{\beta_u} \sum_i E_u(x_i,u_i,y_i,\beta_u)
\end{aligned} \tag{4.33}$$

Let L abbreviate the quantity being maximized in the right hand side of (4.33) and let $E_i(y)$ abbreviate $E_y(x_i, y_i, \beta)$. We can express the gradient of L as follows.

$$\nabla_{\beta_y} L = \sum_{i=1}^{N} \left(\mathrm{E}_{y \sim P_y(y|x_i,\beta)} \left[\nabla_{\beta_y} E_i(y) \right] - \nabla_{\beta_y} E_i(y_i) \right) \tag{4.34}$$

A similar equation holds for (4.33).

The first term in the right hand side of (6.22) is the expectation over the model distribution, while the second term is the expectation over the training data. The first term involves $P_{\beta_y}(y|x_i)$, which is a very complicated probability distribution and usually is extremely difficult to compute. However in such cases, we can estimate the expectation of such distributions by taking the average over a sufficient amount of samples, obtained by sampling y from $P_{\beta_y}(y|x_i)$ using an MCMC sampling process. The regular MCMC sampling method is performed by running a sequence of Metropolis or Metropolis-Hastings sampling steps until the Markov chain mixes - i.e. reaching the stationary distribution. Usually it is a difficult problem to determine how many steps are needed to converge to the stationary distribution within an acceptable error. Contrastive divergence proposed by G. Hinton et. al. (Hinton 2002; Carreira-Perpiñán and Hinton 2005) is a technique that allows us to sample z using only one or two MCMC steps rather than a long running MCMC process. Since we can estimate $\nabla_{\beta_y} L$, we can then optimize (4.33), and similarly (4.33), by gradient descent.

For the experiments reported here we use contrastive divergence (Hinton 2002; Carreira-Perpiñán and Hinton 2005) to sample y rather than a long running MCMC process. In contrastive divergence we initialize y to be y_i and then perform only a few (one or two) MCMC updates to get a sample of y. Contrastive divergence can be motivated by the observation that if y_i is assumed to be drawn at random from $P_\beta(y|x_i)$ then the expected contrastive divergence update is zero. So as β better fits the pairs (x_i, y_i) one expects the contrastive divergence gradient estimate to tend to zero. Furthermore, because only a few updates are

used in the MCMC process, contrastive divergence runs faster and with lower variance than a longer running MCMC process. Contrastive divergence yields a consistent estimator for β whenever the expected update direction is the gradient of a convex potential function.

We will demonstrate the experimental results in the next chapter to show that training monocular cues from stereo pair data alone improves stereo depth estimation. Our unsupervised learning algorithm implicitly trains shape from texture monocular surface orientation cues. Moreover, our stereo model with texture cues, only by unsupervised training, outperformed the results in (Ashutosh Saxena and Ng 2007a). Throughout this chapter, stereo vision was considered as a simple setting to investigate unsupervised learning and hence seems a good place to start. However we believe that our approach to unsupervised learning based on maximizing conditional likelihood can be generalized to other, maybe much more sophisticated models.

4.4 Experimental Results

We implement two training methods in our experiments — supervised and unsupervised. For each of the supervised and unsupervised training methods we train both a version with texture cues for surface orientation and a version without such cues. In supervised training we set z_i (for each training image) by fitting a plane in each segment to the ground truth disparities for that segment. We then train the model using a contrastive divergence implementation of the hard M step (4.23) which we describe in more detail below. In supervised training we use only a single setting of z and run one iteration of (4.23). In unsupervised training we use the same separation into training and test pairs but do not use ground truth on the training pairs. Instead we iterate (4.22) and (4.23) six times starting with initial values for the parameters.

We use the inference algorithm described in section 4.1 to implement the hard E step (4.22). This uses a form of max-product particle belief propagation. Given an assignment z of a plane to each segment, we propose 15 additional candidate planes for each segment by adding Gaussian noise to the plane specified by z. The plane parameters A and B have units of pixels of disparity per pixel in the image, and hence are dimensionless. Typical values of $|A|$ and $|B|$ are from .1 to 1. In the proposal distribution we use Gaussian noise with a standard deviation of .007 for each of A and B and use a deviation of .1 pixels for C. We perform six rounds of proposing and selecting.

We implement the hard M step (4.23) by first breaking it down into (4.33) for training $P(z|x, \beta_z)$ and (4.33) for training $P(y|x, z, \beta_y)$. The form of the match energy (a simple quadratic energy) allows a closed form solution for (4.33). We implement (4.33) by gradient descent using a contrastive divergence approximation of the gradient in (6.22). We perform 8 gradient descent parameter updates with a constant learning rate. To estimate the expectation in (6.22) in each parameter update we generate 10 alternative plane assignments using single MCMC stochastic step starting at z and accepting or rejecting Gaussian noise added once to each plane. The MCMC process proposes a new plane for each segment by adding Gaussian noise and then accepting or rejecting that proposal using the standard Metropolis rejection rule.

4.5 Experimental Results with the Middlebury Dataset

Table 4.1 shows the performance of our system on the Middlebury stereo evaluation (version 2). The numbers shown are for unsupervised training with texture features. In this case all four images where used as unsupervised training data (ground truth disparities were not used in training). Figure 4.2 shows the inferred depth maps for the Middlebury images

Tsukuba			Venus			Teddy			Cones			Avg.
nonocc	all	disc	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc	bad
1.11	1.37	5.79	0.10	0.21	1.44	4.22	7.06	11.8	2.48	7.92	7.32	4.23%
2.58	4.66	11.8	0.47	0.64	6.1	6.72	6.98	16.1	6.93	9.33	16.6	7.41%

Table 4.1: Performance on the Middlebury stereo evaluation. First row is the current best method (A. Klaus and Karner 2006). Second row is our method, in which the numbers shown are for unsupervised training with texture features.

at various points in the parameter training. The figure shows a clear improvement as the parameters are trained.

4.6 Experimental Results with the Stanford Dataset

	RMS Disparity Error (pixels)	Average Error $\lvert \log_{10} Z - \log_{10} \hat{Z} \rvert$
Saxena et al. (Ashutosh Saxena and Ng 2007a)		.074
Unsuper., Notexture	1.286	.075
Unsuper., Texture	1.1608	.0723
Super., Notexture	1.1142	.0709
Super., Texture	1.0319	.0686

Table 4.2: RMS disparity error (in pixels) and average error (average base 10 logarithm of the multiplicative error) on the Stanford stereo pairs for four versions of our systems plus the best reported result from (Ashutosh Saxena and Ng 2007a) on this data. Each system was either trained using the ground truth depth map (supervised) or trained purely from unlabeled stereo pairs (unsupervised) and either used texture cues (Texture) for surface orientation or did not (Notexture). Note that texture information helps improve the performance in both supervised and unsupervised cases.

We have also run experiments on a set of stereo image pairs taken from the Stanford color stereo dataset [1] which has been used to train monocular depth estimation (Ashutosh Sax-

1. http://ai.stanford.edu/ asaxena/learningdepth/data

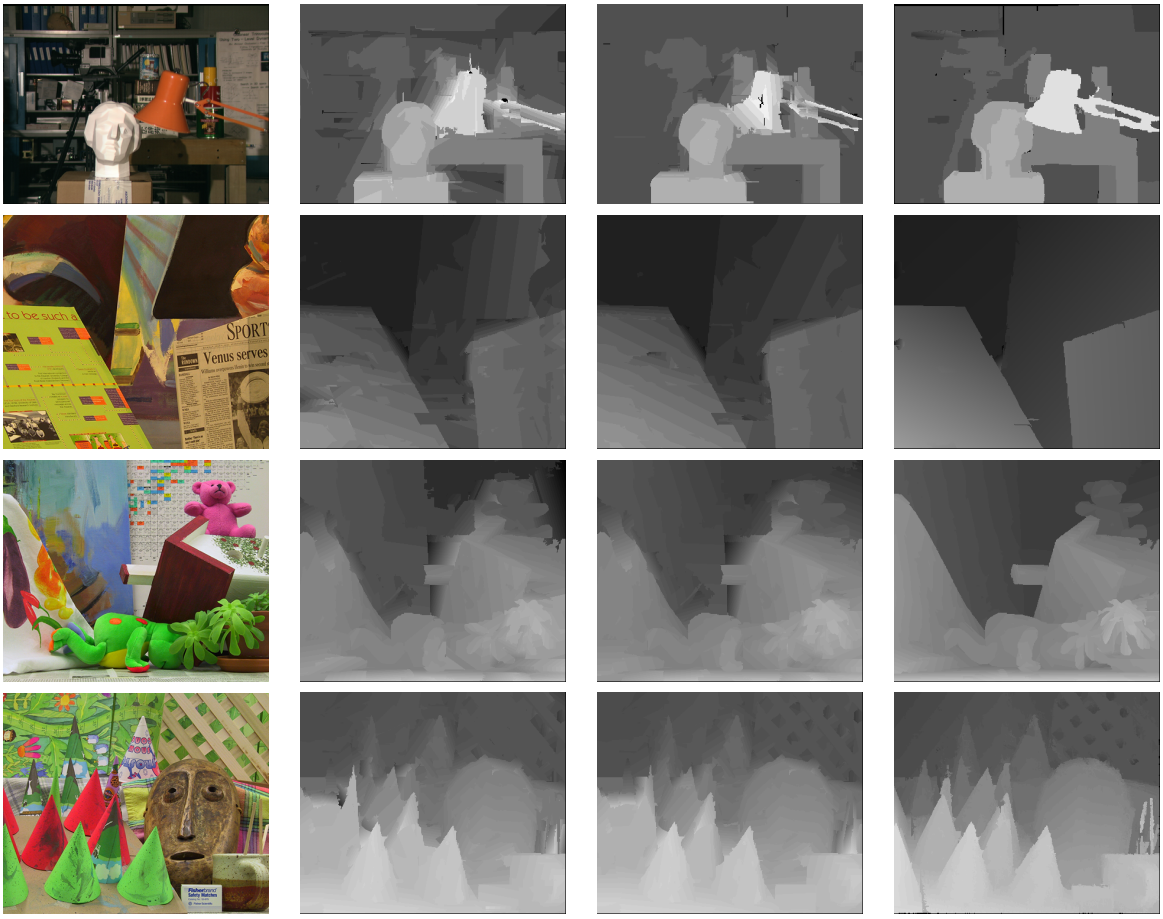


Figure 4.2: Improvement with training on the Middlebury dataset.

ena 2005; Ashutosh Saxena and Ng 2007b,a). The images cover different types of outdoor scenes (buildings, grass, forests, trees, bushes, etc.) and some indoor scenes.

First we performed epipolar rectification on this dataset. Given a pair of stereo images, rectification computes a transformation matrix for each image plane such that pairs of conjugate epipolar lines become collinear and parallel to one of the image axes (usually the horizontal one). The rectified images can be thought of as acquired by a new stereo rig, obtained by rotating the original cameras. The important advantage of rectification is that computing stereo correspondences is made simpler, because search is done along the horizontal lines of the rectified images.

Standard rectification methods such as (Irsara and Fusiello 2006) rely on detecting and matching a sparse set of feature points to compute the epipolar geometry between the left and right image. The rectification then is achieved usually by minimizing a measure of distortion - a score function involving only the point correspondences.

Using a rectification kit from Fusiello et al. (Irsara and Fusiello 2006) failed in 57 out of 257 stereo pairs in the Stanford dataset (The failed cases are cases where the rectification score function exceeded a specified threshold). Therefore we came up with a different rectification approach that we call "'dense rectification"'. First we observe that all stereo pairs in the dataset were taken from a stereo rig with almost the same calibration parameters. We then aim at determining one or only a few rectification solutions for all pairs in the dataset, rather than computing a separate rectification for each pair. Besides, since epipolar rectification is considered an important preprocessing stage of dense stereo matching, the ultimate goal of rectification is also to optimize the dense stereo matching score. Therefore our method strives to find the rectification solutions that directly serve for this purpose: minimizing the dense stereo energy score rather than the feature point correspondence distortion score. To compute the dense stereo energy score for each pair, we simply use the efficient stereo algorithm in (Felzenszwalb and Huttenlocher 2006). Our approach can be summarized as follows:

- Run rectification code from Fusiello et al. (Irsara and Fusiello 2006) on the whole dataset.
- Use stereo energy to pick the best 200 image pairs. (Consider the other 57 images outliers)
- Do K-means clustering on the 200 transformation matrices of these 200 image pairs with $(K = 2, 3, 4)$.

- Pick best K = 2. Compute the 2 mean transformation matrices.
- Apply each of the transformation matrices to rectify the 257 images. For each image we pick the rectified one with better stereo energy score (instead of the rectification score).
- We obtain 257 rectified image pairs.

With our rectification algorithm, we successfully rectified all stereo image pairs in the dataset. The rectified dataset has been put online for public use [2]. Our results are directly comparable to (Ashutosh Saxena and Ng 2007a), i.e we use the same training set and test set (193 pairs for training, 64 pairs for testing). Table 4.2 shows that we outperform their results.

In terms of running time, it takes our algorithm around 10 seconds on average to do inference on one stereo pair, with image size 509x403. One learning step on th whole training set takes about 45 minutes. At each contrastive divergence step, we took 15 samples to compute the gradient of the energy function.

4.7 Conclusion

In many applications we would like to be able to build systems that learn from data collected from mechanical devices such as microphones and cameras. Stereo vision provides perhaps the simplest setting in which to study unsupervised learning. We have formulated an approach to unsupervised learning based on maximizing conditional likelihood and demonstrated its use for unsupervised learning of stereo depth with monocular depth cues. Ultimately we are interested in learning highly parameterized sophisticated models including, perhaps, models of surface types, shape from shading, albedo smoothness

2. http://ttic.uchicago.edu/ ntrinh/shared/stanford/

priors, lighting smoothness priors, and even object pose models. We believe that unsupervised learning based on maximizing conditional likelihood can be scaled to much more sophisticated models than those demonstrated here.

Figure 4.3: Several depth map results on the rectified Stanford stereo dataset. The first 2 row are some example images. The second 2 rows are our output depth maps.

CHAPTER 5

UNSUPERVISED CRF LEARNING FOR MONOCULAR DEPTH ESTIMATION

In this chapter we present our unsupervised CRF learning approach to the problem of monocular depth estimation (MDE). MDE is the problem of predicting the depth at each pixel of a single input image. This is a much more challenging problem than stereo vision. There exist an intrinsic ambiguity between local image information and the 3-D location of the real-world object, due to perspective projection. In other words, a point in the image plane can correspond to any 3-D point lying on a line connecting the camera center and that image point.

In recent work by Saxena et al. (Ashutosh Saxena 2005; Ashutosh Saxena and Ng 2007b,a) an MDE system is trained from images paired with laser range finder depth maps. Here we are interested in training an MDE system using only stereo pairs — pairs of images taken from slightly different angles. Stereo pair training for MDE is interesting for a variety of reasons. First, in some cases it may be easier to obtain stereo pairs than to obtain images paired with laser range finder data. Second, learning MDE from stereo pairs can be viewed as a first step toward learning MDE from video sequences. Video is easily obtained and widely available and could provide an enormous volume of training data. Third, stereo pair training for MDE serves as a case study in unsupervised two-view learning which might provide a model for this form of learning in other applications.

We train a monocular depth estimator using the same stereo pair database as used in (Ashutosh Saxena and Ng 2007a) but without the use of ground truth depth maps. For

learning we use basically the same unsupervised CRF learning algorithm we applied for Stereo Vision with Monocular Cues in Chapter 4. Our hard EM algorithm first estimates depth using stereo alone (E step), then trains the monocular predictor (M step), then re-estimates depth using both stereo and monocular cues, then retrains the monocular predictor, and so on. While most of the performance of the monocular predictor is achieved by training on the initial stereo depth estimates, a modest improvement is seen for both view prediction and for ground truth prediction at later EM iterations. The performance on ground-truth prediction is similar to that reported in (Ashutosh Saxena and Ng 2007a) in spite of the absence of ground truth in training.

In this chapter, first we present our depth estimation model with monocular depth cues. Next, we introduce the notion of view prediction and its relation to the conditional likelihood that the learning algorithm wants to maximize. We then describe the application of our unsupervised CRF learning algorithm for this specific model, aiming at training the MDE. The last section demonstrates experimental results.

5.1 The model

Our model is simply an extension of the standard dense stereo model.

$$E_w(I^{(1)}, I^{(2)}, d) = \begin{cases} \lambda_d \sum_p \min\left(||Y_i^{(1)}(p) - Y_i^{(2)}(p + d_i(p))||^2, \tau_d\right) \\ +\lambda_s \sum_{p,q \in N(p)} min(|d(p) - d(q)|, \tau_s) \\ +\sum_p (d(p) - w \cdot X^{(1)}(p))^2 \end{cases} \tag{5.1}$$

where $(I^{(1)}, I^{(2)})$ is a stereo pair, d is the depth map of $I^{(1)}$, $d(p)$ is the depth value assigned to pixel p, $Y_i^{(1)}(p)$ is the image feature vector corresponding to pixel p in $I^{(1)}$, $X^{(1)}(p)$ is the monocular feature vector of p, $N(p)$ denotes the set of neighbors of pixel

p, and w is the MDE parameter. For this current model, we assume that other parameters $\lambda_d, \lambda_s, \tau_d, \tau_d$ are fixed, and the only parameter we want to learn is w.

Note that (5.1) defines an MRF, in which the features are stereo, smoothness and monocular features, the hidden labels are the pixel depth values. The first term in (5.1) is the match energy measuring how well the disparity assignment d agrees with the input image pair, the second term is the smoothness energy enforcing the smoothness assumption, and the third term is the MDE term. For MDE we want to construct a disparity map d given only the first view $I^{(1)}$. We assume a feature vector $X^{(1)}(p)$ at each pixel p of $I^{(1)}$. A particular choice of features will be discussed in Section 5.4 but we can think of $X^{(1)}(p)$ as containing a constant feature (a bias feature), information about the y component of the pixel p, and the value of various filters applied to image $I^{(1)}$ at the pixel p. Our MDE energy term now is the sum squared difference between the assigned disparity $d(p)$ and a locally predicted disparity $w \cdot X(w)$.

We now define a monocular disparity predictor $d_w^*(I^{(1)})$ determined by a parameters vector w as follows.

$$d_w^*(I^{(1)}) = \underset{d}{\operatorname{argmin}} \ (\sum_p (d(p) - w \cdot X^{(1)}(p))^2 + \lambda_s \sum_{p,q \in N(p)} min(|d(p) - d(q)|, \tau_s) \tag{5.2}$$

The second term here is exactly the robust L_1 smoothing term in (5.1). (5.2) itself also defines an MRF, without the stereo feature.

5.2 Stereo Pair View Prediction and Conditional Likelihood

Here we introduce the notion of view prediction. In view prediction the goal is to predict the second view given the first. View prediction training can be expressed with the following schema where the distortion and complexity functions are application specific.

$$f^* = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \sum_{i=1}^{M} \text{Distortion}\left(f\left(x_i^{(1)}\right), x_i^{(2)}\right) + \lambda \text{Complexity}(f) \tag{5.3}$$

In the system developed here we take $f\left(x^{(1)}\right)$ to be an image derived by applying an inferred disparity map to the image $x^{(1)}$ and the distortion function is simply the pixel-wise squared error (the L_2 distance) between $x^{(2)}$ and $f\left(x^{(1)}\right)$. A log-loss version of view prediction can be formulated as follows where $P_w(x^{(2)}|x^{(1)})$ is a conditional probability (or probability density) parameterized by w.

$$w^* = \underset{w}{\operatorname{argmin}} \sum_{i=1}^{M} \ln \frac{1}{P_w\left(x^{(2)}|x^{(1)}\right)} + \lambda \text{Complexity}(w) \tag{5.4}$$

For stereo pairs we might define $P_w\left(x^{(2)}|x^{(1)}\right)$ by inferring a disparity map from $x^{(1)}$ under parameter setting w and then using the inferred disparity map to predict a probability density in color space at each pixel of $x^{(2)}$. Note that view prediction training as defined by (5.3) or (5.4) is meaningful without labels.

We consider stereo pairs of the form $(I^{(1)}, I^{(2)})$ where $I^{(1)}$ and $I^{(2)}$ are images each of which is an assignment of a color vector to pixels. We write $I^{(j),c}(p)$ for the color c component at pixel p in image $I^{(j)}$. Let d denote a disparity map — a function that assigns a disparity $d(p)$ to each pixel p. Disparity defines a mapping between the two images. The position (x, y) in $I^{(1)}$ corresponds to the position $(x, y + d(x, y))$ in image $I^{(2)}$. We let p range over pixels where each pixel is defined by a pair of integer coordinates (x, y) and we

write $p+d(p)$ as an abbreviation for $(x, y+d(x,y))$ where (x,y) is the pixel p. We assume that $d(p)$ is integral. We will mostly work with disparity d rather than depth Z where d and Z are related by $Z = bf/d$ where b is the baseline (the distance between the two cameras) and f is the focal length. We assume that b and f are known.

Given a disparity map d and a reference image $I^{(1)}$ we construct a prediction $\hat{I}^{(2)}$ which approximately satisfies the following equation.

$$\hat{I}^{(2),c}(p + d(p)) = \frac{\sigma^{(2),c}}{\sigma^{(1),c}}(I^{(1),c}(p) - \bar{I}^{(1),c}) + \bar{I}^{(2),c} \tag{5.5}$$

Here $\bar{I}^{(1),c}$ and $\sigma^{(1),c}$ are the mean and standard deviation of all the color c pixel values in image $I^{(1)}$. The quantities $\bar{I}^{(2),c}$ and $\sigma^{(2),c}$ are defined similarly for image $I^{(2)}$. The use of pixel means and variances for the two images corrects for different biases and gains, in each color channel separately, of the two cameras. We could have assumed knowledge of camera bias and gain of the two cameras rather than knowledge of pixel mean and variance of $I^{(2)}$. But in any case the mean and variance of the pixels in $I^{(2)}$ is only a very small amount of information about $I^{(2)}$. Note that equation (5.5) both overspecifies and underspecifies $\hat{I}^{(2)}$ as a function of $I^{(1)}$ and d. Overdetermined values in $\hat{I}^{(2)}$ are set to the pixel value in $I^{(1)}$ with largest disparity (the closest point) and missing values in $\hat{I}^{(2)}$ are filled in from neighboring values preferring values derived from pixels with low disparity (farthest points).

The overall view predictor $f_w\left(I^{(1)}\right)$ is then defined by (5.5) applied to $I^{(1)}$ and $d_w^*\left(I^{(1)}\right)$. We will work with the following distortion function where P is the number of pixels and C is the number of colors.

$$\text{Distortion}(\hat{I}^{(2)}, I^{(2)}) = \frac{1}{PC} \sum_{p,c} \frac{(\hat{I}^{(2),c}(p) - I^{(2),c}(p))^2}{\left(\sigma^{(2),c}\right)^2} \tag{5.6}$$

Here we have defined distortion to be a percentage of variance. Note that if we take $\hat{I}^{(2)}$ to be simply the mean pixel value of $I^{(2)}$ then we get exactly 100% distortion.

We have pointed out that the view prediction error (5.6) corresponds directly to the conditional probability $P_w\left(I^{(2)}|I^{(1)}\right)$, i.e. the probability of the right image given the left image. This probability is exactly the conditional probability in our unsupervised CRF model:

$$\beta^* = \underset{\beta}{\text{argmax}} \sum_{i=1}^{N} \ln P_\beta(u_i|x_i) \tag{5.7}$$

where u corresponds to the right image $I^{(2)}$, x corresponds to the left image $I^{(1)}$, and β is the model parameter, which corresponds to w.

5.3 Unsupervised CRF Learning for MDE Model

We are now interested in training the parameter vector w so as to minimize the expected distortion (5.6) over fresh stereo pairs. We assume a collection of training pairs

$$\left(I_1^{(1)}, I_1^{(2)}\right), \ldots, \left(I_N^{(1)}, I_N^{(2)}\right)$$

.

We now formulate the hard EM algorithm which first estimates disparity using a classical (untrained) stereo algorithm (E step), then trains a monocular estimator from the inferred disparity map (M step), then re-estimates disparity using both binocular and monoc-

ular cues, retrains the monocular predictor, and so on. We formulate this EM bootstrap algorithm as coordinate descent on a joint energy function of both the disparity map d and the parameters w. Formulating this algorithm as coordinate descent guarantees convergence.

The joint monocular and binocular energy function was defined in (5.1). In our experiments we take $Y_i^{(j)}(p)$ to consist of the color vector at p and the gradient of each color channel at p for a total of $3 + 6 = 9$ feature values for three colors.[1] The hard EM learning algorithm is then defined by the following optimization problem.

$$w^* = \operatorname*{argmin}_w \sum_{i=1}^{N} \min_d E_w(I_i^{(1)}, I_i^{(2)}, d) \tag{5.8}$$

Our training algorithm is coordinate descent on (5.8). We alternatively optimize the "coordinates" w and d_i. We initialize d_i to be the disparity map minimizing the first two terms of (5.1). This corresponds to classical (untrained) binocular inference of the disparity map. Given an initial value for the disparity maps d_i we alternate the following two updates.

$$\begin{aligned} w &:= \operatorname*{argmin}_w \sum_{i=1}^{N} E_w(I_i^{(1)}, I_i^{(2)}, d_i) && (5.9) \\ &= \operatorname*{argmin}_w \sum_{i=1}^{N} \sum_p (d_i(p) - w \cdot X_i^{(1)}(p))^2 \\ d_i &:= \operatorname*{argmin}_d E_w(I_i^{(1)}, I_i^{(2)}, d) && (5.10) \end{aligned}$$

1. Before computing color vector and gradient features the images are normalized to remove bias and gain effects by subtracting the mean color and then dividing each color channel by the standard deviation of that color channel.

		Number of iterations			
		1	2	3	4
Baseline	View Prediction Error	0.458	-	-	-
	Ground Truth RMS	8.644	-	-	-
Monocular	View Prediction Error	0.428	0.419	0.418	0.418
	Ground Truth RMS	1.933	1.921	1.921	1.921

Table 5.1: Quantitative comparison between the ground plane baseline and our model using the average testing error per pixel as functions of the number of iterations. For each model we report: (a) The average view prediction error, measuring the predicted right frame and the ground truth right frame, i.e. the quantity on the left side of (5.6). (b) The RMS disparity error compared with the ground truth (in pixels), measuring the average difference per pixel in disparity value between the predicted disparity and the ground truth disparity.

Note that (5.9) is a least squares regression problem that can be solved exactly with standard codes. Also note that regularization of w can easily be incorporated into (5.9). We approximately solve (5.10) using the efficient loopy BP algorithm of Felzenszwalb and Huttenlocher (Felzenszwalb and Huttenlocher 2006). To relate this to the learning algorithm described earlier in Chapter 4, please note that equation (5.9) corresponds exactly to equation (4.23), and (5.10) corresponds exactly to equation (4.22).

For the experiments reported in Section5.4, parameters of the algorithm other than w are set by hand.

5.4 Experimental Results

We ran the EM bootstrap training algorithm on a set of rectified stereo pairs taken from a color stereo dataset.[2] The images cover different types of outdoor scenes (buildings, grass, forests, trees, bushes, etc.), and some indoor scenes. The images were epipolar rectified using a rectification kit from Fusiello et al. (Irsara and Fusiello 2006). Initial disparity maps were computed for all images using Felzenszwalb and Huttenlocher's efficient loopy BP

2. The dataset is available at http://ai.stanford.edu/ asaxena/learningdepth/data.

Figure 5.1: Resulting disparity maps from (a) the initial stereo algorithm, (b) the trained monocular predictor, (c) the combined monocular and binocular predictor.

algorithm on the first two energy terms of (5.1). By examining the resulting rectifications and depth maps it was clear that some rectifications had failed. At this point we removed from the dataset all pairs for which the energy value achieved by loopy BP was above a specified threshold. This left 204 out of an original 250 stereo pairs on which to train the monocular depth estimator.

The monocular feature vector $X^{(1)}(p)$ consists of local image features computed at 3 different scales, which capture the texture and color cues (similar to (Ashutosh Saxena and Ng 2007a)). We also used the pixel image coordinates as features and included 1 constant feature for a bias term. We have $X^{(1)}(p) \in R^{54}$ (17 image features at each scale for 3 scales, plus 2 spatial features and 1 bias feature). Following (Ashutosh Saxena and Ng 2007a) we made disjoint copies of this feature set for different height levels in the image so that the monocular predictor is trained separately for each height level. We use 40 discrete height levels.

Training the monocular predictor separately at different height levels exploits the fact that these images were taken from a single robot with a fairly stable ground plain orientation relative to the cameras. The stability of the ground plane in these images suggests a baseline monocular prediction in which the disparity at each height level is simply predicted to be the average disparity for that height level across all training data as estimated in the learning algorithm. We will call this the ground plane baseline (even though it is not really a ground plane). The ground plane baseline is stronger than the baseline used in (Ashutosh Saxena and Ng 2007b).

We performed K-fold cross-validation using hard EM stereo pair training and view prediction testing. For each fold, 80% of the data was used for training and 20% was used for holdout testing. For testing, monocular inference was performed using loopy BP to minimize the energy function described in (5.2). Some of the resulting monocular disparity maps for holdout test stereo pairs are shown in Figure 5.1, demonstrating the ability of our monocular predictor to predict depth from different scenes. For the test pairs we also computed disparity maps using binocular only and combined monocular and binocular cues. The disparity prediction system combining both monocular and stereo cues appears to improve the accuracy of the disparity obtained from the initial stereo algorithm, although we did not have quantitative ground truth for the test pairs.

The hard EM algorithm was run for four iterations of bootstrapping resulting in four weight vectors. Each monocular feature weight vector was tested by two measures on test data. First, we measured the view prediction error using formula (5.6) on the holdout stereo pairs. Second, we computed an error relative to ground truth on images from a second data set of 63 single images paired with laser range finder depth maps.[3] We related disparity d to depth Z using $Z = c/d$ where the constant c is determined by a fit to the entire data set

3. We picked all the images with prefix *'img-stats'* from the dataset described in (Ashutosh Saxena and Ng 2007b).

Left image Stereo depthmaps Monocular depthmaps

Figure 5.2: MDE results on some testing images

(the same value of c is used for all test images). We reported RMS disparity error which is the square root of the average over pixels of $(d - c/Z)^2$. The results, averaged across all folds, are shown in Table 5.1.

Table 5.1 shows that our algorithm converges quickly after only a few iterations. The prediction error improves for at least one iteration of training. This again suggests that the monocular cues improve the disparity estimation relative to binocular cues only. The table shows a dramatic improvement in RMS disparity error over the ground plane baseline. We obtained the average error in disparity estimate of 2 pixels, much better than the performance of the ground plane baseline (8.6 pixels).

We also run our hard EM learning algorithm on the real-world road driving rectified stereo sequences acquired and provided by Daimler AG (Liu and Klette 2007). The dataset is the composition of seven sequences, each of them presenting different traffic, road and

lighting conditions. Ground truth depth is not available. For this dataset, we trained our MDE model on six sequences and tested on the remaining sequence. As these are grayscale images, we used 15 image features at each scale. Again for testing, monocular inference was performed using loopy BP to minimize the energy function described in (5.2). We demonstrated the resulting monocular disparity maps of a few frames in the testing sequence in figure 5.2. The results show that the monocular model is able to provide good depth predictions from different scenes in general, although there were still image parts which were not generalized well.

We have emphasized view prediction as a way of evaluating monocular depth estimation in the absence of any ground truth depth labels. We relate the learning problem using view prediction error to the problem of maximizing conditional likelihood in hard conditional EM learning. In Chapter 6, view prediction is employed again as an evaluation metrics for a structure and motion estimation framework. We have implemented our unsupervised CRF learning algorithm based on hard conditional EM for training MDE from stereo pairs only. The algorithm has shown to perform well by both view-prediction measures and by comparisons with ground truth labels, even though ground truth labels were not used in training. Furthermore, performance improved (modestly) over several iterations of learning which indicates that the monocular cues used in the EM process improve depth estimation when both monocular and binocular cues are used (as is reported in (Ashutosh Saxena and Ng 2007a)). In the second experiment, although the desired qualitative analysis could not be performed, we have demonstrated that MDE can also be used for Vision-based driving assistance.

CHAPTER 6

DEPTH AND MOTION ESTIMATION FROM STEREO SEQUENCES

We introduce a unified framework for scene structure and motion estimation on road-driving stereo sequences (Trinh and McAllester 2010). This framework is based on the slanted-plane scene model that has become widely popular in the stereo vision community. Our algorithm iteratively and alternately solves for scene structure and motion. Surface estimation is done using our own slanted-plane stereo algorithm. Motion estimation is achieved by solving a MRF labeling problem using Loopy Belief Propagation. We show that with some specific assumptions about the motion of the camera and the scene, the motion estimation problem can be reduced to a 1D search problem along the epipolar lines. We also propose a novel evaluation metrics, based on the notion of view prediction error previously introduced in Chapter 5. This metrics can be used to evaluate the performance of structure and motion estimation algorithms on stereo sequences without ground truth data. Experimental results on road-driving stereo sequences demonstrate that our algorithm successfully improves the view prediction error although it was not designed to directly optimize this quantity.

6.1 Introduction

Multi-view video sequence is the richest form of image data for scene reconstruction. A subtype in this form that we consider very interesting are stereo video sequences. So far,

to our knowledge, this useful data type has hardly been investigated and exploited for the purpose of recovering scene structure and motion. Some of the very few research work that has discussed this issue was introduced by D. Min et al. (Min and Sohn 2006) and F. Huguet et al. (Huguet and Devernay 2007), in which the authors presented variational methods for scene flow estimation from calibrated stereo image sequences. This data type is interesting for two reasons. First, as opposed to a multi-view video sequence, a stereo sequence can easily be captured using only one moving calibrated binocular camera. Second, this data type provides us with enough constraints to conveniently compute location and motion of scene points simultaneously, since coupling dense stereo matching with motion estimation helps decrease the number of unknowns per image point. More specifically, for stereo sequences, we can formulate structure and motion estimation as an energy minimization problem, in which the model is either an extension of a stereo vision model ((Scharstein and Szeliski 2002)) to also handle scene point motion, or an extension of a Structure from Motion or optical flow model ((Simon Baker and Szeliski 2009)) under a stereo setup. One example of using this latter formulation is the work presented in (Zhang and Kambhamettu 2001).

In this section, we follow the former approach. Based on our slanted-plane stereo model, we develop a new algorithm for structure and motion estimation on road-driving stereo sequences. This framework is based on the slanted-plane scene model that has become widely popular in the stereo vision community (Scharstein et al. 2001). Our algorithm iteratively and alternately solves for scene structure and motion. Surface estimation is done using our own slanted-plane stereo algorithm. Motion estimation is achieved by solving a MRF labeling problem using Loopy Belief Propagation. We show that with some specific assumptions about the motion of the camera and the scene, the motion estimation problem can be reduced to a 1D search problem along the epipolar lines. We also propose a novel

evaluation metrics, based on the notion of view prediction error. Again, it is also much easier and more affordable to collect unlabeled stereo video data than to collect stereo video with associated ground truth (which usually requires using multiple sensors followed by a data fusion step). We argue that view prediction error can be used to evaluate the performance of structure and motion estimation algorithms on stereo sequences without ground truth data. Experimental results on road-driving stereo sequences support our argument by demonstrating that our algorithm successfully improve the view prediction error although it was not designed to directly optimize this quantity. We believe view prediction can be used as a quantitative performance measure on unlabeled multi-view datasets in a variety of applications.

6.2 Proposed approach for Structure and Motion

In this section, we present a detailed description of our unified formulation for scene structure and motion. At each time step t in the video sequence we consider the stereo pair at time t and at time $t+1$ (Figure 6.3). We only observe the three frames: I_L^t, I_R^t and I_L^{t+1} and the frame I_R^{t+1} is unobserved. We want to estimate the 3-D location and motion for all pixels in I_L^t. To simplify the derivation of our framework, we use the following assumptions: the camera's viewing direction is the same direction of the Z axis, and all motion vectors in the scene are parallel to the Z axis. Later in section 6.3 we show that our approach still delivers good estimates even in sequences where these assumptions are violated, e.g: when there is small camera rotation, or rotation of moving objects.

6.2.1 Connection between Disparity and Motion

Our assumptions about the motion of the camera and the scene guarantee the following constraints:

- The image location of the epipoles in all frames of the sequence is constant: As the epipole is simply the projection of the next camera center on the previous frame, this is obvious since the camera always moves along its viewing direction.

- From one frame to the next, a pixel translates along the epipolar line connecting that pixel and the epipole: since all 3D points in the scene move in the same direction with the Z axis, a 3D point always stay in the same 3D line which is parallel to the Z axis. Therefore the epipolar line, which is the projection of this 3D line on the image plane, stays constant, given that the epipole also stays constant. (Figure 6.1)

Consider a 3D point P in the scene. Let R be the distance from the point to the Z axis. Let r_t be the 2D distance from the projection of P to the epipole in the video frame at time t. We can derive the following mathematical relation between r_t, r_{t+1}, v_t and d_t as follows (This relationship is also illustrated in Figure 6.2):

$$\begin{aligned} r_t &= \frac{Rf}{z_t} = \frac{Rf}{fh/d_t} = \frac{Rd_t}{h} \\ r_{t+1} &= \frac{Rf}{z_{t+1}} = \frac{Rf}{z_t - \Delta z_t} \\ &= \frac{Rf}{fh/d_t - \Delta z_t} = \frac{Rd_t}{h(1 - d_t \Delta z_t / fh)} \end{aligned}$$

$$\frac{r_{t+1}}{r_t} = \frac{1}{1 - d_t \Delta z_t / fh} = \frac{1}{1 - d_t v_t} \tag{6.1}$$

$$\frac{d_{t+1}}{d_t} = \frac{r_{t+1}}{r_t} = \frac{1}{1 - d_t v_t} \tag{6.2}$$

where: f is the focal length, z_t is the depth of P at time t (the actual distance of P w.r.t the camera), h is the stereo baseline, d_t is the disparity of the projected pixel of P, v_t is

the velocity value of P. Note that in the derivation above we make use of the following relationship between d_t and z_t: $z_t = fh/d_t$.

6.2.2 3-frame Model for Depth and Motion estimation

At each time step t in the video sequence we consider the stereo pair at time t and the left frame at time $t+1$. The frames are labeled as in Figure 6.3. By equation (6.1), we can see that if we know d_t and v_t, then we can compute r_{t+1}, which means solving for the correspondence between I_L^t and I_L^{t+1}. In other words, we converted the 2D optical flow field problem to the stereo disparity estimation problem and a 1D velocity assignment problem. We define the following energy functions:

$$E = E_{Stereo} + E_S^v + E_M^v \tag{6.3}$$

$$E_S^v = \sum_{P,Q} \min\left(\tau_v, \lambda_v \left|v_P - v_Q\right|\right) \tag{6.4}$$

$$E_M^v = \sum_p \sum_k \lambda_k \left(\begin{array}{rl} & \phi_k^t(p) \\ - & \phi_k^{t+1}(p + Q(p, d_p, v_p)) \end{array} \right)^2 \tag{6.5}$$

where E_{Stereo} is the function in (3.2), $Q(p, d_p, v_p)$ is the function mapping a pixel in I_L^t to a pixel in I_L^{t+1}, and $\phi^t(p)$ is the k-dimensional feature vector corresponding to pixel p in frame at time t. The objective is to find the optimal co-assignment for depth and velocity that minimize the global energy function in (6.3).

$$(d^*, v^*) = \underset{(d,v)}{\operatorname{argmin}}(E)$$

The optimization of E_{Stereo} is introduced in section 4.1.

Minimizing the sum of the other two energy terms: $E_{Motion} = E_S^v + E_M^v$ can be considered another MRF labeling problem: we want to assign a velocity value to each superpixel such that E_{Motion} is minimized. The first term E_S^v penalizes the difference in velocity between two adjacent superpixels, the second term E_M^v is the data cost of assigning a velocity v_p to pixel p, taking into account the image data agreement between frame I_L^t and frame I_L^{t+1}. We solve this MRF labeling problem by Loopy Belief Propagation. We used the max-product BP algorithm with conceptually parallel updates. In our actual implementation however, they were performed sequentially.

Finally, once we have obtained an initial estimate of v_t, we can fix v_t and optimize the function $E_{Stereo}^{3frame} = E_{Stereo} + E_M^v$. Note that this function has 1 more data term compared to the previous stereo function as we can now incorporate data from I_L^{t+1}. We construct an iterative algorithm alternately optimizing the disparity d_t and the velocity v_t for frame I_L^t. Here is the outline of our algorithm:

1. Compute the epipole at I_L^t using a version of the 8-point algorithm.
2. Estimate the disparity map d_t of I_L^t using our stereo algorithm.
3. Use SIFT feature matching to obtain a set of initial velocity values: Each pair of matched SIFT features give us 1 initial velocity.
4. Estimate v_t given d_t: Run Loopy BP to optimize E_{Motion} and assign a velocity value to each superpixel in I_L^t.
5. Fixing v_t, reestimate d_t by optimizing E_{stereo}^{3frame}.
6. Repeat from Step 4.

6.3 Experimental Results

6.3.1 Evaluation Metrics

We use view prediction as evaluation metrics for our experiments here. The idea of using view prediction error has been investigated in Section 5.2, in which we described stereo pair view prediction. View prediction is a general notion that has been motivated by the two-view learning approach in machine learning (Blum and Mitchell 1998). In two-view learning, the goal is to predict the second view given the first. In our context the goal of view prediction is to predict the unobserved data y given the observed data x and the latent variables w.We can express a probabilistic interpretation of view prediction as follows:

$$w^* = \underset{w}{\operatorname{argmin}} \sum_{i=1}^{m} \ln 1/P_w(y|x)$$

where $P_w(y|x)$ is the conditional probability of the unobserved data given the observed data parametrized by w. As our algorithm never observed I_R^{t+1}, in our problem setting we might define $P_w(y|x)$ as the $P(I_R^{t+1}|I_L^t, I_R^t, I_L^{t+1}, d_t, v_t)$. The idea is that the more accurate we estimate (d_t, v_t), the more accurate we can predict I_R^{t+1}.

Given the estimated depth and velocity d_t, v_t, we can compute the prediction $\hat{I}_R^{t+1}$ as follows:

$$I_L^t \overset{d_t,v_t}{\rightarrow} I_L^{t+1} \overset{d_{t+1}}{\rightarrow} \hat{I}_R^{t+1}$$

where d_{t+1} is computed using (6.2). An example of a predicted fourth view is shown in Figure 6.4.

The view prediction error is defined as the pixel RMS between I_R^{t+1} and $\hat{I}_R^{t+1}$:

$$Err(I_R^{t+1}, \hat{I}_R^{t+1}) = \sqrt{\frac{\sum_{p=1}^{N}(I_R^{t+1}(p) - \hat{I}_R^{t+1}(p))^2}{N}} \quad (6.6)$$

We can see a clear analogy between equation (6.6) and equation (5.6) from Section 5.2. The difference is that the computation of the prediction $\hat{I}_R^{t+1}$ here is more complicated, involving both the estimated disparity and velocity. In the absence of ground truth data, it is natural to use view prediction error as a useful tool for quantitative analysis, i.e. to measure how good the estimation of latent variables is. In the specific setting of our problem, the latent variables $w^* = (d^*, v^*)$.

6.3.2 Results on road-driving stereo sequences

We tested our algorithm on 7 gray scale road-driving stereo sequences provided by Daimler AG. Each sequence contains from 250 to 300 rectified, bias gain corrected stereo pairs, taken under different light condition and road setting. Ground truth depth and motion is not available for these datasets. The algorithm estimates the dense map of disparity and velocity value for each left frame in the sequence. Figure 6.5 demonstrates our results on several frames in a sequence.

Table 6.1 shows the average view prediction error (6.6) over seven video sequences after 3 iterations of the algorithm. The results shows that the algorithm successfully improved the estimation over time. The improvement stops after three iterations. Note that the pixel intensity values are scaled to be in the range of $[0 - 1]$.

	Average View Prediction Error
Iter 1	0.0692
Iter 2	0.0622
Iter 3	0.0621

Table 6.1: Average view prediction error (in pixel intensity value) on 7 Daimler road-driving stereo sequences after each iteration of the algorithm.

6.4 Unsupervised Learning of the Three-frame Model

In Chapter 4, we demonstrated our algorithm for unsupervised learning of a highly parameterized stereo vision model involving the shape from texture cues - training the model parameters from unlabeled stereo pair training data. Our approach to unsupervised learning is based on maximizing conditional likelihood.

Throughout Chapter 4, we used stereo vision as a simple setting to investigate unsupervised learning. However we also argued that our approach to unsupervised learning based on maximizing conditional likelihood can be generalized to other, maybe much more sophisticated models.

In this chapter, we formulate our unsupervised learning algorithm specifically for the model described in Section 6.2. We rewrite the 3-frame depth and motion estimation model here:

$$E = E_{Stereo} + E_S^v + E_M^v \tag{6.7}$$

$$E_S^v = \sum_{P,Q} \min\left(\tau_v, \lambda_v \left|v_P - v_Q\right|\right) \tag{6.8}$$

$$E_M^v = \sum_p \sum_k \lambda_k \left(\begin{array}{rl} & \phi_k^t(p) \\ - & \phi_k^{t+1}(p + Q(p, d_p, v_p)) \end{array} \right)^2 \tag{6.9}$$

6.4.1 Unsupervised training using Hard Conditional EM/Bootstrap training

In our three-frame model, we denote x to be the frame I_L^t, $y = (y^{(1)}, y^{(2)})$, where $y^{(1)}$ is the frame I_R^t, $y^{(2)}$ is the frame I_R^{t+1}, and $z = (d, v)$ to be the latent variable, i.e the assignment of disparity and velocity to all pixels in x.

We are interested in training the parameter vector $\beta = (\beta_d, \beta_v)$ of the model in (6.7) so as to optimize the conditional probability of y given x.

$$\beta^* = \underset{\beta}{\operatorname{argmax}} \sum_{i=1}^{N} \ln P_\beta(y_i|x_i) \tag{6.10}$$

$$= \underset{\beta}{\operatorname{argmax}} \sum_{i=1}^{N} \ln P_\beta(y_i^{(1)}, y_i^{(2)}|x_i) \tag{6.11}$$

Hard conditional EM locally optimizes the following version of (6.10).

$$\beta^* = \underset{\beta}{\operatorname{argmax}} \sum_{i=1}^{N} \max_{z} \ln P_\beta(y_i, z|x_i) \tag{6.12}$$

$$= \underset{\beta}{\operatorname{argmax}} \sum_{i=1}^{N} \max_{(d,v)} \ln P_\beta(y_i^{(1)}, y_i^{(2)}, d, v|x_i) \tag{6.13}$$

Specifically, hard conditional EM iterates the following two updates.

$$z_i := \underset{z}{\operatorname{argmax}} P_\beta(y_i, z|x_i) \tag{6.14}$$

$$(d_i, v_i) := \underset{(d,v)}{\operatorname{argmax}} P_\beta(y_i^{(1)}, y_i^{(2)}, d, v|x_i) \tag{6.15}$$

$$\beta := \underset{\beta}{\operatorname{argmax}} \sum_{i=1}^{N} \ln P_\beta(y_i, z_i|x_i) \tag{6.16}$$

$$(\beta_d, \beta_v) := \underset{(\beta_d,\beta_v)}{\operatorname{argmax}} \sum_{i=1}^{N} \ln P_{(\beta_d,\beta_v)}(y_i^{(1)}, y_i^{(2)}, d_i, v_i|x_i) \tag{6.17}$$

Equation (6.14) is the hard E step and equation (6.16) is the hard M step. In the case of our three-frame model, the hard E step is implemented using the inference algorithm described in Section 6.2. This iterative algorithm finds the optimal co-assignment for depth d and velocity v that minimize the global energy function in (6.3).

To implement the hard M step, we first rewrite the probability in the right hand side of (6.16) as follows.

$$P_{(\beta_d,\beta_v)}(y^{(1)}, y^{(2)}, d, v|x) = P_{\beta_d}(y^{(1)}, d|x) P_{\beta_v}(y^{(2)}, v|d, x) \tag{6.18}$$

The first term, $P_{\beta_d}(y^{(1)}, d|x)$ can be further factorized as demonstrated in (4.30), (4.31) and (4.32).

The second term, $P_{\beta_v}(y^{(2)}, v|d, x)$ can also be factorized in a very similar way.

$$P_{\beta_v,\beta_y}(y^{(2)}, v|d, x) = P_{\beta_y}(y^{(2)}|v, d, x)P_{\beta_v}(v|d, x) \tag{6.19}$$

$$P_{\beta_y}(y^{(2)}|v, d, x) = \frac{e^{(-E_M^v(x,y,v,\beta_y)\ )}}{Z_y(x, v, \beta_y)} \tag{6.20}$$

$$Z_y(x, v, \beta_y) = \sum_y e^{(-E_M^v(x,y,v,\beta_y)\ )}$$

$$P_{\beta_v}(v|d, x) = \frac{e^{(-E_S^v(x,v,\beta_v)\ )}}{Z_v(x, \beta_v)} \tag{6.21}$$

$$Z_v(x, \beta_v) = \sum_v e^{(-E_S^v(x,v,\beta_v)\ )}$$

Given this factorization, the hard M step in equation (6.16) then can be implemented the same way as described in equations (4.33), (4.33) and (6.22).

Specifically, let $E_i(v)$ abbreviate $E_S^v(x, v, \beta_v)$, the contrastive divergence update equation for β_v can be expressed as follows.

$$\nabla_{\beta_v} L = \sum_{i=1}^{N} \Big(\mathrm{E}_{v\sim P_v(v|x_i,\beta)} \left[\nabla_{\beta_v} E_i(v)\right] - \nabla_{\beta_v} E_i(v_i) \Big) \tag{6.22}$$

A similar equation holds for β_y and $E_M^v(x, y, v, \beta_y)$.

6.4.2 Unsupervised training using fourth view prediction

Here we are interested in training the parameter vector of the model in (6.7) so as to minimize the fourth view prediction error (6.6). First, assume that we have a way to compute the gradient of the view prediction error function (6.6) with respect to the latent variables d, v. Using contrastive divergence can be one of the possibilities. We call this gradient G. Second, we can also assume that the gradient of the latent variables d and v with respect to the model parameters β can also be computed. Let g be this gradient.

By applying the chain rule, the gradient of the view prediction error function with respect to β can be computed as follows.

$$\frac{\partial Err(I_R^{t+1}, \hat{I}_R^{t+1})}{\partial \beta} = G^T g \tag{6.23}$$

We can then use gradient descent to find the optimal β^* optimizing the fourth view prediction error (6.6).

6.5 Conclusion

In this chapter we emphasized view prediction as a way of evaluating scene structure and motion estimation from stereo video data in the absence of any ground truth labels. We introduced a new algorithm for structure and motion estimation on road-driving stereo sequences. Based on specific assumptions about the motion of the camera and the scene, we can reduced the 2D optical flow problem to a 1D velocity value problem. Our algorithm iteratively and alternately solve for structure and motion. Scene structure estimation is done using our own plane-based stereo algorithm. Velocity estimation is completed by solving a MRF labeling problem using Loopy BP. Experiments on road-driving stereo sequences

showed encouraging results, even with video sequences where the scene and camera motion do not fully comply with our assumptions.

Performance analysis was done using our novel evaluation metrics based on the notion of view prediction error. We argue that this evaluation metrics is quite appropriate for algorithms working with stereo sequences as well as multiple view image data, when ground truth data is not available. Our experimental results support this argument. We used hand-tuned parameters for our model in this chapter. Ideally, these parameters should be estimated by an automated method, usually through learning using a labeled training data set. With view prediction error, we believe the problem we solve in this chapter can be another setting where we can apply the unsupervised learning approach based on maximizing conditional likelihood. In other words, the model parameters can be learned using only unlabeled stereo video data.

Since we could relate the learning problem using view prediction error to the problem of maximizing conditional likelihood in hard conditional EM learning, as demonstrated in Chapter 5, in addition to the fact that the problem we address here is also modeled as CRFs, there is obviously the possibility of using this problem as another setting where we can apply our unsupervised CRF learning approach. It is hoped that a general notion of view prediction will eventually facilitate unsupervised learning in a variety of cases with mutual information between views.

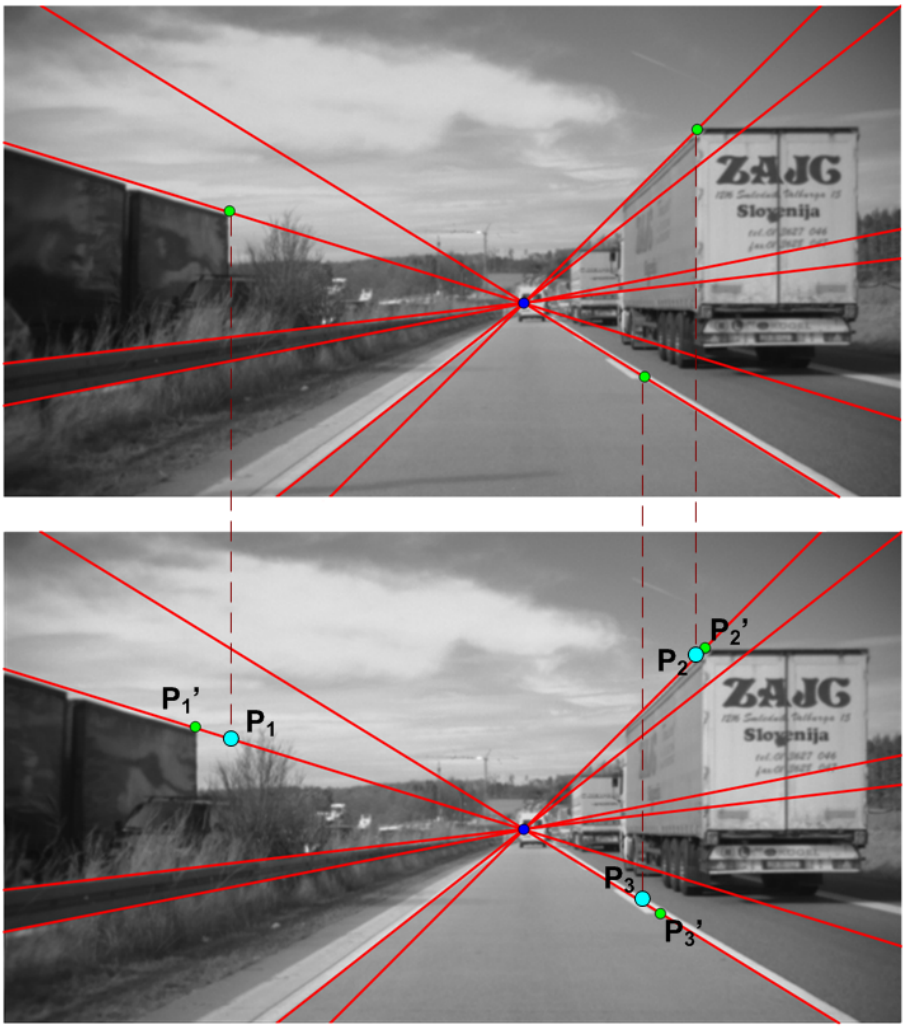


Figure 6.1: As the camera moves forward, each pixel moves along its corresponding epipolar line. The distance it moves depends on both their depth and velocity. Since the truck on the left has highest velocity w.r.t the camera, P_1 moves the longest distance. On the other hand, the truck on the right has almost zero velocity w.r.t the camera, hence P_2 almost stands still. P_3 reflects the motion of the camera w.r.t the road (the scene background).

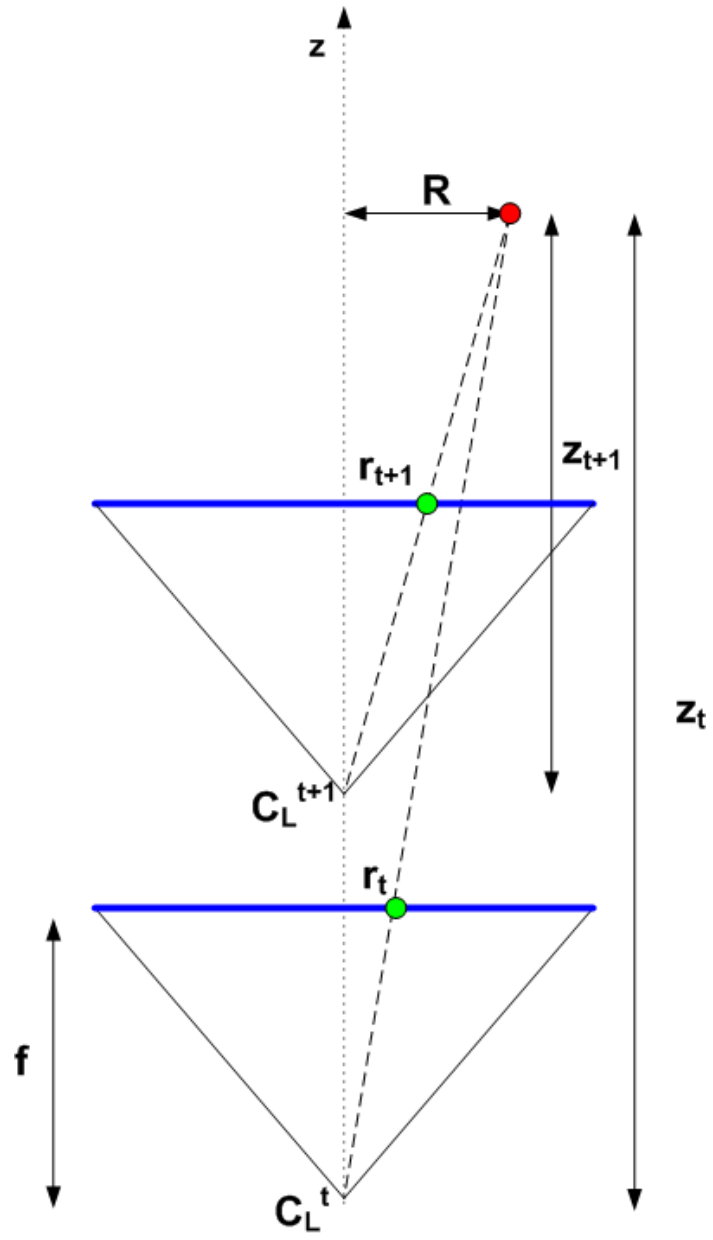


Figure 6.2: The geometric interpretation of equation (6.1).

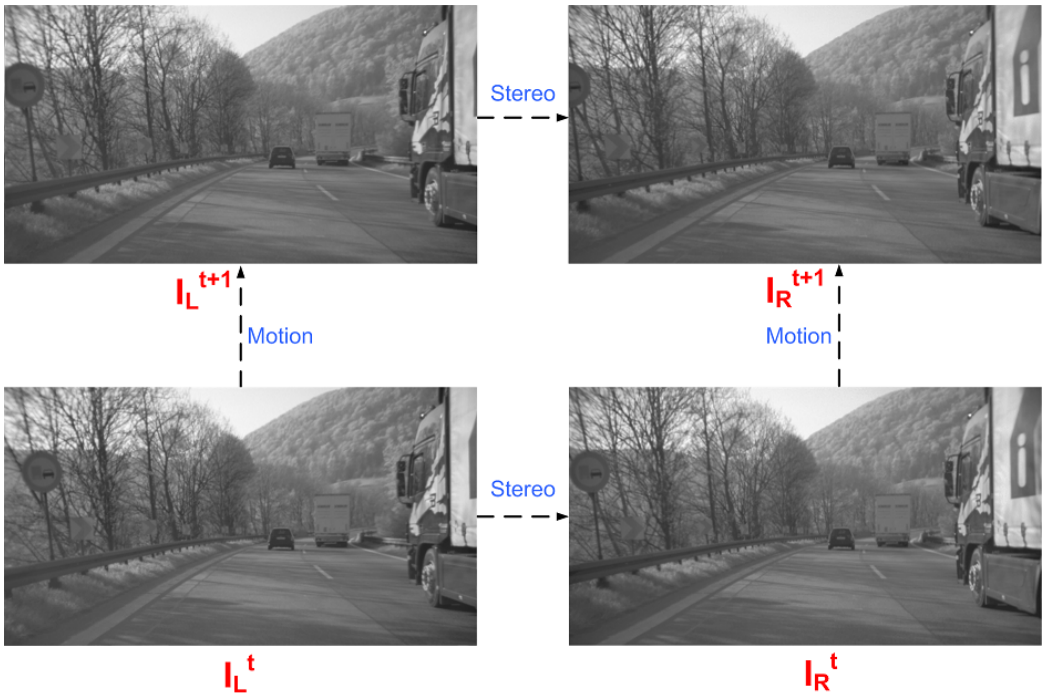


Figure 6.3:

Figure 6.4: Top: original fourth frame I_R^{t+1}. Bottom: predicted fourth frame $\hat{I}_R^{t+1}$. The black pixels are missing values caused by bilinear interpolation. These pixels are excluded when computing the view prediction error.

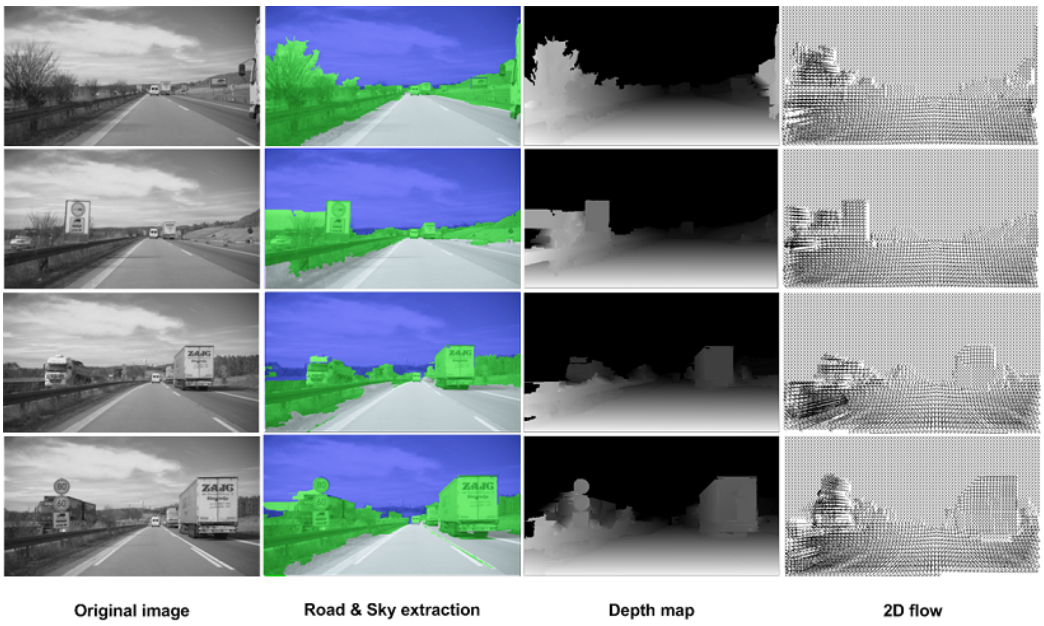


Figure 6.5: Results on a Daimler road driving sequence.

CHAPTER 7

DISCUSSIONS AND CONCLUSIONS

In this thesis, we demonstrated machine learning techniques to build computer vision systems for 3-D scene geometry recovery from images. Scene structure can be recovered from different types of input image data. We focused on three data types: two-view stereo pairs, single image, and stereo video sequences. Building these systems requires algorithms for doing inference as well as learning the parameters of MRFs or CRFs.

For inference in systems with continuous valued variables, or discrete-valued variables with very large domains, it is impossible to directly use a standard discrete MRF inference techniques such as Loopy BP, graph cuts or tree-reweighted message passing. For such systems, we proposed to use a generic Particle-based Belief Propagation (PBP) algorithm closely related to previous work, which we then formulated specifically for our MRF labeling problems. Although we only described a specific use of this generic PBP algorithm, we believe it can be used as an approximate inference scheme for a wide variety of problems that can be formulated by a probabilistic graphical model, especially those containing many random variables with very large or continuous domains. Our approach creates a set of particles for each variable, representing samples from the posterior marginal. The algorithm then continues to improve the current marginal estimate by constructing a new sampling distribution and draw new sets of particles. Its shown by experiments that the algorithm is consistent, i.e. approaches the true values of the message and belief functions with finite samples. Besides accuracy, PBP algorithm also provides good estimate for properties of the distribution, and a representation of state uncertainty. Although an adaptive choice

for the sampling distribution, and such iterative resampling processes require further work to analyze, our results seem to support the notion of sampling from the current marginal estimates themselves, whether from fitted distributions or via a series of MCMC steps.

Learning in MRF involves estimating the optimal parameters of the energy model. For learning, unlike previous work, we trained our system without using ground-truth labeled data. We introduce the unsupervised CRF learning algorithm, based on maximizing conditional likelihood using hard conditional EM. We demonstrated the application of our unsupervised CRF learning algorithm for different scene geometry problems

In unsupervised learning one usually formulates a parameterized probability model and seeks parameter values maximizing the likelihood of the unlabeled training data. Specifically for stereo vision, the model defines the conditional probability of the right image given the left. We introduced a slanted-plane stereo vision model in which we used a fixed over-segmentation to segment the left image into coherent regions called superpixels. We then assign a disparity plane for each superpixel. We formulated the problem of inferring plane parameters as a MRF labeling problem, which can be solved by an energy minimization method. The MRF is a graphical model in which superpixels define nodes and the adjacency between superpixels define edges. Our stereo energy function balances between a data matching term and a smoothness term. We then used our learning algorithm to train this highly parameterized stereo vision model involving the shape from texture cues. We showed that this unsupervised learning algorithm implicitly trains shape from texture monocular surface orientation cues. We demonstrated that training monocular cues from stereo pair data improved stereo depth estimation. Our stereo model with texture cues, only by unsupervised training, outperformed the results in related work on the same stereo dataset. Stereo vision provides perhaps the simplest setting in which to study unsupervised learning. We have formulated an approach to unsupervised learning based on

maximizing conditional likelihood and demonstrated its use for unsupervised learning of stereo depth with monocular depth cues. Ultimately we are interested in learning highly parameterized sophisticated models including, perhaps, models of surface types, shape from shading, albedo smoothness priors, lighting smoothness priors, and even object pose models. We believe that unsupervised learning based on maximizing conditional likelihood can be scaled to much more sophisticated models than those demonstrated in this paper.

In Section 4.5, we demonstrated the results of our stereo algorithm on two different datasets: the Middlebury dataset and the Stanford dataset. We achieved quite respectable performance on the Stanford dataset, obtaining better accuracy than (Ashutosh Saxena and Ng 2007a) both in terms of RMS disparity and average log depth. On the Middlebury dataset, however, our algorithm results in higher eror than state of the art. Moreover, the use of surface orientation monocular cues have clearly improved the performance on the Stanford dataset, but this is not the same case on the Middlebury dataset. There are a few reasons for these. First, the main goal of our experiments is to use stereo as a simple setting to validate the effectiveness of our learning technique, i.e. to show that the UCRF succesfully learns model parameters that improve the objecive function, using only unlabeled training data. In order to achieve state of the art performance in a standard stereo dataset, such as the Middlebury, other requirements must be fulfilled. These usually include both much more careful, data-specific tuning of parameters after multiple evaluation steps, as well as richer, more sophisticated stereo models. For example, many stereo models use geometric (Kolmogorov and Zabih 2001, 2002) or probabilistic (C. Strecha 2004) occlusion model explicitly.

We also observe that the use of the monocular surface orientation cues clearly helped improved the performance on the Stanford dataset, while this is not the case on the Middlebury dataset. Ths can be explained by the different characteristics of two datasets. First,

the Stanford dataset (257 stereo pairs) is much larger than the Middlebury (4 stereo pairs), and therefore is more appropriate for our learning algorithm. Second, all stereo pairs in the Stanford dataset are taken by the same stereo rig, with relatively constant viewing angle and height w.r.t the ground plane. Moreover, they have similar scene types, and the scene usually include many large enough planar surfaces. These features made it much more effective to use the surface orientation cues. On the other hand, this fact may indicate that the learned parameters for the monocular texture term are tuned more finely to the characteristics of the traiing data and do not generalize well to dataset that are quite different.

all stereo pairs in the Stanford dataset were captured by the same stereo rig, with relatively constant height from the ground plane.

Of course our algorithm is not without shortcomings.

We also applied our unsupervised CRF learning approach to the problem of monocular depth estimation (MDE). We learned the MDE model using stereo pairs without ground truth depth maps. While most of the performance of the monocular predictor is achieved by training on the initial stereo depth estimates, a modest improvement is seen for both view prediction and for ground truth prediction at later EM iterations. The MDE performance is similar to that reported in previous work on the same dataset despite the absence of ground truth in learning.

In this thesis, we also addressed the use of an interesting image data type - stereo video sequences. Specifically, for stereo sequences, we can formulate structure and motion estimation as an energy minimization problem, in which the model is either an extension of a dense stereo vision model to also handle scene point motion. Based on the constructed slanted-plane stereo model, we introduced a new algorithm for structure and motion estimation on road-driving stereo sequences. Based on specific assumptions about the motion

of the camera and the scene, we can reduce the 2D optical flow problem to a 1D velocity value problem. Our algorithm iteratively and alternately solve for structure and motion. Surface estimation is done using our own slanted-plane stereo algorithm. Velocity estimation is achieved by solving a MRF labeling problem using Loopy BP. Performance analysis was done using our novel evaluation metrics based on the notion of view prediction error. Experiments on road-driving stereo sequences showed encouraging results, even with video sequences where the scene and camera motion do not fully comply with our assumptions. We emphasized view prediction as a way of evaluating scene structure and motion estimation from stereo video data in the absence of any ground truth labels. We argue that this evaluation metrics is quite appropriate for algorithms working with stereo sequences as well as multiple view image data, when ground truth data is not available. Our experimental results support this argument. We used hand-tuned parameters for our model in this paper. Ideally, these parameters should be estimated by an automated method, usually through learning using a labeled training data set. Our thesis addressed a theoretical discussion regarding this issue, although we did not actually implement the idea. With view prediction error, we believe the scene structure and motion estimation from stereo video problem can be another setting where we can apply our unsupervised learning approach based on maximizing conditional likelihood. In other words, the model parameters can be learned using only unlabeled stereo video data.

REFERENCES

M. Sormann A. Klaus and K. Karner. Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. In *International Conference on Pattern Recognition*, 2006.

M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Trans. SP*, 50(2):174–188, February 2002.

Andrew Y. Ng Ashutosh Saxena, Sung H. Chung. Learning depth from single monocular images. In *Advances in Neural Information Processing Systems*, 2005.

Jamie Schulte Ashutosh Saxena and Andrew Y. Ng. Depth estimation using monocular and stereo cues. In *IJCAI*, 2007a.

Min Sun Ashutosh Saxena and Andrew Y. Ng. 3-d depth reconstruction from a single still image. *International Journal of Computer Vision*, 2007b.

Stan Birchfield and Carlo Tomasi. Multiway cut for stereo and motion with slanted surfaces. In *International Conference on Computer Vision*, 1999.

Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the Workshop on Computational Learning Theory*, 1998.

L. Van Gool C. Strecha, R. Fransens. Wide-baseline stereo from multiple views. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2004.

M. A. Carreira-Perpiñán and G.E. Hinton. On contrastive divergence learning. In *10th Int. Workshop on Artificial Intelligence and Statistics (AISTATS 2005)*, 2005.

N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages I: 886–893, 2005a.

Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005b.

Martial Hebert Derek Hoiem, Alexei A. Efros. Geometric context from a single image. In *International Conference on Computer Vision*, 2005.

A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, New York, 2001.

Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.

Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2), September 2004.

Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient belief propagation for early vision. *Int. J. Comput. Vision*, 70(1):41–54, 2006. ISSN 0920-5691. doi: http://dx.doi.org/10.1007/s11263-006-7899-4.

M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981.

S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 721–741, 1984.

S. Godsill and T. Clapp. Improvement strategies for Monte Carlo particle filters. In J. F. G. De Freitas A. Doucet and N. J. Gordon, editors, *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, New York, 2001.

JM Hammersley and P Clifford. Markov fields on finite graphs and lattices. Unpublished Manuscript, 1971.

R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2nd edition, 2004.

W. K Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, pages 97–109.

Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

F. Huguet and F. Devernay. A variational method for scene flow estimation from stereo sequences. In *International Conference on Computer Vision*, 2007.

A. Ihler and D. McAllester. Particle belief propagation. In *International Conference on Artificial Intelligence and Statistics*, 2009.

L. Irsara and A. Fusiello. Quasi-euclidean uncalibrated epipolar rectification. In *Rapporto di Ricerca RR 43/2006, Dipartimento di Informatica - Università di Verona*, 2006.

W. Freeman J. Yedidia and Y. Weiss. Generalized belief propagation. In *Advances in Neural Information Processing Systems*, 2000.

R. Kaucic, R. Hartley, and N. Dano. Plane-based projective reconstruction. In *International Conference on Computer Vision*. IEEE Computer Society, 2001.

Z. Khan, T. Balch, and F. Dellaert. MCMC-based particle filtering for tracking a variable number of interacting targets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1805–1918, November 2005.

Ross Kindermann and J. Laurie Snell. *Markov Random Fields and Their Applications*. Americal Mathematical Society, 1980.

A. Klaus, M. Sormann, and K. Karner. Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. In *International Conference on Pattern Recognition*, 2006.

D. Koller, U. Lerner, and D. Angelov. A general algorithm for approximate inference and its application to hybrid Bayes nets. In *Conference on Uncertainty in Artificial Intelligence 15*, pages 324–333, 1999.

V. Kolmogorov and R. Zabih. Computing visual correspondence with occlusions using graph cuts. In *International Conference on Computer Vision*, 2001.

V. Kolmogorov and R. Zabih. Multi-camera scene reconstruction via graph cuts. In *European Conference on Computer Vision*, 2002.

Dan Kong and Hai Tao. A method for learning matching errors in stereo computation. In *BMVC*, 2004.

John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*, pages 282–289. Morgan Kaufmann, San Francisco, CA, 2001.

Stan Z. Li. *Markov Random Field Modeling in Computer Vision*. Springer-Vewrlag, 1995.

Zhifeng Liu and Reinhard Klette. Performance evaluation of stereo and motion analysis on rectified image sequences. In *CITR-TR-207, The University of Auckland, ISSN 1178-3524*, 2007.

M.I.A. Lourakis and A.A. Argyros. The design and implementation of a generic sparse bundle adjustment software package based on the levenberg-marquardt algorithm. Technical Report 340, Institute of Computer Science - FORTH, Heraklion, Crete, Greece, Aug. 2004. Available from `http://www.ics.forth.gr/~lourakis/sba`.

David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

N. Metropolis and S. Ulam. The monte carlo method. *J. Amer. Statist. Assoc*, 1949.

D. Min and K. Sohn. Edge-preserving simultaneous joint motion disparity estimation. In *International Conference on Pattern Recognition*, 2006.

R. M. Neal, M. J. Beal, and S. T. Roweis. Inferring state sequences for non-linear systems with embedded hidden Markov models. In *Advances in Neural Information Processing Systems 16*, 2003.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufman, San Mateo, 1988.

R. Yang H. Stewnius Q. Yang, L. Wang and D. Nistr. Stereo matching with color-weighted correlation, hierarchical belief propagation and occlusion handling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(3), 2008.

Ariadna Quattoni, Sybor Wang, Louis-Philippe Morency, Michael Collins, and Trevor Darrell. Hidden conditional random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(10):1848–1852, October 2007.

J. Cryer R. Zhang, P. Tsai and M. Shah. Shape from shading: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 690–706, 1999.

Stefan Roth and Michael J. Black. Fields of experts: A framework for learning image priors. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005.

J. Malik S. Belongie and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002.

D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 2002.

D. Scharstein, R. Szeliski, and R. Zabih. http://vision.middlebury.edu/stereo/, 2001.

Daniel Scharstein and Chris Pal. Learning conditional random fields for stereo. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.

J.P. Lewis Stefan Roth Michael J. Black Simon Baker, Daniel Scharstein and Richard Szeliski. A database and evaluation methodology for optical flow. Technical Report MSR-TR-2009-179, December 2009.

N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3D. In *ACM Transactions on Graphics*, pages 25(3):835–846, August 2006.

Jian Sun, Nan-Ning Zheng, and Heung-Yeung Shum. Stereo matching using belief propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(7):787–800, 2003. ISSN 0162-8828. doi: http://dx.doi.org/10.1109/TPAMI.2003.1206509.

B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment - a modern synthesis. In *Vision Algorithms'99*, 1999.

Hoang Trinh and David McAllester. Unsupervised learning of stereo vision with monocular cues. In *British Machine Vision Conference*, 2009.

Hoang Trinh and David McAllester. Structure and motion from road-driving stereo sequences. In *IEEE Workshop on 3D Information Extraction for Video Analysis and Mining - CVPR 2010*, 2010.

R. van der Merwe, N. de Freitas, A. Doucet, and E. Wan. The unscented particle filter. In *Advances in Neural Information Processing Systems 13*, December 2001.

Z. Wang and Z. Zheng. A region based stereo matching algorithm using cooperative optimization. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.

O. Veksler Y. Boykov and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1222–1239, 2001.

Li Zhang and Steven M. Seitz. Estimating optimal parameters for mrf stereo from a single image pair. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(2), 2007. based on "Parameter Estimation for MRF Stereo", CVPR 2005.

Y. Zhang and C. Kambhamettu. On 3d scene flow and structure estimation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2001.