A Bayesian Framework for Panoramic Imaging of Complex Scenes

by

Alan Brunton

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the MCS degree in
Computer Science

School of Information Technology and Engineering
Faculty of Engineering
University of Ottawa

Abstract

This thesis presents a Bayesian framework for generating a panoramic image of a scene from a set of images, where there is only a small amount of overlap between adjacent images. Dense correspondence is computed using loopy belief propagation on a pair-wise Markov random field, and used to resample and blend the input images to remove artifacts in overlapping regions and seams along the overlap boundaries.

Bayesian approaches have been used extensively in vision and imaging, and involve computing an observational likelihood from the input images and imposing a priori constraints. Photoconsistency or matching cost computed from the images is used as the likelihood in this thesis. The primary contribution of this thesis is the use of and efficient belief propagation algorithm to yield the piecewise smooth resampling of the input images with the highest probability of not producing artifacts or seams.

Acknowledgements

I would like to thank my supervisors, Dr. Gerhard Roth, Dr. Eric Dubois and Dr. Chang Shu, for their advice and support throughout my term of study.

I would also like to thank the National Research Council of Canada for the use of their facilities and equipment, and the University of Ottawa and NSERC for funding my studies through the NAVIRE project on virtual navigation in real environments. I would like to thank the other students and professors involved in the NAVIRE project for sharing their thoughts and ideas.

Dr. Roth encouraged me to pursue graduate studies and was instrumental in securing this opportunity for me. His advice and insight on computer vision and image-based rendering has helped me immensely in understanding these fields. His ability to lighten the mood and his eagerness to discuss any and all subjects made my time at the NRC more enjoyable.

Dr. Dubois has been very accessible in dealing with funding and other administrative matters for my studies. He has also provided important reality checks when I tried to make problems more difficult than they needed to be. He has given helpful ideas for algorithms and literature tips on Markov random fields.

Dr. Shu has, willingly and without reproach, expertly compensated for my tendency to dive into a task without first consulting the literature. He introduced to me probabilistic, and in particular Bayesian, frameworks for vision and imaging, including the belief propagation algorithm. His teaching and advice on these topics and others has proved invaluable, as has his counsel and perspective on matters in general.

I am very grateful to have had an opportunity to work with each of my supervisors.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This thesis presents a Bayesian framework for generating panoramic images of scenes with arbitrary geometry captured by either multiple images in a radial configuration with respect to the scene.

Bayes' theorem, or Bayes' rule, states that the probability of X given Y is proportional to the product of the prior probability of X and the probability of Y given X: $P\left(X|Y\right) \propto P\left(X\right) P\left(Y|X\right)$ [1]. $P\left(X\right)$ is the prior on X in that it is independent of Y. This leads to a probabilistic framework for solving numerical problems. By computing the probability of observing some event $Y = y$ given some explanation X, or *observational likelihood* $p\left(Y = y|X = x\right)$, and multiplying the result by the prior probability $p\left(X = x\right)$ one can compute the probability of the explanation, given the evidence (observation) at hand.

This framework is used to compute a panoramic image by dense correspondence among the input images. The observation event is the set of input images, and the observational likelihood is a Gaussian distribution with respect to a photoconsistency matching cost, which is the dissimilarity of colours between two or more pixels (or patches of pixels). The prior distribution favors a piecewise smooth correspondence between the panorama and the input images.

Section 1.1 provides a motivation for using a Bayesian approach to dense correspondence for panoramic image generation. Section 1.2 states the problem of this thesis both intuitively and formally, and Section 1.3 describes the structure of the remainder of this thesis.

1.1 Motivation

The task of creating a panoramic image from a set of planar images is a well-studied subproblem of the general task of image-based rendering. Typically, it is accomplished by constraining the camera motion, eg. to be purely rotational [31]; by assuming the scene geometry allows the camera motion to be approximated by planar warping, eg. when the scene is at very large distance relative to the camera motion [35]; by assuming the geometry is locally planar [33, 41]; or by computing dense correspondence or registration to explicitly compensate for parallax between overlapping input images [34, 37, 38, 48].

Parallax occurs when two input images "look" at the same scene point from different directions. That is, the pixels in the two images that correspond to the 3D scene point in question correspond to 3D rays in scene space that intersect at the scene point, but are not parallel; they do not have the same origin. For this to happen, the two input images must not share a common center of projection, i.e. they must have been captured at different positions in the scene. Further, the scene point must be sufficiently near. As the distance or depth of the scene point from the images' centers of projection becomes much larger than the distance between them, the rays of the corresponding pixels become parallel in the limit.

When parallax does occur, local compensation is needed; planar projective transformation between image planes will, on its own, result in "ghosting" or "echo" artifacts where nearby scene points appear blurred or repeated in the panorama [34]. Dense or quasi-dense correspondence techniques [34, 38, 39, 43, 44, 47, 48] match individual pixels or small groups of pixels across two or more input images to prevent such artifacts.

To generate a full panorama, one needs either many input images or a wide baseline between input images. Wide baseline configurations often result in only a small amount of overlap between adjacent input images and regions of the panorama seen only from a single image. With one exception ([48]) the above dense correspondence techniques produce satisfactory results only when the input images are densely spaced, such that there is significant, at least fifty percent, overlap between adjacent input images [48]. When there is less overlap between input images, there may be significant portions of the panorama viewed by only one input image. Applying local dense correspondence techniques, or simple stereo fusion, in such cases results in seams or discontinuities along the boundaries between regions of the panorama covered by only one input image and regions of overlapping input images. The work of Kang et al. [48] dealt with avoiding such seams while performing parallax compensation in overlapping regions for a multi-sensor configuration with approximately ten percent overlap between adjacent input images.

The goal of this thesis is to develop a flexible method for generating panoramic imagery for arbitrary scene geometry and arbitrary input image configurations: handling both narrow and wide baseline, large and small overlap, configurations; avoiding ghosting artifacts where input images overlap and avoiding discontinuities between overlapping and non-overlapping regions. The primary panoramic image format considered here is the cubic panorama, eg. Figure 1.1, which samples the plenoptic function across the surface of a cube. The plenoptic function, which is defined formally in Chapter 2, denotes the incident light in a given direction at a given location in a scene.

Bayesian methods are powerful tools for inferring or estimating unobservable quantities of a scene, given some observations about the scene. In the case of generating an immersive panoramic image of a scene from a set of planar input images of that scene, one would like to estimate the most probable dense correspondence between the images, given an observation extracted from the images. To represent the correspondences of a pixel in the panoramic image to be generated with the input images, the depth from the panoramic center of projection is used. The estimated panoramic depth map is used to resample the input images into a seamless and artifact free panoramic image. Photoconsistency is used as the observational likelihood of each of a set of candidate depths. The primary contribution of this thesis is the use of efficient hierarchical belief propagation [13] to impose a Markov smoothing prior to find the resampling depths most probable to not produce artifacts or seams, given the photoconsistency observations made from the images [60]. A further contribution of this thesis is an implementation of Felzenzwalb's hierarchical belief propagation for graphics processing units (GPUs) [61].

1.2 Problem Statement

This thesis seeks to produce a complete and seamless plenoptic sampling of a complex scene by fusing a set of incomplete plenoptic samplings from nearby locations. Section 2.2 explains how this is a sub-problem of image-based rendering as defined by McMillan and Bishop [32]. More plainly, the goal is to generate a panoramic image of a scene from a set of planar images of the scene. Let a complex scene be loosely defined as one in which objects may be sufficiently near to the capture locations so as to cause parallax between images.

It is assumed that the scene is static for the set of input images; that the images were captured synchronously, or simply that the scene did not change from the time at which the first image was captured to the time the last image was captured. It is further assumed that the positions and orientations from which the input images were captured were previously determined,

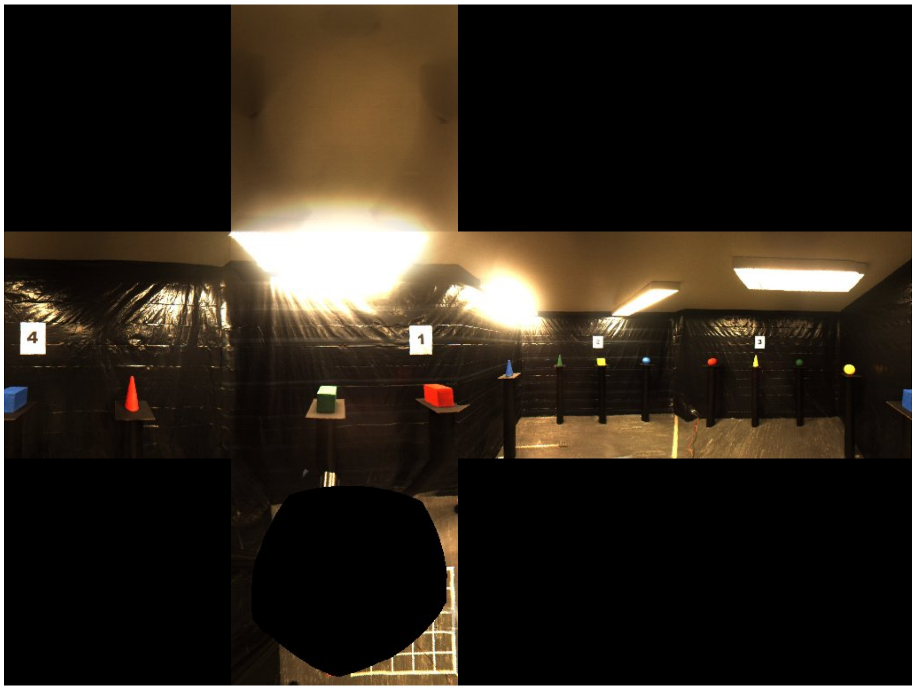


Figure 1.1: Flattened or unfolded cubic panorama.

relative to some common coordinate frame. No assumptions are made about the actual positions and orientations, just that they are known. The only assumption made about the geometry of the scene is that it is piecewise smooth.

The problem is formulated as one of computing dense correspondence between the panoramic image and the input images. Denote the panoramic image by Π and the set of input images by $\mathcal{I} = \{\mathcal{I}_j\}$ for $j = 0, 1, \ldots, n-1$, where n is the number of input images. Π is a parameterization or sampling lattice where each pixel $p \in \Pi$ represents a sampling direction from a common centre of projection. The task is then to compute, for every pixel $p \in \Pi$, the corresponding pixel $\hat{p}_j \in \mathcal{I}_j$, if such a correspondence exists for $\mathcal{I}_j$. Since a panoramic pixel p may correspond to pixels in multiple input images, it is more compact to represent the correspondence of a panoramic pixel with the input images as the depth from the panorama's center of projection rather than as a set of pixel coordinate pairs. The problem of dense correspondence thus becomes that of determining the depth at each panoramic pixel, $z(p)$.

This thesis uses Bayesian belief propagation to estimate the *maximum a posteriori (MAP)* depth at each panoramic pixel, given the set of input images:

$$z^{MAP}(p) = \arg\max_{z(p)} P\left(z(p) \mid \mathcal{I}\right) \qquad (1.1)$$

where the posterior distribution is computed according to Bayes' theorem

$$P\left(z\left(p\right)|\mathcal{I}\right) \propto P\left(z\left(p\right)\right)P\left(\mathcal{I}|z\left(p\right)\right) \tag{1.2}$$

where $P\left(z\left(p\right)\right)$ denotes prior probability of the depth $z\left(p\right)$ at pixel p, and $P\left(\mathcal{I}|z\left(p\right)\right)$ denotes the observational likelihood that the set of input images would be generated, given the depth $z\left(p\right)$ was assumed to be correct. $P\left(z\left(p\right)\right)$ is a prior distribution in that it does not consider any observation of the scene itself, and constrains the scene geometry to be piecewise smooth. This prior is imposed by a Markov random field. The observational likelihood is a function of the *photoconsistency* of the input images for the given depth.

1.3 Thesis Organization

Chapter 2 reviews the theoretical background and literature of direct relevance to this thesis. Planar imaging and image sensor models are reviewed, followed by panoramic imaging and image-based rendering and a review of Markov random fields applied to vision and imaging problems. These topics are reviewed to illustrate by what concepts they relate to this thesis. The review concludes with a more extensive discussion of Bayesian belief propagation and its application to vision and imaging.

Chapter 3 describes how loopy belief propagation is efficiently applied to panoramic imaging. The image sampling and calculation of the photoconsistency observation, energy minimization by belief propagation, and resampling of the input images are discussed in turn.

Chapter 4 describes a programmable graphics hardware implementation of the panorama generation algorithm described in Chapter 3. A review of other vision and imaging applications, and other general purpose computation, implemented in graphics hardware is followed by a description of the implementation of the different aspects of the algorithm.

Chapter 5 discusses the experimental setup and results of this thesis, including the hardware used, sample panoramas and input image sets, and running time comparisons to validate the implementation technique. Chapter 6 draws some conclusions from the results of Chapter 5, and discusses possibilities for future work.

Chapter 2

Background and Literature Review

This thesis touches on many topics within computer vision, imaging and computer graphics. This review describes the connections between the different techniques used in this thesis, and research to which it is related. Fundamental to these topics are pin-hole projection and image-sensor models, so this review begins with these concepts in Section 2.1.

After reviewing the basics of planar imaging, Section 2.2 discusses panoramic imaging and image-based rendering. Specifically, the discussion focuses on panoramic image formats and different ways of capturing panoramas using image-based rendering techniques.

The goal of the first two sections of review is to describe the important concepts and research under these headings as they relate to this thesis. Thus, it will not include extensive surveys in these areas.

A similar review follows on the use of Markov random fields (MRFs), or Markov networks, in computer vision and image processing. Section 2.3 also contains a detailed description of the Markov model used in this thesis.

Finally, Section 2.4 describes Bayesian belief propagation and its application to computer vision and image processing. Belief propagation (BP) is described on both Bayesian networks and Markov networks, and on both singly-connected networks and networks with loops (loopy belief propagation). The use of belief propagation in vision and imaging is reviewed as well.

A core concept connecting planar imaging, panoramic imaging and IBR is the *plenoptic function*. It is a 7D function describing an intensity distribution of light in terms of viewing position (V_x, V_y, V_z), viewing direction (θ, ϕ), time and wavelength [30]

$$\mathcal{I} = \mathcal{P}\left(V_x, V_y, V_z, \theta, \phi, t, \lambda\right) \tag{2.1}$$

making it very difficult to evaluate in general. However, an image or a video frame is captured at the same time t for all pixels, so one only need consider that parameter when considering multiple (asynchronous) images of a dynamic scene. The wavelength λ is usually parameterized by capturing multiple channels, for example red, green and blue responses for each pixel:

$$\mathcal{I}_{R,t} = \mathcal{P}\left(V_x, V_y, V_z, \theta, \phi, t, \lambda_R\right)$$

and similarly for green and blue, where λ_R is the mean wavelength for red light.

The plenoptic function is central to panoramic imaging and image-based rendering. A panoramic image is a complete sampling of the plenoptic function at a particular position and time (and a particular, eg. RGB, parameterization of the wavelength). That is, a panorama samples the plenoptic function over the complete range of the direction parameters θ and ϕ. In practice, a panorama usually samples the plenoptic function a nearly complete range of θ and ϕ. Image-based rendering deals with interpolating and fusing discrete, incomplete samplings of the plenoptic function, images, into continuous or complete plenoptic samplings.

2.1 Pin-hole Projection and Geometric Camera Models

Panoramic imaging is naturally related to imaging under the standard pin-hole camera model. Cubic panoramas in particular consist of six pin-hole projected images. In the pin-hole model, the focal length is the distance f of the image or focal plane from the center of projection or focal point (the "pin-hole" through which the light corresponding to each pixel passes). For simplicity, let the center of projection be the origin of the 3D coordinate system, known as the camera or sensor coordinate frame, locate the focal plane at $Z = f$, parallel to the XY-plane, and locate the image plane at $Z = -f$, parallel to the XY-plane. This is shown as an orthogonal diagram in Figure 2.1, with the focal and image planes in the "physical image frame" and the "conceptual image frame", respectively. Physically, the focal plane must be on the opposite side of the pin-hole, or lens in a real sensor, from any point seen in the image. Conceptually, however, it is easier to think of the image plane in front of the focal point.

The 3D point P is projected onto the image plane by dividing its components by its depth d_P, giving *normalized image coordinates*, and multiplying by the focal length f to give *image coordinates* $[x, y]$. That is, $x = \left(\frac{f}{d_P}\right) X$, and $x = \left(\frac{f}{d_P}\right) X$. A pixel p is the image element at coordinates $[u, v]$ from the top-left of the image. The optical center, o, is the point at pixel coordinates (u_0, v_0) which is the orthogonal projection of the center of projection onto the image plane.

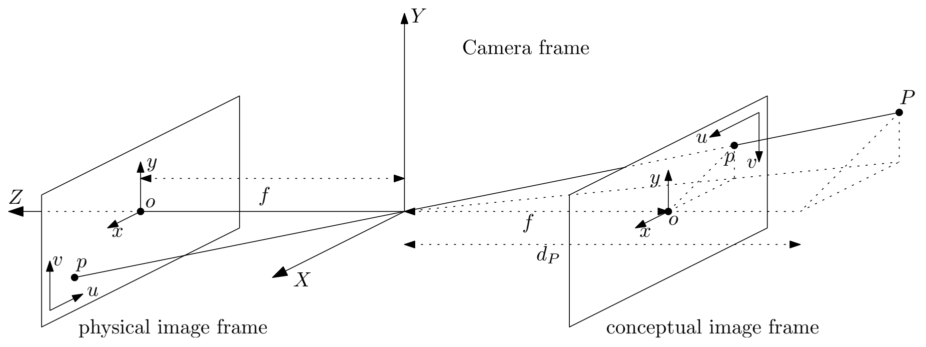


Figure 2.1: The pinhole camera model.

In real sensors, the light is not focused by a pin-hole, but by a lens. The lens may distort the projection of light from the ideal pin-hole model shown in Figure 2.1. Calibrating this distortion, along with the focal length and optical center, is crucial in performing accurate projection into real images, and is known as *intrinsic calibration*. Calibrating the position and orientation of the camera, i.e. the rigid body transformation of the camera coordinate frame, with respect to some external coordinate system is known as *extrinsic calibration*. The most commonly used calibration techniques are that of Tsai [55] and Zhang [56]. The technique of Tsai uses known points in a plane or in a known 3D configuration undergoing a precisely known translation, whereas the more flexible approach of Zhang exploits the homographic (2D projective) transformation between a planar pattern and its image.

The perspective projection, dividing camera coordinates by depth then multiplying by the focal length, can be considered multiplication by a scale factor. Hence, the projection of 3D camera coordinates onto 2D pixel coordinates can be modeled with the matrix [56]

$$A = \begin{bmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

where $f_x = s_x f$ and $f_y = s_y f$ are the focal length multiplied by the horizontal and vertical pixel sizes, s_x and s_y, respectively. The parameter γ represents the skewness of the image axes. The transformation of a point $\mathbf{M}$ in a coordinate system other than camera coordinates, call it the world coordinate frame, to the camera frame and into the images is given by

$$s\tilde{\mathbf{m}} = A\,[R\,|\mathbf{t}]\,\tilde{\mathbf{M}} \tag{2.3}$$

where s is the aforementioned scale factor, $\mathbf{m}$ is the 2D image of $\mathbf{M}$, $\tilde{\mathbf{m}}$ and $\tilde{\mathbf{M}}$ are the respective points augmented in homogeneous coordinates, and R and $\mathbf{t}$ are the rotation and translation of the world frame with respect to the camera frame, respectively [56].

Two kinds of lens distortion will be considered here: radial and tangential. When dealing with radial distortion, the difference between the ideal pin-hole projection $[x, y]$ and the real (distorted) image point $[x_d, y_d]$ is a function of the radius of the ideal projection from the optical center of the image. Using two radial distortion coefficients, k_1 and k_2, the radial distortion is [57, 58]

$$\delta_r(x) = x\left(k_1 r^2 + k_2 r^4\right) \tag{2.4}$$

$$\delta_r(y) = y\left(k_1 r^2 + k_2 r^4\right) \tag{2.5}$$

where $r^2 = x^2 + y^2$. For most cameras only radial distortion needs to considered, in which case the real (distorted) image coordinates are [57, 58]

$$x_d = x + \delta_r(x) \tag{2.6}$$

$$y_d = y + \delta_r(y)\,. \tag{2.7}$$

However, in systems using wide-angle lenses it can be necessary to account for tangential distortion too. Tangential distortion with coefficients p_1 and p_2 can be computed as follows [58]

$$\delta_t(x) = 2p_1 xy + p_2\left(r^2 + 2x^2\right) \tag{2.8}$$

$$\delta_t(y) = p_1\left(r^2 + 2y^2\right) + 2p_2 xy \tag{2.9}$$

and the distorted image coordinates become [58]

$$x_d = x + \delta_r(x) + \delta_t(x) \tag{2.10}$$

$$y_d = y + \delta_r(y) + \delta_t(y)\,. \tag{2.11}$$

The above gives the geometric relationship between a planar image and the plenoptic function and provides the mathematical basis for using multiple images to construct a complete plenoptic function. The image is a sampling of the plenoptic function positioned at its center of projection, in a set of directions that map onto the image plane under pin-hole projection and some distortion. Multiple images can be fused into a complete plenoptic sampling by mapping the images onto a panoramic manifold and averaging the projected plenoptic samples.

2.2 Panoramic Imaging and Image-Based Rendering

In contrast to the planar case, a panoramic image samples the plenoptic function in all, or nearly all, directions from a center of projection. Clearly this cannot be done by projecting onto a single plane. Instead, the plenoptic function is measured across other surfaces, e.g. cylindrical, spherical, cubic or octahedral. Whereas a planar image is parameterized by (u, v), a cylindrical panorama measures the plenoptic function at pixel locations (θ, v), where $\theta \in [0, 2\pi)$ is the azimuth, or angle about the vertical axis, and v is the Cartesian ordinate along the vertical axis (as in Section 2.1). The radius of the cylinder is the effective focal length of a cylindrical panorama. The ends of the cylinder are often ignored.

A spherical panorama, eg. Figure 2.2, is parameterized by (θ, α), where θ is again the azimuth, and $\alpha \in [\pi/2, -\pi/2]$ is the altitude, or angle of elevation above of the horizontal plane. Since both parameters are angular, there is no equivalent to focal length for a spherical panorama, in contrast with cylindrical panoramas where the vertical parameter is parallel with the cylinder's axis.

A cubic panorama is comprised of six planar images, corresponding to the faces of the cube and sharing a common center of projection at the center of the cube. Hence, each face image is square, and has a focal length one half its width, giving horizontal and vertical fields of view of $\pi/2$. A pixel in a cubic panorama is then addressed by three parameters (i, u, v), where $i \in \{0, 1, \ldots, 5\}$ is the face index, and (u, v) are as in Section 2.1.

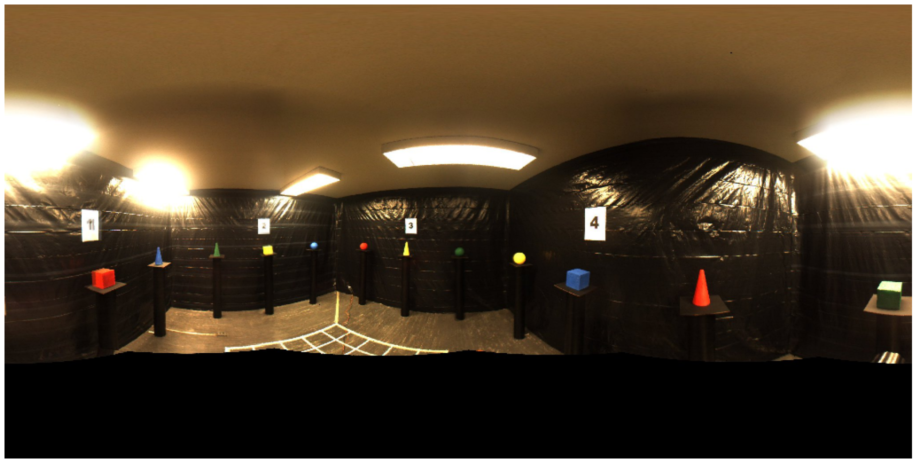


Figure 2.2: Spherical panorama.

Panoramic images cannot be captured by a camera modeled by pin-hole projection plus distortion. One can capture panoramas using such a sensor in conjunction with a special lens or a parabolic mirror, but this thesis will consider only capture methods involving multiple pin-hole projected images in configurations to cover the panoramic field of view. This includes capturing multiple views of what is assumed to be a static scene with the same camera, and capturing multiple images simultaneously with a multi-sensor system. Such techniques hold a major resolution advantage over mirror- or lens-based solutions, which project the entire scene onto the pixels of a single sensor, losing detail and incurring severe distortion.

Fusing, stitching, or mosaicking a panoramic image from multiple views is a special case of image-based rendering, as defined by McMillan and Bishop [32]

> Given a set of discrete samples (complete or incomplete) from the plenoptic function, the goal of image-based rendering is to generate a continuous representation of that function.

Discrete samples of the plenoptic function are images, either panoramic or planar. A continuous representation is one that can be used to generate a view of the scene from a continuous set of viewpoints. Thus, taking multiple planar images in a radial configuration and generating a view in any direction from a viewpoint near or inside the radial configuration is a case of IBR.

Chen introduced QuickTime VR at *SIGGRAPH* 1995 [31]. To capture panoramas, a camera is placed on a tripod and images are captured at roughly equal intervals of rotation about the vertical axis. The purely rotational nature of the change in viewpoint from image to image simplifies the registration between image frames. Szeliski and Shum [33] use a 2D projective transformation, or homography, to map pixels to viewing directions in the scene for each of a set of input images captured by a hand-held camera. They use this registration model to estimate the focal length and convert the transformations to a three parameter rotation model. They overcome errors in the registration by registering the first image to the last image. The difference in the rotations is then converted to a quaternion and distributed evenly over the input images. This registration error, or misalignment, is also used to improve the estimate of the focal length of the camera that captured the input images.

In the hand-held case, some translation of the camera is unavoidable, and could result in significant parallax between images for objects that are sufficiently close to the camera. Shum and Szeliski [34] account for such local misalignments, which may be due to parallax, lens distortion or optical center calibration error, or object motion with a patch-based local alignment following their feature-based global alignment [33]. This removes so-called ghosting or echo artifacts. However, it does not address the issue of significant parallax between input images because it is based on matching over arbitrary local 2D motion. Other dense sampling techniques for panoramic mosaicking include manifold projection [37], multiple-center-of-projection images [38], and plane-sweep techniques [39, 40].

Sparse matching, or sampling, techniques include [41, 35]. Zhu et al. [41] triangulate feature matches between images to obtain depth at each match, triangulate the projection of the set of matches into the output mosaic, and assume the scene is planar between vertices. As with the dense sampling techniques, Zhu et al. assume a dense arrangement of images. Brown and Lowe [35] model the motion as planar warping computed from reliable, scale-invariant features [36] to extract one or multiple panoramas from a set of input images. Panoramas are identified by finding sets of images that contain matching features, then for each panorama the intrinsic camera parameters and capture poses are optimized to minimize error over the matching features. The panoramic mosaic is blended seamlessly using a multi-band technique wherein low frequencies are blended using a weighted average of the overlapping input images and high frequencies are taken from the image with the highest weight. This effectively hides the seams despite illumination changes from image to image.

Reducing the number of input images used to construct a panorama while maintaining equal panoramic field of view requires increasing the field of view of the input images, decreasing the amount of overlap between images, or both. This is a particularly desirable objective in the case of multi-sensor systems, which capture multiple images synchronously. Such a system, used for experiments in this thesis, is the Ladybug from Point Grey Research [63], which captures synchronously from six sensors. Five of the sensors point radially about the vertical and the sixth points upwards. Each of the sensors uses wide-angle lenses, and there is only about ten percent overlap between adjacent images. This configuration covers approximately three quarters of the full spherical panorama.

The small amount of overlap and the high lens distortion due to the wide-angle lenses make registering the images more difficult. Further, the sensors do not share a common center of projection; they are translated as well as rotated with respect to one another. So, while it is generally less pronounced than for a sequence captured with a hand-held camera, there is parallax between adjacent sensors for close objects, and ghosting or echo artifacts can occur in the regions of the panorama where the images overlap if simple alpha blending and feathering is used without adjusting the sampling position to account for parallax. Alpha blending refers to a rendering operation where the final colour of the destination pixel is a combination of its current colour and the colour of the incoming source pixel, based on their respective alpha values or weights. Feathering refers to gradually reducing the alpha for an input image close to the edge of the image. Calibration of the positions and orientations of the sensors with respect to a common coordinate frame provided by the manufacturer eliminates the need for registration. More detail on the Ladybug camera system, pictured in Figure 2.3, is given in Section 5.1.

Figure 2.3: The Ladybug multi-sensor camera system [63].

The dense sampling approaches discussed above generally do not satisfactorily compensate for parallax when the amount of overlap between input images is less than fifty percent [48]. Simply applying stereo fusion in this situation results in discontinuities at the borders of the overlap regions. To avoid ghosting or echo artifacts when rendering sequences captured with the Ladybug, Kang et al. developed an algorithm called *multi-perspective plane sweep (MPPS)* [48]. The MPPS algorithm operates on pairs of input images, performing stereo matching in the overlap region, but also varying the view point for each pixel-column or group of columns in the overlapping region of the rectified input images. This is in contrast to standard stereo fusion where the viewpoint for the mosaic is located at a fixed point. For example, if rectified input images $\hat{\mathcal{I}}_0$ and $\hat{\mathcal{I}}_1$ overlap for N columns of pixels $c_0, \ldots, c_{N-1}$, then the stereo correspondence for c_0 is computed relative to the center of projection of $\hat{\mathcal{I}}_0$. For column c_{N-1}, dense stereo is computed relative to the center of projection of $\hat{\mathcal{I}}_1$, and the viewpoint is varied linearly in between. The overlap is then blended by a weighted average of the colors at corresponding pixels in the two images, and then projected back into the rectified images, overwriting the corresponding pixels. The weights are computed as a linear function of the distance of the pixel from the two edges of the overlap region. Figures 2.5 and 2.6 [48] illustrate the results of single-viewpoint fusion compared with the MPPS, respectively, for a pair of image captured using the Ladybug system. Notice the discontinuities in Figure 2.5 are removed in Figure 2.6. Instead, the wood beam is slightly bent in the overlap region compared with the non-overlapping regions, but this is a far less noticeable artifact than either the discontinuities, or the ghosting in Figure 2.4.

The result is two images with different centers of projection, with the overlapping regions warped to account for parallax. The MPPS performs this task for each overlapping pair of radially-oriented images; the top sensor is a special case and the top image must be fused with each radial image separately. The total result is six images with distinct centers of projection, and with regions warped to account for parallax with adjacent images. Rendering involves standard planar mosaicking and blending of these parallax-warped images. Uyttendaele et al. [49] use the Ladybug and the MPPS and other techniques to generate interactive image-based walk-through sequences of real-world environments.

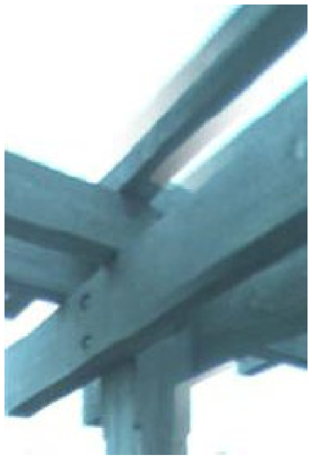


Figure 2.4: Ghosting artifact from using direct blending and feathering of overlapping Ladybug images [48].

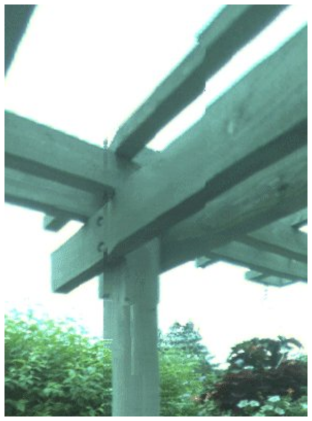


Figure 2.5: Discontinuities introduced by single-viewpoint stereo fusion in a small overlap setting [48].

The goal of this thesis is to compute a single, seamless panorama without ghosting or echo artifacts, with a single center of projection. While generating a panorama with a single

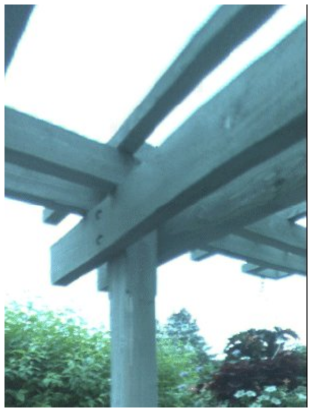


Figure 2.6: Ghosting in Figure 2.4 and discontinuities in Figure 2.5 are removed by the MPPS [48].

center of projection and then viewing it perspectively is equivalent to planar mosaicking of the parallax-warped images in the MPPS, this thesis is also motivated by the application of other computer vision algorithms to the resulting panoramas: eg. object detection, recognition and tracking, feature tracking and structure from motion, among others. Constraining the panoramic format to have a single center of projection avoids additional complications to the mathematics of these tasks, maintaining that similarity to their application to planar images. Applying such algorithms to the warped overlap images may be problematic, and resampling the images the extra time to generate the single-point-of-projection panorama will degrade image quality. While resampling may have an unremarkable impact for viewing, the effect on computer vision algorithms may yet be deleterious. Further, maintaining the high panoramic image quality gives more flexibility in how they are used.

To obtain a seamless and artifact-free panoramic image, in this thesis a global optimization algorithm is used within a Bayesian framework. Specifically, Bayesian belief propagation is used on a Markov network. The Markov network contains loops, so the optimization is not strictly global, however it is optimal over very large neighborhoods in the solution space. These techniques are used to obtain dense correspondence between input images, in the form of depth from the panorama's center of projection, from which to resample and blend the input images.

Bayesian techniques leverage Bayes' theorem: $P(X|Y) \propto P(X)\,P(Y|X)$. The posterior probability of some unknown quantity X given some observed quantity Y, $P(X|Y)$, is computed in terms of an observed likelihood, $P(Y|X)$ and a known, or approximated, prior probability on X, $P(X)$. Here, a photoconsistency observation is used as the likelihood and a smoothing prior is imposed by the Markov random field. In regions of the panorama covered by only one image, no photoconsistency observation can be made because colors from multiple images are needed to compute photoconsistency. This uncertainty is expressed by assigning a uniform distribution over depth in these regions.

Fitzgibbon et al. [14] use a photoconsistency likelihood within a Bayesian framework for IBR. They enumerate local maxima of the likelihood by first densely sampling in depth; they compute photoconsistency at 500 depths for each output pixel. Then they find local maxima of the likelihood in color space by running gradient descent from multiple (≈ 20) random starting points in color space, and then clustering the results to obtain several local maxima. This process reduces the optimization to a discrete labeling problem.

Instead of a smoothing prior, however, Fitzgibbon et al. use a texture prior. They extract a library of 5×5-pixel texture patches from the input images. The prior probability of an output pixel is then a Gaussian function of its distance to its nearest neighbor in the texture library.

In their survey of image-based representations, Shum et al. [42] describe a continuum of IBR techniques based on the degree to which they incorporate geometric information. They partition the continuum into three categories of IBR techniques: those that use no geometry, those that use implicit geometry, and those that use explicit geometry. This thesis would fall near view interpolation [43] on the continuum, between techniques that use no geometry, such as the mosaicking techniques described above or concentric mosaics [45], and techniques that use implicit geometry, such as view morphing [44]. For more IBR techniques, the interested reader is referred to the survey of Shum et al. [42].

2.3 Markov Random Fields for Vision and Imaging

Let $\mathcal{F}$ be a field of discrete random variables sampled at a lattice $\Lambda = (V, E)$, where V is the set of lattice sites with $|V| = S$, and E is the set of edges connecting them. Let each random variable $F_s \in \mathcal{F}$ for all $s \in V$ have L possible values or *labels* $f = 0, 1, \ldots, L-1$, and denote a configuration by the vector of length S, $\mathbf{f} = [f_s \in \{0, 1, \ldots, L-1\}; \forall\, s \in V]$, such that $\mathcal{F} = \mathbf{f} \leftrightarrow F_s = f_s \;\forall\, s \in V$. Let the set of all possible configurations be denoted $\mathcal{F}_S^L$. The field $\mathcal{F}$ is a Markov random field (MRF) if

$$P(\mathcal{F} = \mathbf{f}) > 0 \;\forall\, \mathbf{f} \in \mathcal{F}_S^L \tag{2.12}$$

and

$$P(F_s = f_s | \{F_r = f_r\} \;\forall\, r \in V, r \neq s) = P(F_s = f_s | \{F_r = f_r\} \;\forall\, r \in N(s)) \tag{2.13}$$

where $N(s) \subseteq \{r\} : r \in V; r \neq s$ is the neighborhood of s [2, 3, 6]. The first condition (2.12) is that all possible configurations have non-zero probability, however small, and the second (2.13) is Markov property itself, that the probability at a site s depends only on its Markov blanket, or neighborhood $N(s)$.

A Gibbs distribution on a field $\mathcal{F}$ is one of the form

$$P(\mathcal{F} = \mathbf{f}) = \frac{1}{Z} e^{-U(\mathbf{f})} \tag{2.14}$$

where Z is a normalization constant called the partition function and $U(\mathbf{f})$ is the energy of the configuration **f**. A *clique* is defined as subset $C \subseteq V$ such that every pair of distinct sites $s, r \in C$ are neighbors in Λ, i.e. $(s, r) \in E$. Notice that the neighborhood of a site is the union of the sites that share a clique with it: $N(s) = \{r \, \forall r : (s, r) \in E\}$. In this way, the edges represent statistical dependencies between the random variables in the Markov network. The energy function can be written

$$U(\mathbf{f}) = \sum_{C \in C_\Lambda} U_C(\mathbf{f}) \tag{2.15}$$

where $U_C(\mathbf{f})$ is the energy or cost contribution from each clique in the set C_Λ of all cliques in Λ for the configuration $\mathcal{F}$ [2, 6]. The clique energy is in fact a function of only the assignments or labelings $F_s = f_s$ for $s \in C$, and not the entire configuration. The Clifford-Hammersley theorem states that the field $\mathcal{F}$ is an MRF if and only if $\mathcal{F}$ can be described by a Gibbs distribution (2.14) [2, 3, 6].

From (2.15), the Gibbs distribution (2.14) can be rewritten as

$$P(\mathcal{F} = \mathbf{f}) = \frac{1}{Z} \prod_{C \in C_\Lambda} e^{-U_C(\mathbf{f})} = \frac{1}{Z} \prod_{C \in C\Lambda} \phi_C(\mathbf{f}) \tag{2.16}$$

which gives a probability distribution corresponding to the energy function (2.15), where ϕ_C are probability distributions called clique potentials. Note that minimizing (2.15) and maximizing (2.16) are equivalent.

Bayesian formulations are naturally well suited to vision and imaging problems: prior knowledge of the scene or image, and an observation (i.e. the input images, or a measurement computed from the input images) are used to estimate the posterior distribution of a property of the scene or image, given the observation. Bayes theorem states

$$P(X = x | Y = y) = \frac{P(Y = y | X = x) P(X = x)}{P(Y = y)} = P_L(Y = y | X = x) P(X = x) \tag{2.17}$$

where X is the unknown random process (the quantity of the scene or image one wishes to estimate), and Y is the observed process. The posterior probability of X given the observation Y

is the product of the (normalized) observational likelihood $P_L\left(Y=y|X=x\right)=\frac{P(Y=y|X=x)}{P(Y=y)}$ and the prior distribution on X, $P\left(X\right)$.

The probability distribution of an MRF is completely described by its clique potentials, so an MRF that models the posterior distribution contains cliques that model the prior distribution and cliques that represent the observational likelihood. This is accomplished by taking an MRF $\mathcal{F}_X$ that describes the prior distribution on the lattice $\Lambda_X=\left(V_X,E_X\right)$, and "coupling" it with an MRF $\mathcal{F}_{Y|X}$ that models the likelihood on $\Lambda_{Y|X}=\left(V_{Y|X},E_{Y|X}\right)$. That is, adding either edges, or sites and edges, to obtain $\mathcal{F}_{X|Y}$ on $\Lambda_{X|Y}=\left(V_{X|Y},E_{X|Y}\right)$ where $V_{X|Y}=V_X\cup V_{Y|X}$ and $E_{X|Y}=E_X\cup E_{Y|X}$.

The pre-eminent use of MRF models for imaging is the Bayesian restoration of images from Geman and Geman [2]. They use three coupled MRFs: the intensity process $\mathcal{F}$ on a pairwise or first-order lattice with sites Z_m corresponding to the image pixels; a binary line process $\mathcal{L}$ (representing edges or discontinuities in the image) on the dual of the pixel lattice, D_m; and they incorporate their observation field $\mathcal{G}$ through an image degradation model (blurring and noise) on a second-order lattice with the same sites, Z_m, as the intensity field. The line and intensity fields interact such that for $s,r\in Z_m$ and $d\in D_m$ between s and r, $U_{sr}\left(\mathbf{f}\right)=0$ if $L_d=1$. That is, if there is an edge or line between pixel sites s and r, then the clique energy is zero, regardless of the values of F_s and F_r. Otherwise $U_{sr}\left(\mathbf{f}\right)$ is non-zero representing the "bond" between sites r and s. They also use a four-state line process, with state (label) 0 indicating no edge, and the other states indicating the orientation of the edge.

The observation field is obtained without any additional sites, but by adding edges to create a second order neighborhood system on Z_m, such that the neighborhoods modeling the statistical dependencies of the observation field correspond to the neighborhoods used as input to the image degradation model. The intensity field is the input to the degradation model, which is then compared to the observed degraded image to obtain a cost for the cliques of $\mathcal{G}$.

Konrad and Dubois [5] note that a vector field can have Markov properties as well as a scalar field (the only difference is the dimensionality of the state space of the field), and use this to apply the line field-coupled model of Geman and Geman to the Bayesian estimation of motion vector fields. Since motion estimation requires at least two images, they use an observation model on the probability of next image g_{t+}, given the previous image g_{t-} and the estimated motion vector field $\mathbf{d}_t$: $p\left(\tilde{g}_{t+}|\mathbf{d}_t,l_t,g_{t-}\right)\approx e^{-U_g\left(\tilde{g}_{t+}|\mathbf{d}_t,g_{t-}\right)}$, where $\tilde{g}_{t+}$ is an interpolation of g_{t+}, l_t is the line field, and U_g is the Gibbs energy for the observation field. That is, they assume the observation is independent of the line process. Using a displacement model of coupled motion and motion discontinuity fields $\left(D_t,L_t\right)$ they decompose the distribution according to Bayes' theorem: $p\left(\mathbf{d}_t,l_t|g_{t-}\right)=p\left(\mathbf{d}_t|l_t,g_{t-}\right)p\left(l_t|g_{t-}\right)$. They note that the single image frame will not contribute much information to the motion field and use an energy function independent of g_{t-}: $p\left(\mathbf{d}_t|l_t, g_{t-}\right) \approx e^{-U_{\mathbf{d}}(\mathbf{d}_t|l_t)}$. They elegantly close the loop by modeling the line (motion discontinuity) field energy in terms of the previous frame: $U_l\left(l_t|g_{t-}\right)$, which is naturally computed in terms of the image gradient ∇g_{t-}.

Geiger and Girosi [3] use *mean field approximation* to average a surface field and a line field into a single field. They use this simplified model to reconstruct both intensity and depth surfaces. Black and Rangarajan [4] showed how the line field can be unified with outlier rejection through the use of robust statistics. They model the line process as an outlier process with respect to the smoothing prior, and similarly model outliers in the data (observation) field. They then show how these outlier processes can be approximated with robust estimators. More recently, Sun et al [12] use the robust estimation techniques of Black and Rangarajan to approximate a coupled disparity-line field for stereo.

Now consider an MRF $\mathcal{F}_\Pi$ on a pairwise or first-order lattice $\Lambda = (V, E)$ and a set Φ of potential functions on cliques in Λ. A first-order lattice, or graph, is one in which all cliques are 2-cliques, consisting of two sites. Further, let V be comprised of two subsets: sites $\{y_p\}$ represent observed random variables $Y = \{Y_p\}$, and sites $\{x_p\}$ represent hidden quantities $X = \{X_p\}$ of the scene, which we wish to infer, for every pixel $p \in \Pi$ where Π is the panoramic image, or face of the cubic panorama, to be generated. Let Π consist of P pixels. The values or labels of $\mathcal{F}_\Pi$ represent candidate locations from which to resample the input images into panoramic form. These sampling locations are stored in the form of distances, or depths, from the panorama's center of projection.

In this case a configuration $\mathbf{f} = [\mathbf{x}|G]$ for the P-element labeling vector $\mathbf{x}$ and the $P \times L$ observation matrix G, where $\mathbf{x} = [f_p]$, and $G = \left[\mathbf{y}_p\right]$ for all $p \in \Pi$. Each observation vector $\mathbf{y}_p$ is of length L, and consists of a photoconsistency observation (cost), computed from the input images, for each labeling $X_p = f_p \in \{0, 1, \ldots, L-1\}$. The assignment of a configuration to the field then follows $\mathcal{F}_\Pi = \mathbf{f} \leftrightarrow \left(X = \mathbf{x} \leftrightarrow X_p = f_p, Y = G \leftrightarrow Y_p = \mathbf{y}_p\right)\ \forall\, p \in \Pi$.

For every pixel p, $(x_p, y_p) \in E$. The hidden nodes $\{x\}$ are connected in a rectangular grid lattice such that $(x_p, x_q) \in E$ if and only if pixels p and q are non-diagonal neighbors in Π. Now, let the first potential function be the *local evidence*

$$\phi\left(x_p, y_p\right) = P\left(Y_p|X_p\right) \tag{2.18}$$

denoting the observational likelihood of Y_p (i.e. that the input images could have been observed) given that a particular labeling of X_p is the correct one. Let the second potential function be the *compatibility matrix*

$$\psi\left(x_p, x_q\right) = P\left(X_p|X_q\right) \tag{2.19}$$

denoting the Markov prior probability of labeling x_q given a labeling of x_p. The clique potentials are the corresponding probability density functions

$$\phi_p\left(f_p\right) = p\left(Y_p = \mathbf{y}_p | X_p = f_p\right) \tag{2.20}$$

denoting the likelihood of the local photoconsistency observation $\mathbf{y}_p$ given the labeling $X_p = f_p$, and

$$\psi_{pq}\left(f_p, f_q\right) = p\left(X_p = f_p | X_q = f_q\right) \tag{2.21}$$

denoting the prior probability of the labeling $X_p = f_p$ given the labeling $X_q = f_q$ for hidden neighbors x_p and x_q.

From (2.16), one can write the overall joint probability of all nodes in V as [2, 6, 11]

$$P\left(X, Y\right) = c_{X,Y} \prod_{(p,q)\in E} \psi\left(x_p, x_q\right) \prod_{p\in\Pi} \phi\left(x_p, y_p\right) \tag{2.22}$$

where $c_{X,Y}$ is a normalization constant and (p, q) is shorthand for the edge (x_p, x_q). One can also write the posterior probability of the hidden quantities X given the set of observations Y as

$$P\left(X|Y\right) = c_{X|Y} \prod_{(p,q)\in E} \psi\left(x_p, x_q\right) \prod_{p\in\Pi} \phi\left(x_p, y_p\right) \tag{2.23}$$

where $c_{X|Y} = \frac{c_{X,Y}}{P(Y)}$ is another normalization constant. Maximizing (2.22) is referred to as *marginalization* and results in the *minimum mean squared error* (MMSE) configuration $\mathbf{f}^{MMSE}$. Maximizing (2.23) results in the *maximum a posteriori* (MAP) configuration $\mathbf{f}^{MAP}$. For an individual variable X_p, the MMSE and MAP estimates are [1, 9]

$$f_p^{MMSE} = \sum_{f_p} f_p \sum_{q\in\Pi, q\neq p} p\left(X = \mathbf{x}, Y = G\right) \tag{2.24}$$

and

$$f_p^{MAP} = \arg\max_{f_p} \max_{q\in\Pi, q\neq p} p\left(X = \mathbf{x}, Y = G\right) \tag{2.25}$$

respectively.

From (2.14), (2.15), (2.16) and (2.23), the posterior probability of a configuration $X = \mathbf{x}$ given the observation $Y = G$ can be written

$$p\left(X = \mathbf{x}|Y = G\right) = c_{X|Y} e^{-U(\mathbf{x},G)} \tag{2.26}$$

for the energy function

$$U\left(\mathbf{x}, G\right) = \sum_{(p,q)\in E} U_{pq}\left(f_p, f_q\right) + \sum_{p\in\Pi} D_p\left(f_p\right) \tag{2.27}$$

where $U_{pq}(f_p, f_q)$ and $D_p(f_p)$ are the *discontinuity cost* and the *data cost*, respectively, such that $\psi_{pq}(f_p, f_q) = e^{-U_{pq}(f_p, f_q)}$ and $\phi_p(f_p) = e^{-D_p(f_p)}$. In fact, in this case the data cost can be taken as the photoconsistency observation itself: $D_p(f) = y_p(f)$ for $f = 0, 1, \ldots, L-1$, where $y_p(f)$ is the scalar function returning component f of the observation vector $\mathbf{y}_p$. Thus, the probability of a photoconsistency observation being made (from the input images), given a particular labeling f_p of x_p, is $p\left(Y_p = \mathbf{y}_p | X_p = f_p\right) = \phi_p(f_p) = e^{-y_p(f_p)}$. However, the data cost notation, $D_p(f)$, will be used throughout this thesis, in keeping with precedent notation [13].

The maximum a posteriori labeling $\mathbf{x}^{MAP}$ given the observation G can be computed by minimizing the energy, or "objective function", (2.27).

MRF formulations of early vision problems have existed for some time, exact evaluation of either the MMSE (2.24) or MAP (2.25) configuration is NP-hard, hence the need for approximation algorithms. Stochastic relaxation or simulated annealing approaches from statistical mechanics [2, 5] and mean-field approximation [3] among others have been used. Recently, considerable research has concentrated on the algorithms of *graph cuts* and *belief propagation*.

Both graph cuts (e.g. [28]) and belief propagation estimate the MAP configuration. Tappen and Freeman performed a comparison of both algorithms as applied to the stereo problem [16] for MAP estimation using identical MRF parameters and found that results were generally comparable between the algorithms, although graph cuts found lower energy configurations. However, these lower-energy solutions were not necessarily closer to the ground-truth energies. In fact, the ground-truth energy was often significantly higher than both graph cut or BP due to pixels that did not match any in the other image. They subsequently concluded that improvements in MRF stereo algorithms will come from improvements to the MRF model rather than improvements in the optimization algorithm.

All MRF formulations require the definition of parameters, such as weights for different potential functions, the values of which can dramatically affect the performance of the algorithm. The optimal values of these parameters often vary significantly from data set to data set, and often have to be hard-coded by trial and error. In their comparison of graph cuts and belief propagation, Tappen and Freeman use ten combinations of three parameters for each data set. Zhang and Seitz recently presented an expectation maximization (EM) approach to estimating optimal values for these parameters [18].

2.4 Bayesian Belief Propagation

The Bayesian belief propagation algorithm was originally introduced by Judea Pearl, in publications compiled into *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference* [1]. Belief propagation is run on a graph where the nodes represent random variables and the edges represent statistical dependencies between the variables the nodes represent. Beliefs, or probabilities of certain hypotheses given some evidence, are propagated through the network by local update rules.

Pearl originally considered belief propagation for *Bayesian networks*, which are directed acyclic graphs, or trees, and on singly connected Markov networks, or Markov trees. This was because in such settings the algorithm gives an exact solution, and because Pearl was considering the algorithm as applied to diagnostic and decision making systems, not imaging and vision. However, loopy belief propagation, has since had both empirical success and theoretical justification in finding approximately optimal solutions on Markov networks with cycles.

Section 2.4.1 describes the two variants of belief propagation as originally developed by Pearl for Bayesian networks. Section 2.4.2 then discusses how belief propagation works on singly-connected Markov networks, or Markov trees. Section 2.4.3 follows with a discussion of how belief propagation can be used as an approximation algorithm on MRFs that contain loops, such as regular image grid lattices, and its subsequent application to vision and imaging problems.

2.4.1 Belief Updating and Belief Revision on Bayesian Networks

Pearl developed two algorithms, *belief updating* and *belief revision*, which are now both considered variants of the same algorithm, belief propagation. First, consider belief updating on a *causal tree* (Bayes net), where an edge from node X to node Y explicitly represents the causal dependency of Y upon X. The belief distribution of node X can be computed if the following parameters are available:

1. The *causal* support of X, from its parent U: $\pi_X(u) = P\left(u|\mathbf{e}_X^+\right)$.
2. The *diagnostic* support of X, from the j-th child of X, Y_j: $\lambda_{Y_j}(x) = P\left(\mathbf{e}_{Y_j}^-|x\right)$.
3. The fixed conditional probability $P(x|u)$ relating X to its immediate parent U.

Here, $\mathbf{e}_X^-$ stands for the evidence contained in the tree rooted at X, and $\mathbf{e}_X^+$ stands for the evidence contained in the rest of the network. The lower-case x and u represent specific values of the variables X and U, respectively. Pearl also extended this algorithm to *causal polytrees*: Bayesian networks where node X may have multiple causal parents U_i.

The first quantity above, $\pi_X(u)$, is the message received by X from its parent U. The second quantity, $\lambda_{Y_j}(x)$, is the message received by X from its j-th child Y_j. The third quantity is a matrix expressing the compatibility of an assignment $X = x$ with the assignment to the immediate parent $U = u$, and is equivalent to the Markov prior potential, or compatibility matrix ψ, described in Section 2.3. Belief updating proceeds by performing three steps at each node in the network: belief updating, bottom-up propagation, and top-down propagation.

Belief updating for node X involves inspecting the parameters listed above and updating at X its belief

$$BEL(x) = \alpha\lambda(x)\pi(x) \tag{2.28}$$

where

$$\lambda(x) = \prod_j \lambda_{Y_j}(x) \tag{2.29}$$

and

$$\pi(x) = \sum_u P(x|u)\,\pi_X(u) \tag{2.30}$$

and α is a normalizing constant such that the beliefs sum to 1 over x.

In bottom-up propagation, node X computes, for each possible value u of U, the message

$$\lambda_X(u) = \sum_x \lambda(x)\,P(x|u) \tag{2.31}$$

and sends it to the parent node U.

In top-down propagation, X computes, for each possible value x of X, the message

$$\pi_{Y_j}(x) = \alpha\pi(x)\prod_{k\neq j}\lambda_{Y_k}(x) \tag{2.32}$$

and sends it to the child node Y_j.

These steps are run over the entire network. The update formulas (2.28), (2.31) and (2.32) preserve the probabilistic meaning of the messages and beliefs such that after at most one global iteration

$$\lambda_X(u) = P\left(\mathbf{e}_X^-|u\right), \tag{2.33}$$

$$\pi_Y(x) = P\left(x|\mathbf{e}_Y^+\right) \tag{2.34}$$

and

$$BEL(x) = P(x|\mathbf{e}) \tag{2.35}$$

where $\mathbf{e} = \mathbf{e}_X^+ \cup \mathbf{e}_X^-$ is the total evidence in the network [1, 9, 7]. That is, the belief at node X is the marginal probability of X, given all the available evidence. One global iteration simply means that all three steps are performed for every node in the network.

Belief revision is identical to belief updating, except that the summations in (2.30) and (2.31) are replaced by maximizations. Hence, the two algorithms became known as the "sum-product" and "max-product" variants of Bayesian belief propagation, respectively. The best explanation of the conceptual difference between the two comes directly from Pearl (p. 240):

> ...*belief updating*, i.e., assigning to each hypothesis in a network a degree of belief *BEL*, which changes smoothly and incrementally with each new item of evidence. This chapter applies Bayesian analysis and belief networks to another task: revision of belief commitments, where by *belief commitment* we mean the categorical but tentative acceptance of a subset of hypothesis that together constitute the most satisfactory explanation of the evidence at hand. In probabilistic terms, that task amounts to finding the most probable instantiation of all hypothesis variables, given the observed data. The resulting output is an optimal list of jointly accepted propositions, a list that may change abruptly as more evidence is obtained.

In the terminology of Section 2.3, the sum-product algorithm computes the MMSE configuration, or marginalization, of the network, while the max-product algorithm computes the MAP configuration of the network.

The probabilistic meanings of the messages and beliefs in belief revision, or max-product belief propagation, are slightly different than their counterparts in belief updating, or sum-product belief propagation. Let $\mathbf{W}$ stand for the set of all variables considered, including those in $\mathbf{e}$, and call any assignment $\mathbf{w}$ to the variables in $\mathbf{W}$ that is consistent with $\mathbf{e}$ an *extension* or *explanation* of $\mathbf{e}$. The most probable explanation (MPE) of $\mathbf{e}$ is the extension $\mathbf{W} = \mathbf{w}^*$ that maximizes the conditional probability $p(\mathbf{w}|\mathbf{e})$. The revised belief of an assignment $X = x$ is the probability of the most probable extension of $\mathbf{e}$ that is also consistent with the assignment $X = x$ [1]

$$BEL^*(x) = \max_{\mathbf{w}'_x} p(x, \mathbf{w}'_x|\mathbf{e}) \tag{2.36}$$

where $\mathbf{W}'_X = \mathbf{W} - X$; that is all other variables besides X. The MAP assignment for X is the one that maximizes (2.36). The messages obtain the following probabilistic meanings:

$$\lambda_X^*(u) = \max_{\mathbf{w}_{UX}^-} p\left(\mathbf{w}_{UX}^-|u\right) \tag{2.37}$$

is the conditional probability of the most probable assignment to all the variables in the subgraph rooted at X, $\mathbf{w}_{UX}^-$, given the assignment $U = u$. That is, it is the probability of the MPE of $\mathbf{e}_{UX}^{-}$, given the assignment $U = u$. The notation $\mathbf{e}_{UX}^{-}$ comes from the extension to causal polytrees, where the node X might have multiple causes $U_i : i = 1, 2, \ldots$, and is equivalent to the $\mathbf{e}_{X}^{-}$ notation for causal trees. And,

$$\pi_{Y}^{*}(x) = \max_{\mathbf{w}_{X'Y}^{+}} p\left(x, \mathbf{w}_{X'Y}^{+}\right) \tag{2.38}$$

is the maximum joint probability of the assignment $X = x$ and the assignment of all variables in the graph "above" Y, excluding X, $\mathbf{w}_{X'Y}^{+}$.

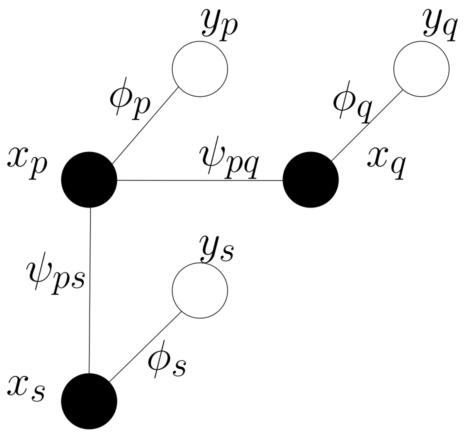


Figure 2.7: A singly connected MRF of three hidden nodes and three observed nodes.

2.4.2 Belief Propagation on Markov Trees

Now consider belief propagation on a singly connected Markov network, such as Figure 2.7, which shows a simple, singly connected MRF corresponding to a three-node segment of X and the corresponding nodes of Y from the MRF $\mathcal{F}_{\Pi}$ described in Section 2.3. The local message passing rules are in fact simpler for singly connected Markov networks than for Bayesian networks, because the edges in the Markov graph are undirected they do not imply any causal dependence, and hence there is no need for π-messages. Thus there is only one type of message, λ-messages.

By substituting into (2.22), we get the joint probability of the network as

$$P\left(X_p, X_q, X_s, Y_p, Y_q, Y_s\right) = \psi\left(x_q, x_p\right)\psi\left(x_s, x_p\right)\phi\left(x_p, y_p\right)\phi\left(x_q, y_q\right)\phi\left(x_s, y_s\right) \tag{2.39}$$

which can then be substituted into (2.24) or (2.25) to solve directly for f_p^{MMSE} or f_p^{MAP}, respectively. For example,

$$\begin{aligned} f_p^{MAP} &= \arg\max_{f_p}\max_{f_q}\max_{f_s}\psi_{qp}\left(f_q,f_p\right)\psi\left(f_s,f_p\right)\phi_p\left(f_p\right)\phi_q\left(f_q\right)\phi_s\left(f_s\right) \\ &= \arg\max_{f_p}\phi_p\left(f_p\right)\max_{f_q}\psi_{qp}\left(f_q,f_p\right)\phi_q\left(f_q\right)\max_{f_s}\psi_{sp}\left(f_s,f_p\right)\phi_s\left(f_s\right) \end{aligned} \tag{2.40}$$

is the MAP labeling of x_p in Figure 2.7.

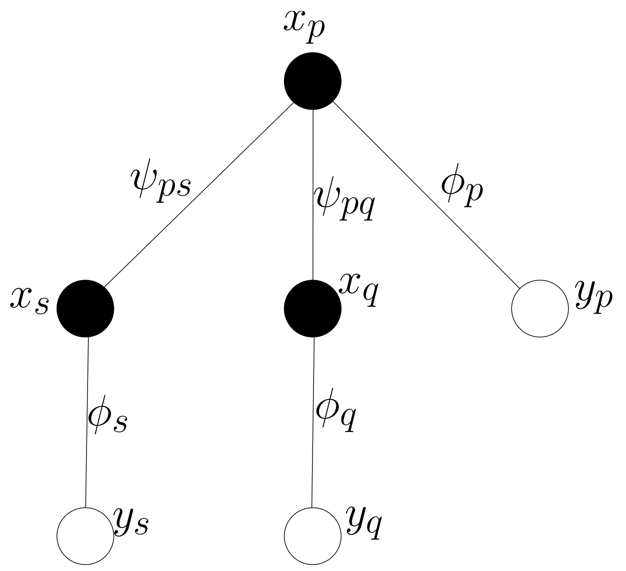


Figure 2.8: The Markov network in Figure 2.7 as a tree rooted at x_p.

However, as discussed in Section 2.3, the Markov networks required to solve most real-world problems are large enough to make exact calculations such as above infeasible. For example, the MRF $\mathcal{F}_\Pi$ has 2 million nodes in most experiments conducted for this thesis. Instead, approximation algorithms are used to estimate the marginal and posterior distributions. While (2.16) is true for general Markov networks, an additional factorization is valid for Markov chains and Markov trees, which leads to a message passing approximation algorithm that estimates either the marginals (2.22) or the posterior (2.23). Figure 2.8 shows the network in Figure 2.7 as a tree rooted at node x_p. Using the Markov properties, one can write the joint probability as [1, 9]

$$P\left(X_p,X_q,X_s,Y_p,Y_q,Y_s\right)=P\left(X_p\right)P\left(Y_p|X_p\right)P\left(X_q|X_p\right)P\left(Y_q|X_q\right)P\left(X_s|X_p\right)P\left(Y_s|X_s\right)$$

which is the same as (2.39), and is a chain rule factorization of the joint probability [1, 9]. One can rewrite (2.40) as

$$f_p^{MAP} = \arg\max_{f_p} p\left(X_p = f_p\right) m_{pp}\left(f_p\right) m_{qp}\left(f_p\right) m_{sp}\left(f_p\right)$$

where the message from y_p to x_p is $m_{pp}\left(f_p\right) = p\left(Y_p = \mathbf{y}_p | X_p = f_p\right) = \phi_p\left(f_p\right)$ because y_p is a leaf node [1]. Making a similar observation for y_q and y_s, The message from x_q to x_p is then

$$m_{qp}\left(f_p\right) = \max_{f_q} p\left(X_q = f_q | X_p = f_p\right) p\left(Y_q = \mathbf{y}_q | X_q = f_q\right) = \max_{f_q} \psi_{qp}\left(f_q, f_p\right) \phi_q\left(f_q\right)$$

and similarly for m_{sp}. Without single-node a priori potentials, one can take the prior on x_p by itself as the uniform distribution $p\left(X_p = f_p\right) = 1\ \forall f_p$. This allows f_p^{MAP} to be rewritten as

$$f_p^{MAP} = \arg\max_{f_p} \phi_p\left(f_p\right) m_{qp}\left(f_p\right) m_{sp}\left(f_p\right)$$

which evaluates to (2.40) upon substitution.

If one then considers the Markov network in Figure 2.7 as a tree rooted at node x_q, one can perform a similar factorization and write the MAP labeling $X_q = f_q^{MAP}$ as

$$f_q^{MAP} = \arg\max_{f_q} \phi_q\left(f_q\right) m_{pq}\left(f_q\right)$$

where

$$m_{pq}\left(f_q\right) = \max_{f_p} \phi_p\left(f_p\right) \psi_{pq}\left(f_p, f_q\right) m_{sp}\left(f_p\right)$$

which by substitution gives

$$f_q^{MAP} = \arg\max_{f_q} \phi_q\left(f_q\right) \max_{f_p} \psi_{pq}\left(f_p, f_q\right) \phi_p\left(f_p\right) \max_{f_s} \psi_{sp}\left(f_s, f_p\right) \phi_s\left(f_s\right)$$

which is analogous to (2.40).

This example shows a method for computing the MAP assignment for each variable in a singly connected network in terms of local message passing rules. These rules generalize to

$$f_p^{MAP} = \arg\max_{f_p} \phi_p\left(f_p\right) \prod_{q \in N(p)} m_{qp}\left(f_p\right) \tag{2.41}$$

for the MAP label of X_p and

$$m_{pq}\left(f_q\right) = \max_{f_p} \psi_{pq}\left(f_p, f_q\right) \phi_p\left(f_p\right) \prod_{s \in N(p) \backslash q} m_{sp}\left(f_p\right) \tag{2.42}$$

is the message from node x_p to x_q, i.e. when solving for f_q^{MAP}; and $N\left(p\right)$ denotes the hidden neighbors of x_p, i.e. the first-order neighborhood of x_p excluding y_p. Hence, $N\left(p\right) \backslash q$ is the first-order neighborhood of x_p excluding y_p and x_q, where the notation $S \backslash s$ indicates the set S excluding the element s.

One can also note that these messages are equivalent to the λ^*-messages from belief revision on a Bayesian network. Since the above passage derived the messages for the max-product algorithm using the Markov properties of the network, the derivation from diagnostic messages will be done for the sum-product algorithm (i.e. from the λ-messages). Without causal dependence between nodes, one can consider all messages as diagnostic messages and derive the message update formula from (2.31). The message $\lambda_{y_p}(f) = p\left(\mathbf{e}_{y_p}^{-} | X_p = f_p\right)$ is the diagnostic support of x_p from the child node y_p,

$$\lambda_{y_p}(X_p = f_p) = \sum_{\mathbf{y}_p} \lambda\left(\mathbf{y}_p\right) p\left(\mathbf{y}_p | f_p\right) = \sum_{\mathbf{y}_p} \phi_p(f_p) = \phi_p(f_p) \tag{2.43}$$

from (2.20), because $\mathbf{y}_p$ is fixed for a particular set of input images, and because $\lambda\left(\mathbf{y}_p\right) = 1$ since y_p is a leaf node [1].

The diagnostic support message sent to a hidden node x_q from its hidden neighbor x_p is

$$\lambda_{x_p}(X_q = f_q) = \sum_{f_p} p(X_p = f_p | X_q = f_q) \lambda(X_p = f_p).$$

By noting that $\psi_{pq}(f_p, f_q) = p(X_p = f_p | X_q = f_q)$ (2.21) and by extending (2.29) such that

$$\lambda(X_p = f_p) = \prod_{v:(x_p, v) \in E, v \neq x_q} \lambda_v(X_p = f_p)$$

where v denotes each neighbor of x_p other than x_q, one can write

$$\lambda_{x_p}(X_q = f_q) = \sum_{f_p} \psi_{pq}(f_p, f_q) \prod_{v:(x_p, v) \in E, v \neq x_q} \lambda_v(X_p = f_p)$$

which becomes

$$\lambda_{x_p}(X_q = f_q) = \sum_{f_p} \psi_{pq}(f_p, f_q) \phi_p(f_p) \prod_{s \in N(p) \backslash q} \lambda_{x_s}(X_p = f_p)$$

because $\lambda_{y_p}(X_p = f_p)$ is constant from (2.43). In the notation of (2.42) the message is written

$$m_{pq}(f_q) = \sum_{f_p} \psi_{pq}(f_p, f_q) \phi_p(f_p) \prod_{s \in N(p) \backslash q} m_{sp}(f_p) \tag{2.44}$$

and the MMSE estimate $X_p = f_p^{MMSE}$

$$f_p^{MMSE} = \sum_{f_p} f_p \phi_p(f_p) \prod_{q \in N(p)} m_{qp}(f_p) \tag{2.45}$$

is the expected value of X_p given all the nodes in the network.

It can be shown [1, 7], that for singly connected Markov networks after at most one global iteration of (2.42) for each node in the network, (2.41) gives the globally optimal MAP estimate.

2.4.3 Loopy Belief Propagation for Vision and Imaging

Both the sum-product and max-product algorithms are guaranteed to converge to exactly optimal solutions for graphs without loops, where loops are undirected cycles in the network (directed loops are by definition not permitted in Bayesian networks) [1]. The introduction of loops into the network may violate the conditional independence assumed among a node's ancestors [1] (p. 195):

> If we ignore the existence of loops and permit the nodes to continue communicating with each other as if the network were singly connected, messages may circulate indefinitely around these loops, and the process may not converge to a stable equilibrium. . . . Such oscillations do not normally occur in probabilistic networks because of the stochastic nature of the link matrices, which tend to bring all messages toward some stable equilibrium as time goes on. However, this asymptotic equilibrium . . . does not represent the posterior probabilities of all nodes in the network. The reason for this is simple: all of our propagation equations were based on some conditional independence assumptions that might be violated in multiply connected networks.

Nonetheless, both the sum-product and max-product (and equivalent min-sum) algorithms have been applied with success to pairwise image-lattice MRFs, which have many loops. This is the model used in this thesis. Examples include [9, 12, 13, 15]. Belief propagation on graphs with loops is known as loopy belief propagation, but is often simply called belief propagation.

On graphs with loops, belief propagation produces solutions that are optimal over large neighborhoods of the solution space. In [8], Weiss and Freeman proved that for an arbitrary graphical model, solutions produced by the max-product algorithm were maxima of *single loops and trees (SLT)* neighborhoods of the configuration. The SLT neighborhood of a configuration $\mathbf{x}^*$ includes all configurations $\mathbf{x}$ that can be obtained from $\mathbf{x}^*$ by selecting an arbitrary subset of nodes in the graph that consists of disconnected combinations of trees and single loops, and assigning arbitrary values to those nodes.

Intuitively, the singly connected restriction ensures that each piece of evidence is only counted (propagated) once per MAP or MMSE estimate. In a graph with loops, evidence will be counted multiple times [7] ("messages may circulate indefinitely around these loops"). However, Weiss showed that on a "balanced" loopy network, although evidence is counted more than once, all evidence is "double counted" equally [7]. An "unwrapped network" of a loopy network is a singly connected network, constructed from a loopy network by replicating nodes and edges, such that performing belief propagation on the unwrapped network is equivalent to performing it on the loopy network. The loopy network is considered balanced if one can construct an unwrapped network such that all nodes and edges are replicated an equal number of times.

Loopy belief propagation, in contrast to a singly connected network, requires more than one global iteration to converge. To reduce subscripts for the sake of clarity, the convention in the following will be that f represents the labeling of x_p and g represents the labeling of x_q. The message at iteration t from node x_p to x_q is the vector $\mathbf{m}_{pq}^t$ of length L, with each component being proportional to how likely node x_p "believes" it is that node x_q will have the label corresponding to that component. Let the scalar function $m_{pq}^t(g)$ denote the message vector component corresponding to label g. After T iterations, a belief vector $\mathbf{b}_p$ is computed for each hidden node x_p, where each component $b_p(f)$ represents the probability that $X_p = f$ given the evidence in the network. In the sum-product algorithm, messages are computed as follows [10, 11]

$$m_{pq}^t(g) = \kappa_1 \sum_f \psi_{pq}(f,g)\,\phi_p(f) \prod_{s \in N(p) \backslash q} m_{sp}^{t-1}(f)\,. \tag{2.46}$$

where κ_1 is a normalization constant. In the max-product algorithm messages are updated in the following way [12, 16, 8]

$$m_{pq}^t(g) = \kappa_2 \max_f \psi_{pq}(f,g)\,\phi_p(f) \prod_{s \in N(p) \backslash q} m_{sp}^{t-1}(f) \tag{2.47}$$

where κ_2 is a normalization constant. After T iterations, in both the sum-product and max-product algorithms, the beliefs are computed [12, 8, 10, 11]

$$b_p(f) = \kappa_3 \phi_p(f) \prod_{q \in N(p)} m_{qp}^{T-1}(f) \tag{2.48}$$

and the MAP and MMSE labeling for node x_p are

$$f_p^{MAP} = \arg\max_f b_p(f) \tag{2.49}$$

and

$$f_p^{MMSE} = \sum_f f b_p(f), \tag{2.50}$$

respectively.

The max-product algorithm can also be converted to the "min-sum" algorithm, which minimizes the energy function (2.27), by taking the negative of the logarithm of the message and belief probabilities. The min-sum algorithm is equivalent to the max-product algorithm; the messages and beliefs are computed in energy space, rather than probability space. Hence, the maximization of a product of probabilities in the max-product algorithm becomes a minimization of a sum of energies, and the message updates become [13]

$$m_{pq}^t(g) = \min_f \left(U_{pq}(f,g) + D_p(f) + \sum_{s \in N(p) \backslash q} m_{sp}^{t-1}(f) \right), \tag{2.51}$$

the beliefs become

$$b_p(f) = D_p(f) + \sum_{q \in N(p)} m_{qp}^{T-1}(f) \tag{2.52}$$

and the MAP label is found by

$$f_p^{MAP} = \arg\min_f b_p(f). \tag{2.53}$$

In this thesis, the min-sum formulation of the max-product loopy belief propagation (BP) algorithm is used to minimize the energy function (2.27) corresponding to the posterior distribution (2.23). This is done mostly for numerical concerns; multiplication of floating point numbers is generally less stable than addition. Also, the algorithm used in this thesis is based on that of Felzenzswalb and Huttenlocher [13], who use the min-sum version.

Freeman et al. [9] apply loopy BP to a number of low-level vision problems, primarily super-resolution. That is, estimating high-resolution detail from a single low-resolution image. (This is distinct from image-sequence super-resolution, which estimates high-resolution detail from multiple low-resolution images.) They define the nodes of their MRF to correspond to image patches: hidden nodes correspond to scene, or high-resolution, patches and observed nodes to patches in the low-resolution image. The scene patches are spaced so that they overlap, and the compatibilities are computed as the agreement between the pixels of neighboring patches which overlap. Specifically, they assume a Gaussian distribution on the difference of the overlapping pixels in the two patches. The candidate scene patches are obtained by finding the best L matches to the low-resolution image patches in a training database of image patches associated with high-resolution scene patches. The distance between each associated image patch and the observed image patch estimates the local evidence.

Yedidia et al. introduced Generalized Belief Propagation (GBP) [10, 11]. They construct messages between groups of nodes by building a hierarchy of regions (connected groups of nodes) by their intersections, such that a message is sent from a region to each of its direct sub-regions. A sub-region s of a region r is a direct sub-region if s is not also a sub-region of another sub-region s' of r. If r and r' are four-node regions (eg. on a pair-wise lattice) that overlap in the region $s = \{u, v\}$ where u and v are nodes in the lattice, then messages will be sent from both r and r' to s. On a pair-wise lattice, r and r' are straightforwardly constructed such that u and v are neighbors. This could be considered a way of "unwrapping" or "loosening" the loops in the network in terms of the way the messages are passed, and the "significant" improvement over standard belief propagation for 2D pair-wise lattices [11] is certainly related to the theoretical results of Weiss [7], and Weiss and Freeman [8].

Since BP is an iterative algorithm, improving its efficiency primarily involves finding ways to either reduce the number of iterations needed or to reduce the amount of computation required for each iteration. Sun et al. [12], in applying loopy BP to the dense stereo problem, achieve a speed-up in the propagation step by about 30-60 percent by observing that each row of the compatibility matrix is a unique peak distribution and that most messages form distributions with a unique peak. The product of two unique peak distributions itself has a unique peak, which lies between the peaks of the first two. This fact can be used to eliminate unnecessary multiplications.

Further, Sun et al. [12] use the line and outlier field unification of Black and Rangarajan [4] on their three coupled MRFs (disparity, line, and observation). By using robust statistics to approximate an outlier process for both the observation and line fields, they effectively handled both depth discontinuities and occlusions. They also illustrate the straightforward incorporation of additional cues into BP by adjusting the compatibility matrix ψ_{pq} if pixels p and q are in different segments. By combining robust statistics with image segmentation they developed a stereo algorithm that was demonstrably one of, if not the most accurate at the time.

As noted in this section and in Section 2.3, graphical models incur significant storage and computational complexity. Running time for belief propagation is $O\left(TL^2S\right)$, where T is the number of iterations, L is the number of labels, and S is the number of sites in the MRF lattice. The quadratic relationship of the running time to the number of labels, or states, of each node is particularly prohibitive toward high-dimensional or large state-space fields. Tappen et al. [15] overcome this by using a set of trained linear regressors as their set of labels to model a field with a much larger state space than the number of regressors (labels). They apply this technique to both super-resolution and Bayer tile color demosaicking, where the fields have ranges equal to the number of desired intensity (color channel) levels, typically 256.

Felzenszwalb and Huttenlocher [13] presented three algorithmic techniques to substantially improve the running time of BP for low-level vision problems. These techniques will receive further exposition in Section 3.2 as they are applied to the resampling of input images into a panorama. First, they noted that for early vision problems, such as stereo, the compatibility matrix is often a function only of the difference between the two labels, as opposed to the actual values of the labels. This leads to a message updating scheme, as described in Section 3.2, that is linear in L instead of quadratic in general.

Second, a four-connected image grid graph is a bipartite graph. That is, $\{x_p\}$ can be partitioned into two subsets A and B such that any node $x_a \in A$ has only neighbors $x_b \in B$. Colouring X in a checkerboard pattern and taking A to be one colour and B as the other is such a partition. Given the messages sent from nodes in A at iteration t, we can compute the messages sent from nodes in B at iteration $t + 1$, and in turn the messages sent from nodes in A at iteration $t + 2$ without ever computing the messages sent from nodes in A at iteration $t + 1$. This means only half the messages need to be updated each iteration.

Third, they use a “multiscale” or hierarchical scheme for coarse-to-fine MAP estimation. Messages are typically initialized to the uniform distribution, that is zero in the min-sum version, but if they are initialized closer to their point of convergence, they should take fewer iterations to converge. This is achieved by defining nodes in level $k + 1$ to be the aggregation of four spatial neighbors in level k. The BP algorithm is then iterated at higher levels first, and the resulting messages are used as initial values for messages between the child nodes in next (lower) level.

Finding the globally optimal solution to the Gibbs energy of a 2D pair-wise MRF is generally NP-complete [28]. However, under easily verifiable conditions, which are met for standard stereo benchmark data sets, a variant of the BP algorithm gives the global optima [17]. Tree reweighted belief propagation (TRBP) is a special case of GBP. The global minimum of an energy function of the same form as (2.27) does not improve many of the inaccuracies in the BP or graph cut solutions (to the stereo problem), suggesting that the problem is not the optimization algorithm, but rather the energy function itself [17]. Tappen and Freeman similarly concluded in their comparison of graph cuts and BP for stereo that performance improvements will come from improving the model of the MRF, not by improving the optimization algorithm, noting that both graph cuts and BP found configurations with significantly lower energy than that of the ground truth disparities [16]. For the Middlebury benchmark [51], between approximately 60 and 80 percent of the energy of the ground truth solution was due to high matching costs of occluded pixels [16].

One way to improve the model is to incorporate additional cues, eg. segmentation [12]. The use of robust estimators to incorporate outlier processes is also an improvement of the model [12]. Since appearing, the work of Felzenszwalb and Huttenlocher [13] has been a popular vehicle for extending BP and applying it to improved models, due to their techniques' simplicity and efficiency. At *CVPR* 2005, five contributions [19, 20, 21, 22, 23] related directly to belief propagation, plus the expectation maximization technique for MRF parameter estimation [18]. Williams et al. [20] use a 3D MRF to estimate both stereo and motion in a stereoscopic video sequence using belief propagation, and cite as future work the use of efficient low-level vision techniques for BP [13] to improve the running time of their algorithm. They couple a 2D disparity field with a temporal field for motion to obtain a 3D MRF with compatibility functions between: hidden and observed nodes at time τ, spatially neighboring hidden nodes at time τ, and hidden nodes at the same spatial location at time τ and time $\tau - 1$.

The following year, several more CVPR contributions related directly to belief propagation, including [24, 25, 26, 27]. In [24], a hierarchical BP technique similar to Felzenszwalb and Huttenlocher [13] is used, along with other techniques, to achieve one of the best stereo results to date. Zhang et al. [25] use efficient BP for early vision [13] to solve a disparity or optical flow MRF coupled with an Illumination Ratio Map (IRM) MRF to perform stereo or tracking robustly under variable illumination. They note that if one field is fixed, BP can be used to solve for the other. Hence, they alternately fix the IRM field and solve for the disparity (optical flow) field, and vice versa. This is similar to the technique of Konrad and Dubois [5] in alternately solving for motion and motion discontinuity fields, and the Iterated Conditional Modes (ICM) [59] method used by Fitzgibbon et al. [14] to alternately optimize the photoconsistency likelihood and texture prior for image-based rendering.

A preliminary version of this thesis appeared at the *Third International Symposium on* 3*D Data Processing, Visualization and Transmission (*3*DPVT)* in June 2006 [60].

Chapter 3

Belief Propagation for Panoramic Imaging of Complex Scenes

In this thesis, loopy Bayesian belief propagation is used to estimate the *maximum a posteriori* (MAP) panoramic image given a set of input images taken from various positions and orientations surrounding the desired center of projection for the panorama. To generate each pixel in the panoramic image, the input images are resampled at the estimated MAP locations corresponding to that pixel. Each pixel in the panorama corresponds to a direction from the center of projection of the panorama, and corresponding sampling locations in the input images are computed by back projecting to some depth and reprojecting the resulting 3D point into the input images. (Input images from which the point is not visible are ignored.) Thus, estimating the MAP panoramic image requires estimating the MAP depth at each pixel in the panorama.

The MAP depth will be computed as the MAP configuration $\mathbf{f}^{MAP}$ of the MRF $\mathcal{F}_{\Pi}$ described in Section 2.3, such that it minimizes the energy function (2.27).

When estimating the MAP depth across the pixels in a panoramic image using a Bayesian framework, the first task is computing the observational likelihood that the input images were generated given a particular depth assignment to each pixel. This is also known as the local evidence for each pixel. At each pixel, by measuring photoconsistency at a set of L depths $z(f)$ for $f = 0, 1, \ldots, L-1$ one can observe the likelihood that the images were generated given the scene has depth $z(f)$ at that pixel, or $\phi_p(f)$ as in (2.20). Section 3.1 describes computation of the data cost.

The second task is to impose a Markov smoothing prior on the likelihood to obtain the MAP scene depth and hence the MAP sampling of the input images. This is done using a highly efficient modification of loopy BP for early vision [13], as described in Section 2.4.

Section 3.2 describes the algorithm in greater detail.

The third task is to resample the input images using the MAP sampling, which is described in Section 3.3. Generating a cubic panorama involves generating six planar perspective views, one for each face. MAP resampling can be adjusted to constrain the views to be consistent where they meet, and avoid seams along the edges of the cube.

3.1 Image Sampling and Data Cost Calculation

For each pixel p in the panorama, the data cost for each label $f = 0, 1, \ldots, L-1$ is computed by first back projecting along that pixel's direction to some depth $z(f)$ obtaining the 3D point $\mathbf{P}_f$. The visibility function of $\mathbf{P}_f$ in image j is $V_j(\mathbf{P}_f) = 1$ if it is visible, and 0 otherwise. Dropping the subscript f for clarity, the projection of $\mathbf{P}$ into image j is $\hat{p}_j$ and the colour resulting from sampling image j at $\hat{p}_j$ is $\mathbf{c}_j(\hat{p}_j)$. Then, the colour of pixel p for label f is the weighted mean of the colour samples from the input images from which the point $\mathbf{P}$ is visible

$$\mathbf{c}_p(f) = \frac{1}{W} \sum_j V_j(\mathbf{P})\, w_j(\hat{p}_j)\, \mathbf{c}_j(\hat{p}_j) \tag{3.1}$$

where $w_j(\hat{p}_j)$ is a weight based on the distance of the reprojection from the center of projection of image j and W is the total weight across all visible images. Let $r(\hat{p}_j)$ be the distance of $\hat{p}_j$ from the optical center of image j and $r_{bound}(\hat{p}_j)$ be the distance from the same to the edge of the image passing through $\hat{p}_j$. Then, the weight is $w_j(\hat{p}_j) = r_{bound}(\hat{p}_j) - r(\hat{p}_j)$.

The colour computed in (3.1) is used as the reference colour for computing the photoconsistency of each pixel p. That is, the data cost is the photoconsistency or similarity of the input images with respect to the weighted mean, as opposed to the similarity of each input image with respect to every other input image.

One possible measure for the data cost is the single-pixel variance of luminances

$$D_p(f) = \frac{1}{W} \sum_j V_j(\mathbf{P})\, w_j(\hat{p}_j) \left(\bar{c}_j(\hat{p}_j) - \bar{c}_p(f)\right)^2 \tag{3.2}$$

where $\bar{c}_j(\hat{p}_j)$ and $\bar{c}_p(f)$ are the scalar luminances of the RGB colours $\mathbf{c}_j(\hat{p}_j)$ and $\mathbf{c}_p(f)$, respectively. An extension of this is to sum over a window $\Omega(p)$ centered on p

$$D_p(f) = \sum_j \sum_{q \in \Omega(p)} \frac{1}{W_q} V_j(\mathbf{Q})\, w_j(\hat{q}_j) \left(\bar{c}_j(\hat{q}_j) - \bar{c}_q(f)\right)^2 \tag{3.3}$$

where W_q is the total weight over all input images for pixel q. Experiments for this thesis include data cost computations using (3.2) and (3.3) with a 5×5 window.

In [52], Birchfield and Tomasi introduced a pixel dissimilarity measure that is insensitive to the effects of image sampling inconsistencies that occur between any pair of images. They symmetrically define the dissimilarity of a pixel p_0 in image I_0 and a pixel p_1 in image I_1 as $d(p_0, p_1) = \min\left\{\bar{d}(p_0, p_1, I_0, I_1), \bar{d}(p_1, p_0, I_1, I_0)\right\}$. These symmetric quantities are defined

$$\bar{d}(p_0, p_1, I_0, I_1) = \min_{q \in \frac{1}{2}(p_1)} \left| I_0(p_0) - \hat{I}_1(q) \right|$$

and correspondingly for $\bar{d}(p_1, p_0, I_1, I_0)$, where $\frac{1}{2}(p_1)$ denotes the square window extending one half-pixel from p_1 along both axes, $I_0(p)$ is the result of sampling I_0 at pixel p, and $\hat{I}_1(q)$ is the linear interpolation at sub-pixel location q of I_1. Kolmogorov and Zabih [53] extend this to colour images by averaging the dissimilarity over the colour channels.

In this thesis, the asymmetric dissimilarity of each input image with respect to the weighted mean (3.1)

$$\mathbf{d}_{p,j}(f) = \min_{q \in \frac{1}{2}(\hat{p}_j)} \left| \mathbf{c}_j(\hat{p}_j) - \mathbf{c}_p(f) \right| \tag{3.4}$$

is used to compute a third version of the data cost used in experiments for this thesis. This dissimilarity is averaged over the colour channels [53] and then summed over the visible input images

$$D_p(f) = \sum_j V_j(\mathbf{P}) \left[\frac{1}{3}\,\frac{1}{3}\,\frac{1}{3}\right] \cdot \mathbf{d}_{p,j}. \tag{3.5}$$

Experiments were not conducted using the symmetric dissimilarity [52, 53], however this would be an interesting future experiment.

Equations (3.2), (3.3) and (3.5) give three ways of computing a photoconsistency observation, or data cost, given a specific pixel p, label f and depth associated with that label $z(f)$. However, this begs the question of what depth to associate with a given label. Simple plane-sweep techniques sample at one depth for each label, over all pixels, as shown for a face of a cubic panorama in Figure 3.1. Figure 3.2 shows a simple example of choosing depths $z_p(f)$, for each label f. The data cost, $D_p(f)$ is computed by reprojecting p into the input images (from which it is visible) from depth $z_p(f)$, as shown in Figure 3.3, and then computing the pixel dissimilarity (3.5). The labels' depths might be linearly spaced over a chosen depth interval, eg. Figures 3.2 and 3.3, or might be chosen based on prior knowledge of the scene or imaging process.

A more robust method is to associate each label with a depth that results in an approximate local minimum of the data cost. As described in Section 2.2, Fitzgibbon et al do this in colour space using gradient descent [14]. In this thesis, each label is associated with a sub-interval of a “global” interval in depth space. The sub-intervals are equal in length and contiguous. By sampling at multiple depths within its sub-interval, each label is associated with an approximate local minimum.

This thesis uses a simple hierarchical multi-sampling technique that results in a sampling period (distance between nearest samples) $T_s = 2^{-K_s}\Delta z$, where Δz is the baseline depth increment (distance between centroids in successive intervals) and K_s is the number of levels in the sampling hierarchy. The hierarchical sampling proceeds by sampling at the interval centroid (sampling level 0), splitting the interval into two and sampling at the centroids of those intervals (level 1), splitting each of them into two and sampling, and so on until level $K_s - 1$, always maintaining the sample with the minimum photoconsistency cost $\hat{z}_p\left(f\right)$ for each pixel p, within the label's interval. This technique, illustrated for $K_s = 4$ in Figure 3.4, simply provides a denser exhaustive sampling in depth, without increasing the number of labels needed for the MRF.

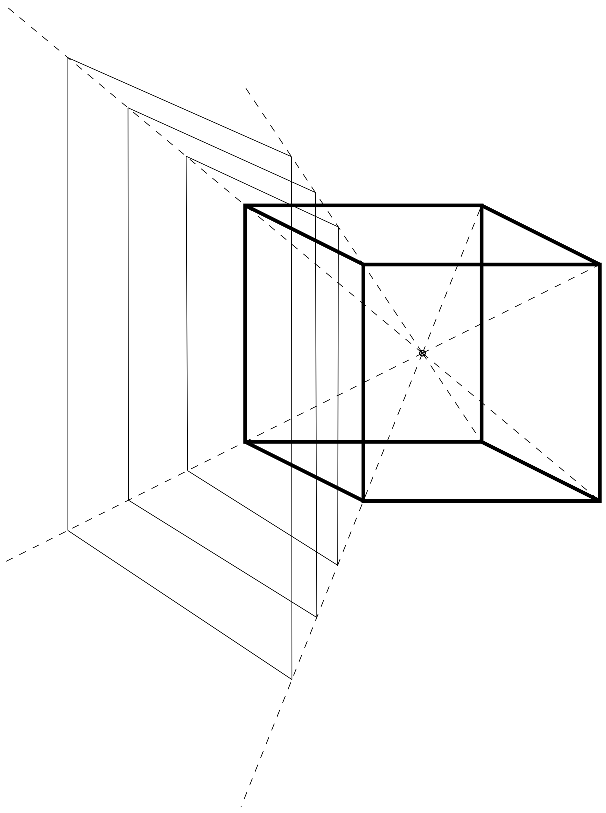


Figure 3.1: Plane-sweeping for a face of a cubic panorama.

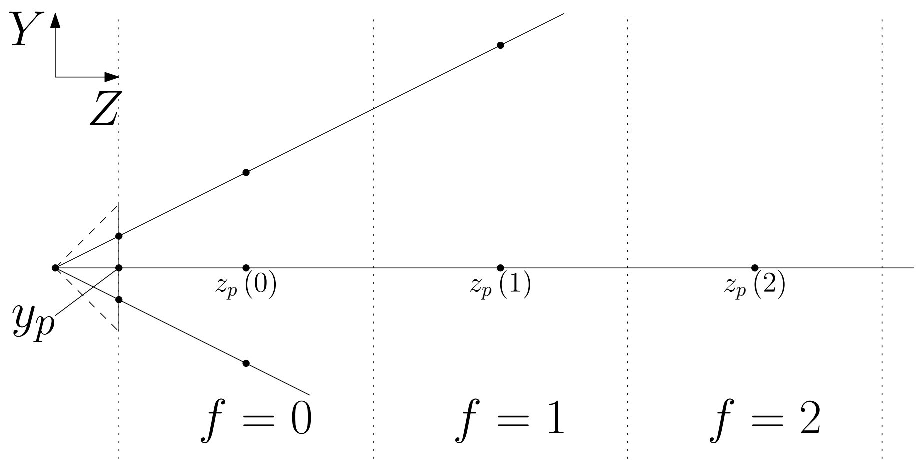


Figure 3.2: Depth sampling associated with the observation node y_p of panoramic pixel p.

3.2 Efficient Belief Propagation for Panoramic Imaging

As noted in Section 2.4, Felzenzswalb and Huttenlocher [13] developed a highly efficient version of the min-sum loopy belief propagation algorithm for low-level vision problems, which we use with minor modifications to minimize the energy function (2.27). Section 3.1 described the different methods used in this thesis for computing the data cost. The other term in (2.27) is the discontinuity cost, for which we use the truncated linear model

$$U_{pq}(f,g) = \min\left(\lambda \left\|f-g\right\|, d_{pq}\right) \tag{3.6}$$

where λ is a scale parameter and d_{pq} is the truncation threshold of the discontinuity cost between pixels p and q. In most experiments $d_{pq} = d$ is constant over the field, however it could be made dependent on edge or similar processes.

It was observed [13] that discontinuity costs like (3.6), that depend only on the difference between the labels f and g and not the specific values of f and g, allow for a two-pass message update algorithm that is linear in the number of labels L. Generally, message updates are $O\left(L^2\right)$ because $U_{pq}(f,g)$ must be evaluated at every label f (of x_p) for every label g (of x_q).

The last two terms of the message update formula (2.51) can be combined into a single term such that (2.51) becomes

$$m^t_{pq}(g) = \min_f \left(U_{pq}(f,g) + h_{pq}(f)\right)$$

where

$$h_{pq}(f) = D_p(f) + \sum_{s\in N(p)\backslash q} m^{t-1}_{sp}(f)\,. \tag{3.7}$$

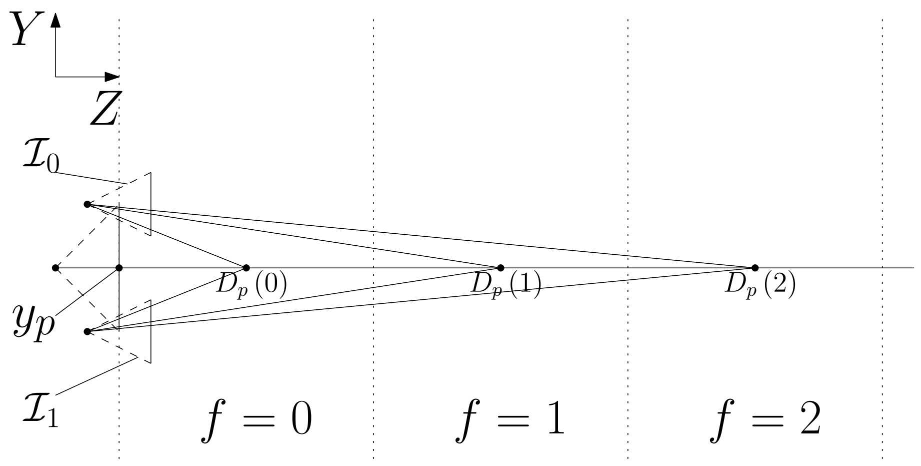


Figure 3.3: The data cost $D_p(f)$ associated with node y_p is made by reprojecting pixel p into the input images ($\mathcal{I}_0$ and $\mathcal{I}_1$) from depth $z_p(f)$.

From (3.6) it is clear that $U_{pq}(f, g+1) = U_{pq}(f, g) + \lambda$ for $g \geq f$ (and $\lambda |g+1-f| \leq d_{pq}$), and hence $U_{pq}(f, g+1) + h_{pq}(f) = U_{pq}(f, g) + h_{pq}(f) + \lambda$. Given this, the observation that $U_{pq}(f, f) = 0$, the fact that $h_{pq}(f)$ is constant over g, and the fact that the message vector components $m^t_{pq}(g)$ are minimized over f, it is clear that

1. $m^t_{pq}(g) \leq h_{pq}(g)$
2. $m^t_{pq}(g) \leq m^t_{pq}(g-1) + \lambda$ (for $g > 0$)
3. $m^t_{pq}(g) \leq m^t_{pq}(g+1) + \lambda$ (for $g < L-1$)
4. $\min_g h_{pq}(g) \leq m^t_{pq}(g) \leq \min_g h_{pq}(g) + d_{pq}$

which leads directly to the following two-pass algorithm. The message vector components are evaluated sequentially over the labels in a forward pass for $f = 0, 1, \ldots, L-1$

$$m^t_{pq}(f) = \{ \begin{array}{ll} h_{pq}(f) & \text{If } f = 0 \\ \min\left(h_{pq}(f), m^t_{pq}(f-1) + \lambda\right) & \text{otherwise} \end{array} \tag{3.8}$$

followed by a backward pass for $f = L-1, L-2, \ldots, 0$

$$m^t_{pq}(f) = \min\left(m^t_{pq}(f), m^t_{pq}(f+1) + \lambda\right) \tag{3.9}$$

$$m^t_{pq}(f) = \min\left(m^t_{pq}(f), \min_g h_{pq}(g) + d_{pq}\right) \tag{3.10}$$

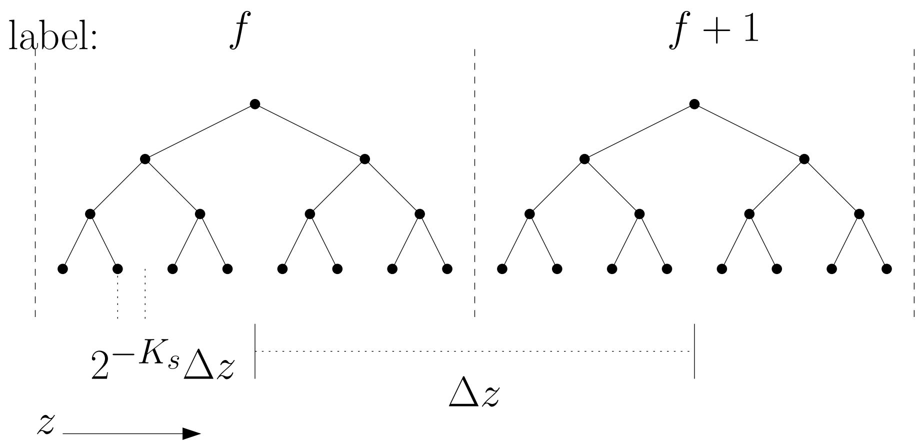


Figure 3.4: Hierarchical multi-sampling technique with $K_s = 4$. The dots indicate depths at which samples are taken from the input images.

where (3.9) is analogous to the forward pass (3.8) and (3.10) applies the truncation threshold for the discontinuity cost (3.6). This algorithm is depicted for a hypothetical message in Figure 3.5. A simple three label example is depicted in Figures 3.8, 3.9 and 3.10. Figure 3.8 shows how the intermediate quantities $h_{pr}(g)$ are computed from the incoming messages $\mathbf{m}_{up}$, $\mathbf{m}_{lp}$, $\mathbf{m}_{dp}$, and the data cost D_p. Figures 3.9 and 3.10 show how the outgoing message vector is modified after the forward and backward passes, respectively, for discontinuity cost scale $\lambda = 1$.

Felzenzswalb and Huttenlocher also observed that a four-connected image grid (one node for each pixel, edges between horizontal and vertical, but not diagonal, neighbors) forms a bipartite graph, where the nodes can be partitioned into two disjoint subsets such that each node only has edges to nodes in the other subset. Thus, the hidden nodes and the edges that connect them of an MRF aligned to an image-grid form a bipartite graph such that the set $\{x\}$ is partitioned into subsets A and B where for every $x_p \in A$ and $(x_p, x_q) \in E$, $x_q \in B$. A bipartite MRF overlayed on an image grid is shown in Figure 3.6.

The consequence of this observation for belief propagation is that only half the messages need to be computed each iteration. In this thesis, for even numbered iterations $t \; mod \; 2 \equiv 0$ messages from nodes in A to nodes in B are updated, and for odd numbered iterations messages from nodes in B to their neighbors in A are updated. Further, in this thesis, message data is stored for only one of the bipartite partitions at any given time, as discussed in Section 4.4.

Consider the case where t is even and messages from A to B are computed. The information needed for this is the data cost and messages from B to A at iteration $t-1$. At iteration

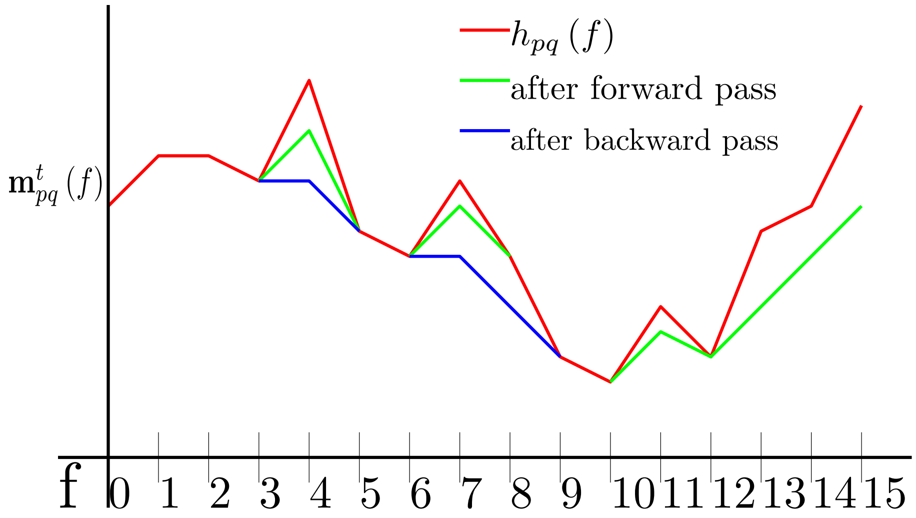


Figure 3.5: Two pass message update algorithm. The horizontal and vertical axes are the label (f) and the message energy, respectively. The red plot shows the initial value of the message vector components computed from the data cost and the incoming messages from previous iterations. The green plot shows the message components after the forward pass (3.8) and the blue plot shows how the message is modified after the backward pass (3.9) and (3.10).

$t+1$, the information needed to compute the messages from B to A is the data cost and messages from A to B at iteration t. Similarly for $t+2$, to compute the messages from A to B only the messages from B to A at iteration $t+1$ are needed. So, at iteration t messages from nodes in A to their neighbors in B were computed without having computed those same messages (from A to B) at iteration $t-1$. At iteration $t+1$, messages from B to A were computed without having been computed at iteration t, and similarly for $t+2$. Hence, only half the messages need to be computed each iteration, and at the end of the T iterations the messages, and hence the beliefs, will be the same as if all messages had been computed every iteration because the messages at iteration t and $t+1$ are the same in the limit (i.e. when the solution converges).

When starting without additional prior knowledge of the scene, it is standard to initialize the messages to a uniform distribution, as discussed in Section 2.4. However, if the messages are initialized closer to their fixed points, the algorithm should require fewer iterations to converge. In [13], multi-resolution MRFs are used to accomplish this by iterating some small number of iterations T at the coarsest resolution and using the resulting messages to initialize the messages at the next, higher resolution MRF, performing T iterations at that resolution, and so on to the highest resolution.

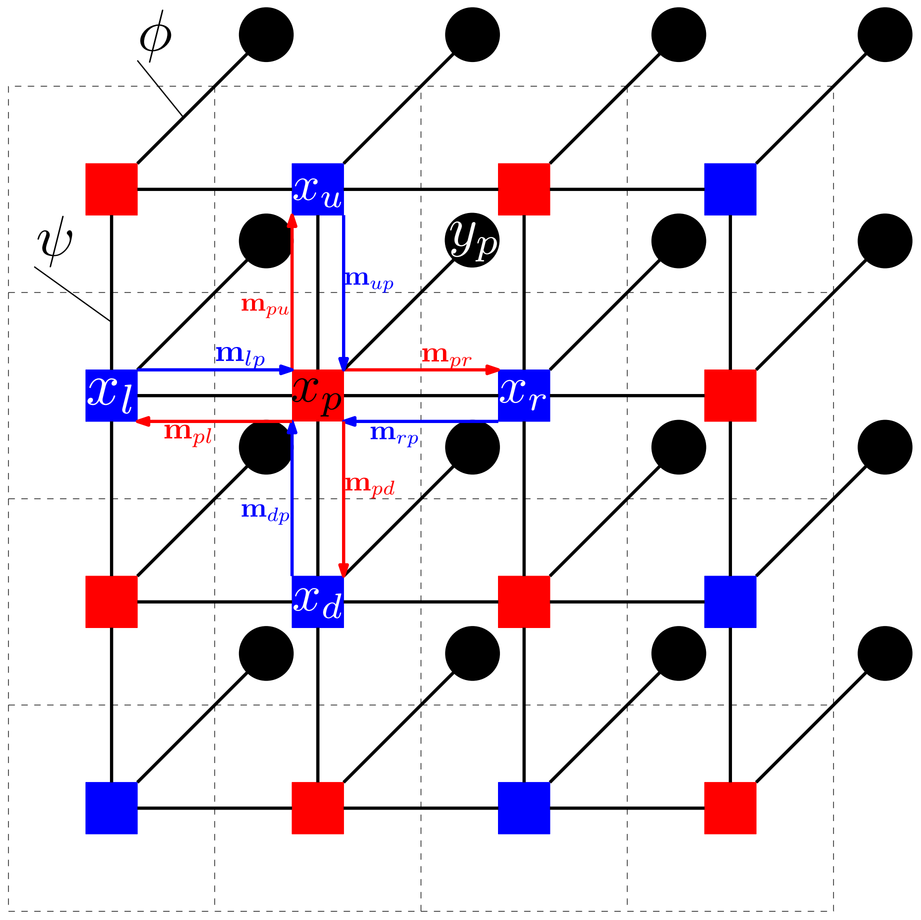


Figure 3.6: MRF with bipartite subsets shown in red and blue.

The MRF at level 0, $\Lambda_0 = \Lambda$ is the original, full-resolution MRF. Starting with $k = 0$, one generates the level $k+1$ MRF by merging spatial neighborhoods of four hidden nodes in Λ_k to obtain a "parent" hidden node in Λ_{k+1} in a quad-tree fashion, as shown in Figure 3.7. The term "parent" is not quite right since the parents are generated from the children, but it will be used here as it does describe the multi-resolution structure once generated, and the coarse-to-fine message initialization can be thought of as a form of "message inheritance".

To make the lower-resolution observations, one can simply sum the data cost for all pixels within the image block corresponding to y_b^k

$$D_b^k(f) = \sum_{p \in b} D_p(f) \tag{3.11}$$

which is equivalent to

$$\phi_b^k(f) = \prod_{p \in b} \phi_p(f) \,. \tag{3.12}$$

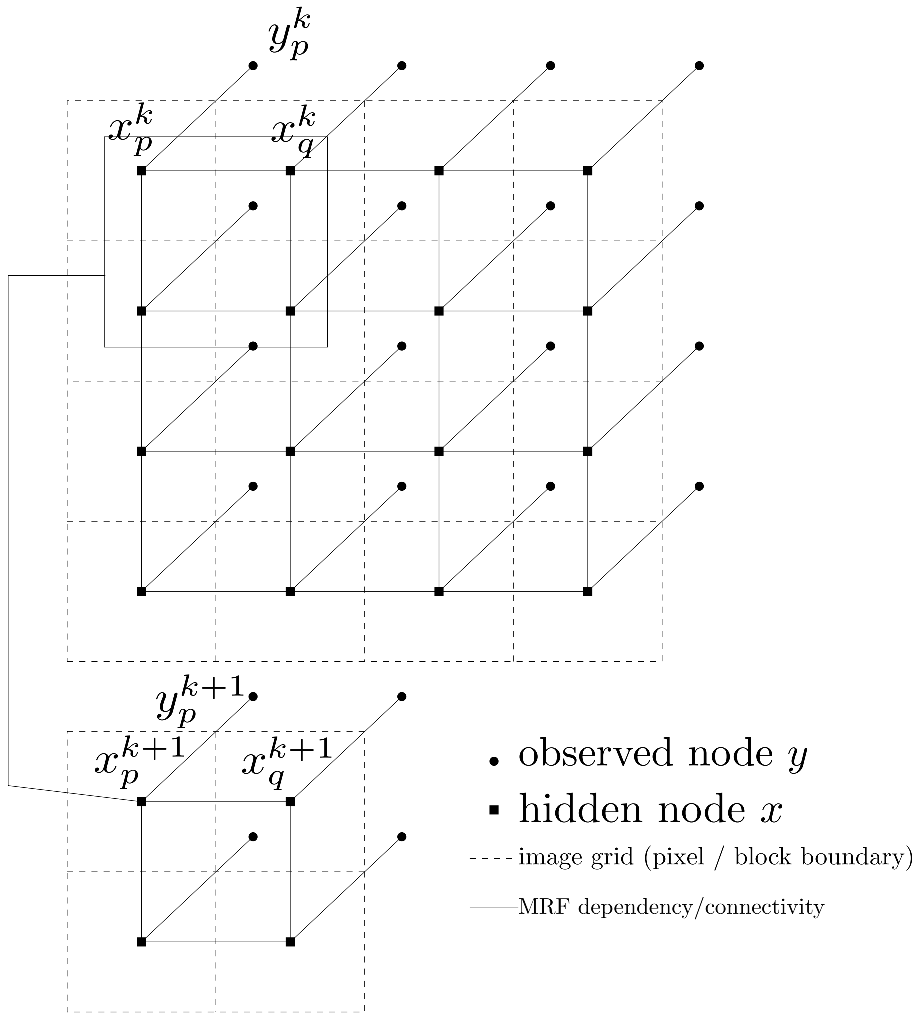


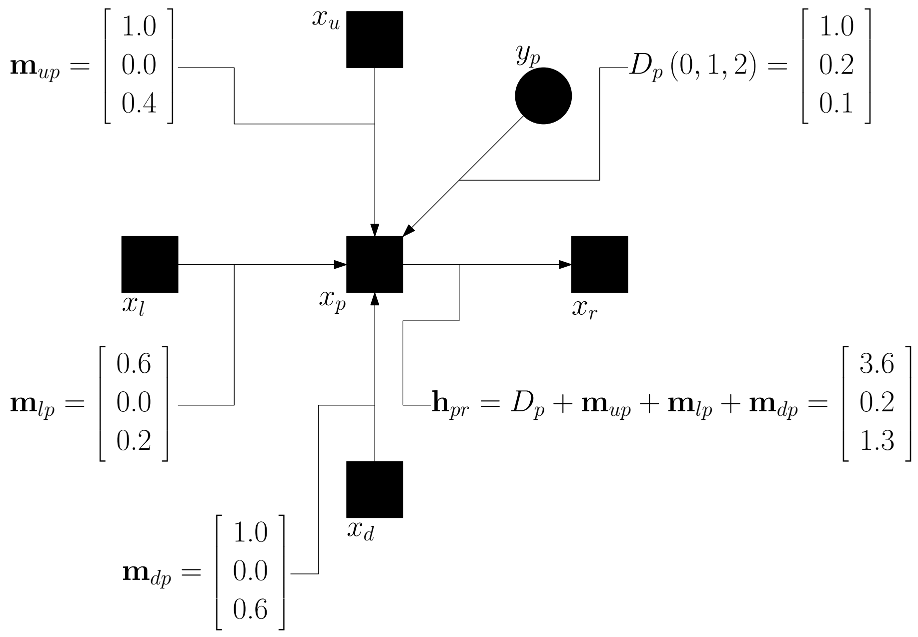
Figure 3.7: Multi-resolution MRFs at adjacent levels.

Message inheritance proceeds down the quad-tree, from level $k+1$ to level k. Let b and c be level $k+1$ blocks and p and q level k blocks such that c and q are the corresponding neighbors (i.e. in the same direction) of b and p, respectively. Then the initial message vector component at level k is

$$m_{pq}^{(k)}(f) = m_{bc}^{T-1}(f) \tag{3.13}$$

for $f = 0, 1, \ldots, L-1$, where $m_{bc}^{T-1}(f)$ is the final message vector component after T iterations at level $k+1$.

The coarse-to-fine message passing has the effect of communicating information across larger distances in panoramic image space. The information communicated at coarser levels is naturally less detailed, so T iterations at level $k+1$ communicates one quarter the information over the same distance as $2T$ iterations at level k, at less than one quarter the computational cost (see Section 4.2).



$\mathbf{m}_{up} = \begin{bmatrix} 1.0 \\ 0.0 \\ 0.4 \end{bmatrix}$
x_u
y_p
$D_p(0,1,2) = \begin{bmatrix} 1.0 \\ 0.2 \\ 0.1 \end{bmatrix}$
x_l
x_p
x_r
$\mathbf{m}_{lp} = \begin{bmatrix} 0.6 \\ 0.0 \\ 0.2 \end{bmatrix}$
$\mathbf{h}_{pr} = D_p + \mathbf{m}_{up} + \mathbf{m}_{lp} + \mathbf{m}_{dp} = \begin{bmatrix} 3.6 \\ 0.2 \\ 1.3 \end{bmatrix}$
x_d
$\mathbf{m}_{dp} = \begin{bmatrix} 1.0 \\ 0.0 \\ 0.6 \end{bmatrix}$

Figure 3.8: Computing intermediate quantity $\mathbf{h}_{pr}$ for a simple example.

3.3 MAP Image Resampling

After T iterations at the full resolution, level 0, MRF, the algorithm computes the belief vectors $\{\mathbf{b}_p\}$ for every pixel p. The *maximum a posteriori* label f_p^{MAP} is computed by (2.53), which in turn gives the MAP depth $z_p^{MAP} = \hat{z}_p\left(f_p^{MAP}\right)$. The back-projection of pixel p to depth z_p^{MAP} gives the 3D point $\mathbf{P}^{MAP}$, which is then reprojected into all sensors from which it is visible. Resampling as in (3.1) produces the MAP colour at pixel p, $\mathbf{c}_p^{MAP}$.

The goal, however, is to compute a seamless resampling of the input images into the panorama, not to recover accurate depth. So motivated, one can apply Gaussian or uniform smoothing to the recovered depth map before resampling the images. The aim of this is to reduce the chance of depth discontinuities, which would cause seams in the panorama, working under the hypothesis that smoothing real depth discontinuities will not noticeably distort the panoramic image. Another option would be to adjust the smoothing kernel based on the confidence of the belief of a given node. Such a confidence measure, or at least its inverse,

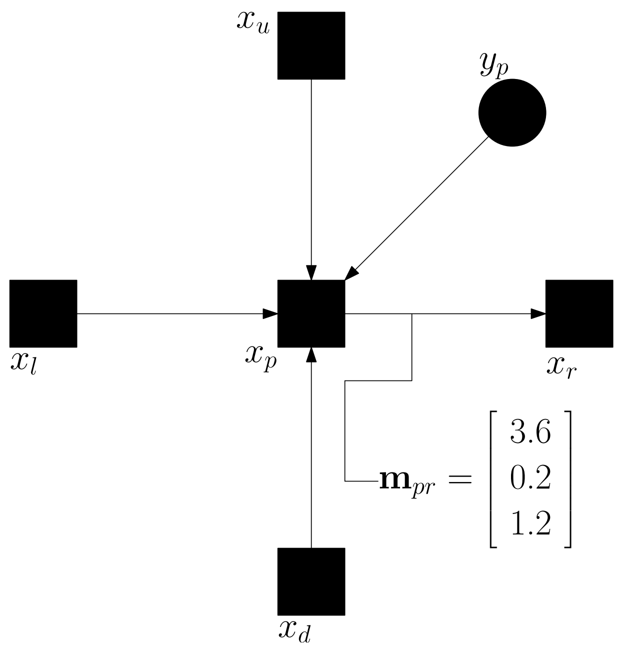
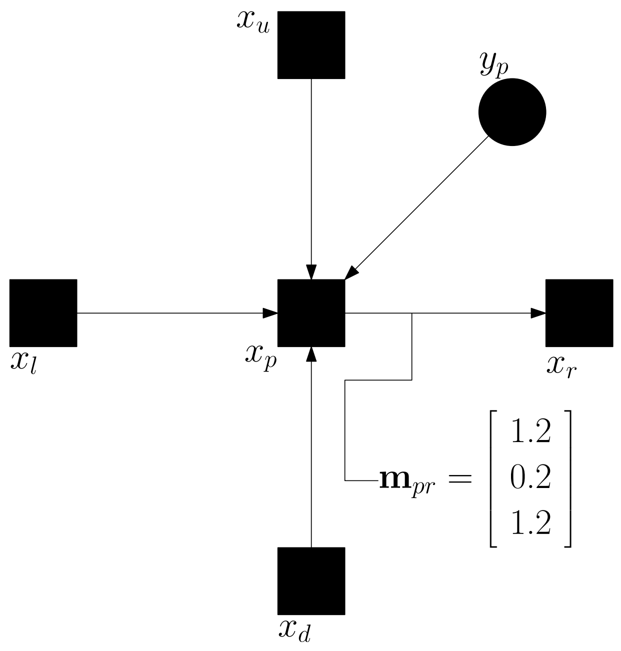

x_u
y_p
x_l
x_p
x_r
$\mathbf{m}_{pr} = \begin{bmatrix} 3.6 \\ 0.2 \\ 1.2 \end{bmatrix}$
x_d

Figure 3.9: After forward pass for message $\mathbf{m}_{pr}$ ($\lambda = 1$).

is readily available in belief propagation: the entropy of the belief vector [12]. Low entropy means high confidence and vice versa.

Further, applying the depth smoothing in cube space, i.e. as a cubic depth map, can reduce the visibility of depth discontinuities at the edges of the cube. This is not a proper solution to the problem of inter-face consistency when generating cubic panoramas as described in this thesis. A proper solution would involve communicating the energy (probability) distributions across the faces of the cube. Another alternative is to use a panoramic image format that can be mapped in a topologically consistent manner into a planar image, such as a spherical panorama, or one that incurs less distortion, such as an octahedral panorama.



x_u
y_p
x_l
x_p
x_r
x_d
$\mathbf{m}_{pr} = \begin{bmatrix} 1.2 \\ 0.2 \\ 1.2 \end{bmatrix}$

Figure 3.10: After backward pass for message $\mathbf{m}_{pr}$ ($\lambda = 1$).

Chapter 4

Hardware Accelerated Belief Propagation for Fast Panoramic Imaging

Using belief propagation to compute a MAP sampling of a set of input images for image-based rendering is computationally very intensive. To improve performance, the parallel nature of the message updates is leveraged by implementing the message passing iterations to run in programmable graphics hardware found in modern desktop and laptop computers, known as the Graphics Processing Unit (GPU).

This chapter is organized as follows. Section 4.1 gives background on general purpose computation with programmable graphics hardware, in particular with respect to imaging and vision problems, and discusses techniques, advantages and limitations of GPU development. Section 4.2 describes the data storage issues in BP on the GPU. Section 4.3 describes the process of sampling the input images and computing the data cost on the GPU. Section 4.4 describes how the message updates are computed on the GPU.

4.1 Graphics Hardware Programming for Vision and Imaging

Driven by the rapid expansion of the computer gaming market, graphics hardware manufacturers have packed increasing computing power onto graphics boards. More significantly, this hardware has become more flexible and programmable, to the point where developers now write short programs that execute on the GPU for every pixel it renders. The result is higher rendering quality and greater realism in games, not simply because more realistic effects can be achieved in real-time, but also because more of the rendering computation is performed on graphics hardware, freeing up the CPU to be used for physics, AI and other tasks that increase the sense of immersion and realism in games. As graphics hardware became more powerful and more flexible, however, developers and researchers began to explore its use for general purpose non-rendering applications, often more efficiently than CPU implementations.

GPUs are single instruction multiple data (SIMD) processors, meaning while they can execute a single instruction at a time, that instruction can be applied to multiple streams of data in parallel. Contemporary GPUs have two programmable stages: vertex processing and fragment processing. Vertex processing involves transforming, displacing and projecting the vertices onto the viewport, a rectangular region of the frame buffer, where the data is eventually rendered. Vertex processing also involves the computation of texture coordinates, which are used in fragment processing to read texture data. This is followed by a process known as rasterization, which maps geometric primitives defined by the vertices onto fragments aligned with the pixels of the viewport. Fragment processing reads texture data from texture coordinates, which are linearly interpolated between those computed at the vertices during vertex processing, and performs additional arithmetic operations to produce the desired output. Fragment colours and depths are output and then undergo other optional, fixed-function processes of blending, stenciling, and so on.

Developers customize vertex processing by writing *vertex shaders*, or *vertex programs*, and fragment processing by writing *fragment shaders* or programs. Vertex shaders output transformed vertices and texture coordinates, which are interpolated and used by fragment shaders to read texture data and write colour data to the frame buffer, and optionally, depth data to the depth buffer. In traditional rendering, the depth buffer is used to ensure that if two fragments project to the same pixel of the frame buffer, only the one nearest the viewpoint will be rendered. In Section 4.3, the depth buffer will be used to ensure a pixel is overwritten only if the photoconsistency cost written as depth is less than the existing cost in the depth buffer.

Vertex and fragment shaders can be written in several different languages. Low-level assembly-style languages have existed for sometime, and high-level C-style languages have been introduced in recent years. OpenGL provides support for both low-level and high-level, as does Microsoft's DirectX. The assembly-style Vertex Shader 2.0 and Pixel Shader 2.0 standards from Microsoft were used in this thesis. The limitations of this standard resulted in many rendering passes, and a more efficient implementation can be achieved using more powerful shading standards, with fewer passes and therefore less overhead [62].

The parallel processing of GPUs has been applied to many non-rendering tasks, including several important tasks in computer vision such as the Canny edge detector and feature point extraction [29]. Yang et al. use graphics hardware for consensus-based scene reconstruction [47], and Gong and Yang use programmable graphics hardware for a real-time stereo technique that include edge-sensitive matching cost aggregation [46]. Many more examples of general-purpose computation on the GPU can be found at the website http://www.gpgpu.org, and the interested reader is thereto referred.

A preliminary report on implementing the Felzenszwalb algorithm [13] on the GPU was presented at the *Third Canadian Conference on Computer and Robot Vision* in Quebec City in June 2006 [61].

4.2 MRF Data Storage

Let us consider the modifications that must be made to store the MRF data needed for BP in graphics memory. Both the data cost and the message data must be re-ordered to be stored effectively in graphics memory and accessed efficiently by the GPU. Conceptually, each message vector should be stored contiguously. That is, the data cost and messages stored sequentially over the entire MRF, as in [13] and shown Figure 4.1. Figure 4.2 shows the re-ordering of the data into texture memory for efficient and parallel access by the GPU. The data cost for each label is stored in a one-channel floating-point texture and the message vector components for each label are stored in a four-channel floating-point texture. Each channel corresponds to a neighbor (right, up, left, down). This allows each iteration of the belief propagation algorithm to run sequentially only in the set of labels $\{0, ..., L-1\}$; at each label, the corresponding message vector components of each pixel are updated in parallel.

Further storage modifications can be made, however. First, as noted in Section 2.4 and 3.2, the MRF corresponding to an image grid is a bipartite MRF. This means not only that half the messages have to be computed each iteration, but also that only half the message data needs to be stored at any given time, as shown in Figure 4.3. Second, at level k the message vector components for 2^{2k} labels can be stored in a single message texture, as shown in Figure 4.4. This can be analogously applied to the data cost. These modifications not only save space, but also provide a disproportionate speed-up due to reduced texture switches, improving the texture cache performance of the GPU.

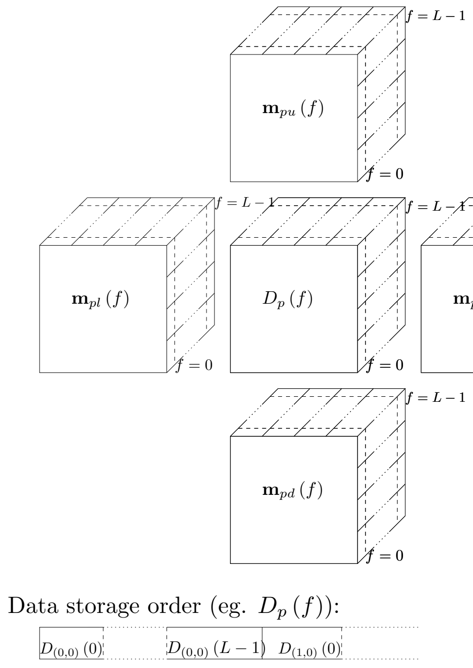


Figure 4.1: Sequential data storage used in [13] for efficient BP on the CPU.

4.3 Image Sampling and Data Cost

The image sampling and data cost techniques described in Section 3.1 were also implemented in graphics hardware, as the geometric and parallel nature of the process make it amenable to GPU implementation. The data cost is computed separately for each face of the cube, since the observation at each pixel is assumed to be independent of the observations at all other pixels. (In the model, $\mathcal{F}_\Pi$, node y_p is only in one clique, $\{x_p, y_p\}$.)

As described in Section 3.1, sampling proceeds by backprojecting each pixel p to a depth $z(f)$ for each label f. The resulting 3D point P is transformed from cube face coordinates to sensor coordinates for each sensor j whose field of view overlaps with that of cube face i. These steps are performed in a vertex shader for each vertex of a quadrilateral aligned with the current face i of the cube. The resulting point P_j is then passed to the fragment shader as a texture coordinate, where it is projected by the pin-hole plus distortion model [56] described in Section 2.1. All steps up to the pin-hole projection are linear transformations, so the results can be linearly interpolated across the surface of the quadrilateral between the vertices.

Once each pixel in the cube face has been reprojected into each input image from which it is visible, a fragment shader samples these images and computes the weighted average of the colour samples (3.1). The weighted average is rendered to a texture, which is read in a

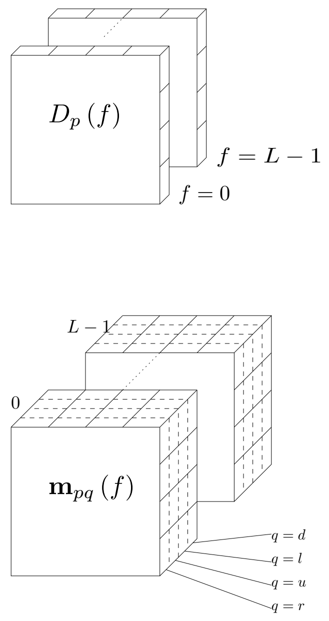


Figure 4.2: Parallel storage of data cost and messages in texture memory for efficient BP on the GPU.

subsequent pass by a fragment shader that computes the photoconsistency based on the dissimilarities of the image samples from their weighted average (3.5).

The depth-wise hierarchical image sampling technique to provide a denser sampling in depth without increasing the number of labels, computes a local minimum of the photoconsistency function within the depth range assigned to each label (see Section 3.1). This is accomplished using the depth buffer in graphics hardware. The depth test comparison is set such that the colour buffers and the depth buffer are only overwritten if the value rendered to the depth buffer is strictly less than the value in the depth buffer. (The default condition is less-than-or-equal-to.) The photoconsistency cost is then written to the depth buffer as well as the data cost texture, which is used as the colour buffer or render target. The depth buffer is cleared for each label. The result is that the data cost for each pixel and each label is the (approximate) local minimum over the depth range assigned to that label.

4.4 Message Propagation

The efficient belief propagation algorithm described in Section 3.2 involves two passes over the labels for each pixel. By rendering a screen-aligned quadrilateral for each label, one can

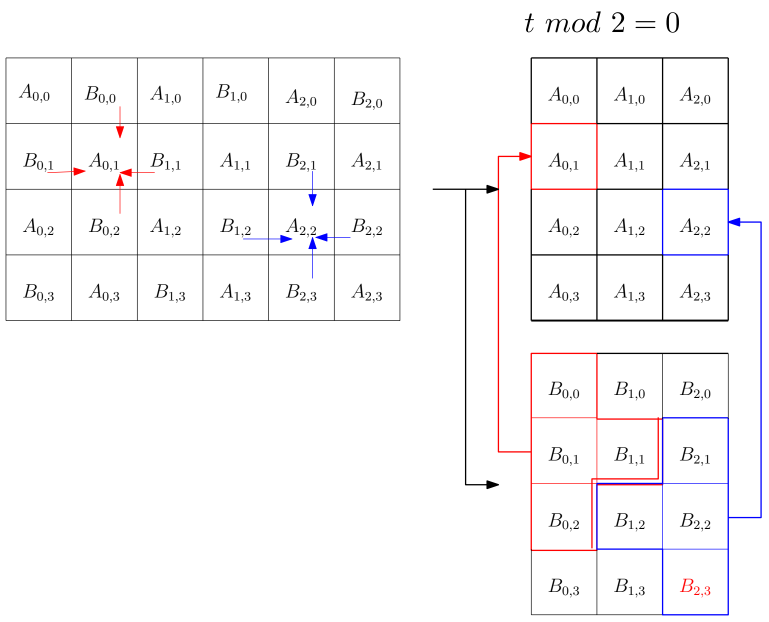
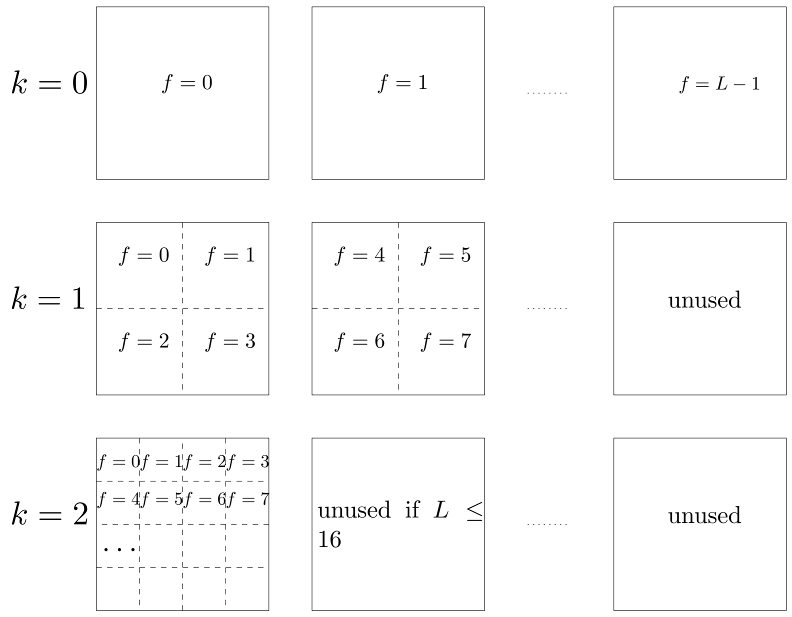

$t \ mod \ 2 = 0$
$A_{0,0}$ $B_{0,0}$ $A_{1,0}$ $B_{1,0}$ $A_{2,0}$ $B_{2,0}$
$B_{0,1}$ $A_{0,1}$ $B_{1,1}$ $A_{1,1}$ $B_{2,1}$ $A_{2,1}$
$A_{0,2}$ $B_{0,2}$ $A_{1,2}$ $B_{1,2}$ $A_{2,2}$ $B_{2,2}$
$B_{0,3}$ $A_{0,3}$ $B_{1,3}$ $A_{1,3}$ $B_{2,3}$ $A_{2,3}$
$A_{0,0}$ $A_{1,0}$ $A_{2,0}$
$A_{0,1}$ $A_{1,1}$ $A_{2,1}$
$A_{0,2}$ $A_{1,2}$ $A_{2,2}$
$A_{0,3}$ $A_{1,3}$ $A_{2,3}$
$B_{0,0}$ $B_{1,0}$ $B_{2,0}$
$B_{0,1}$ $B_{1,1}$ $B_{2,1}$
$B_{0,2}$ $B_{1,2}$ $B_{2,2}$
$B_{0,3}$ $B_{1,3}$ $B_{2,3}$

Figure 4.3: Bipartite packing scheme of message vector components.

use fragment programs to update the messages in the same sequence as the forward and backward passes. However, the fragment programs are executed on the pixels, or fragments, in parallel. One fragment shader computes the forward pass, per (3.8), and is executed for $g = 0, 1, \ldots, L - 1$; another fragment shader computes the backward pass, per (3.9) and (3.10), and is executed for $g = L - 2, L - 3, \ldots, 0$.

Figure 4.4 illustrates the hierarchical data packing described in Section 4.2. For the message data and data cost to be addressed correctly for different levels, the addresses must be offset for the appropriate label. Both the forward and backward passes need the current label (g) and the previous label ($g - 1$ for the forward pass, $g + 1$ for the backward pass). Since these offsets are constant for each label, they can be added to the texture coordinates in a vertex shader. The interpolation between the vertices of the offset texture coordinates is then passed to the fragment shader by the vertex shader, thus saving operations in the fragment shaders. The vertex shader also computes offsets for the previous level ($k + 1$), which is used when inheriting messages from level $k + 1$ to level k.

From (3.8) and (3.7), one can see that the forward-pass fragment shader must read from texture memory the data cost $D_p^k(g)$, the incoming message vector components $m_{qp}^{t-1}(g)$ and the outgoing message vector components computed for the previous label $m_{pq}^t(g - 1)$, for $q \in N(p)$. The operations for computing the intermediate quantity $h_{pq}(g)$ are illustrated in Figure 4.5. From this figure and Figure 4.3, one can see that the locations in texture memory



$k = 0$
$f = 0$
$f = 1$
$f = L - 1$
$k = 1$
$f = 0$ $f = 1$ $f = 2$ $f = 3$
$f = 4$ $f = 5$ $f = 6$ $f = 7$
unused
$k = 2$
$f = 0$ $f = 1$ $f = 2$ $f = 3$
$f = 4$ $f = 5$ $f = 6$ $f = 7$
unused if $L \leq 16$
unused

Figure 4.4: Hierarchical packing scheme used for both the data cost and message vector components.

of the message vector components incoming to node x_p from its neighbors depend on whether the iteration is even or odd, and whether x_p is in an even or odd row of the MRF. The fragment shader constants, or uniform parameters, are used to set neighbor offsets so that the fragment program itself only has to test whether the node is in an even or odd row. The "incoming message channel mask" simply ensures that for each outgoing message $m^t_{pq}(g)$, the incoming messages used are $m^{t-1}_{sp}(g)$ for $s \in N(p) \backslash q$. That is, not $m^{t-1}_{qp}(g)$. The forward-pass fragment shader then adds the discontinuity scale λ to the messages computed for the previous label, and then takes the minimum with $h_{pq}(g)$.

From (3.10), one can see that the backward-pass fragment shader requires $\min_g h_{pq}(g)$ to perform truncation of the discontinuity cost and hence truncation of the message vector components. In fact, this value is required by the backward-pass shader for two reasons. Because messages are additive, data and discontinuity costs are added to incoming messages from the previous iteration, the message vectors must be normalized or they will continue to increase (probabilities will go to zero) and they will not converge. In the min-sum formulation, normalization is accomplished by subtraction, as opposed to division as in the max-product formulation. This thesis normalizes the message vector components by subtracting from them $\min_g h_{pq}(g)$. However, $h_{pq}(g)$ is computed for every g in the forward pass, so making use of multiple render targets, the forward-pass fragment shader also computes $\min_g h_{pq}(g)$, which is then used for normalization and truncation by the backward-pass fragment shader.

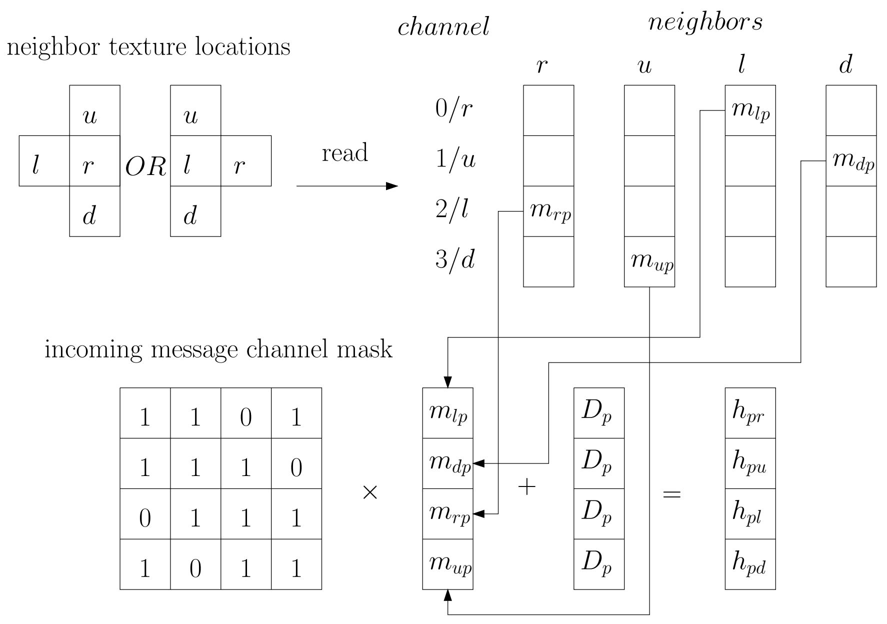


Figure 4.5: Illustration of how the fragment program computes the intermediate quantity $h_{pq}(f)$ used in the message update algorithm described in Section 3.2.

4.5 MAP Image Resampling

As described in Section 3.3, after estimating the MAP label f_p^{MAP} at each pixel $p \in \Pi$ using efficient BP, the images must be resampled at the corresponding depth z_p^{MAP}. Before resampling, these depths are first smoothed to reduce the likelihood of seams appearing in the panorama. The smoothing is performed in cube space to reduce seams along the edges of the cube, where the model is discontinuous.

Specifically, the MAP depths for each face of the cube are loaded into a cubic environment map, which is a special type of texture that is addressed by a 3D direction vector instead of a 2D coordinate vector. Thus, a MAP depth cube $Z^{MAP} = \left\{Z_i^{MAP}\right\}$ for $i = 0, \ldots, 5$, is created. A 5×5 smoothing kernel is applied to this depth cube by computing 2D smoothing kernel sampling coordinates, and then computing the corresponding direction vectors. This last step is essentially a backprojection followed by a rotation according to what face of the cube is being smoothed.

Once the depth cube is smoothed, the input images are resampled and blended for each face of the cubic panorama.

Chapter 5

Experiments

Experiments for this thesis were conducted using the Point Grey Ladybug multi-sensor panoramic camera [63]. While part of the motivation for this approach is the generality and flexibility to be applied under different camera configurations, time did not permit experimentation in this regard. Future work should include experiments with hand-held camera scenarios, and other multi-sensor platforms.

5.1 Multi-sensor Panoramic Camera

This section describes the Point Grey Ladybug camera system. Unless otherwise indicated, information about the Ladybug has been taken from the reference manual [63]. Pictured in Figure 2.3, this is a six-sensor panoramic camera which captures synchronous images such as those in Figure 5.1. The sensors are $\frac{1}{3}$-inch CCD Sony ICX204AK, with a pixel size of $4.65\mu m$ and a resolution 1024×768 pixels. A Bayer colour filter array (CFA) causes the sensor to capture only red, green or blue at any one pixel, and the other colour channels are interpolated from neighboring pixels.

Notice the wide field of view of the images in Figure 5.1, and the high distortion due to the wide-angle lenses used in the system. Calibrating this distortion, and therefore other intrinsic camera parameters, is very difficult. The manufacturer provides a black-box solution that appears to be based on a spline mapping. However, at the time the programming was done for this thesis, it was not clear how to use this black-box calibration in a GPU implementation. To obtain a model and implement it on programmable graphics hardware an open, if imper-

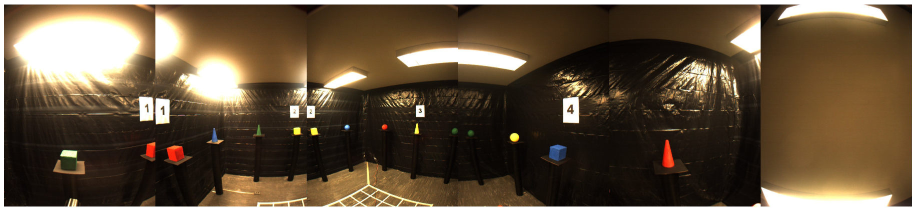


Figure 5.1: Input images captured by the Ladybug of an indoor scene.

fect, calibration was used. The solution used for generating the results in this thesis was to calibrate the intrinsic parameters using a planar-pattern method based on that of Zhang [57], and then compute a look-up table of residual displacements due to unaccounted distortion. This look-up is generated by computing the residual distortion at a set of known scene-image correspondences, and then interpolating the residuals over the densely sampled look-up table. Investigating alternative calibration methods for such lenses is an interesting and challenging topic, but beyond the scope of this thesis.

In fact, it is possible to use the manufacturer's calibration independently of their API. The API provides functionality to obtain a 2D pixel location from a 3D position in space for a given sensor. It also provides a function to return a set of 3D points on the surface of a sphere of radius 5.0 m, and in the solid angle or patch aligned with a given sensor's field of view. This set of points can be projected into the image using the manufacturer's calibration, and the resulting pixel locations can be stored in a look-up table. Points on a cube, or other panoramic surface, can then be projected onto the spherical patch, through the sensor's center of projection, and the corresponding pixel location read from the look-up table. The projection of point on a cube onto the spherical patch through the sensor's center of projection is shown in Figure 5.2. The point O is the origin of the panoramic coordinate system and O_j is the center of projection of sensor j. The unknowns are the angle A and the length s, which gives the corresponding point on the sphere. One can solve for A using the cosine rule: $X^2 = S^2 + t^2 - 2St\cos A$. One can the solve for s using the cosine rule on the smaller triangle, $r^2 = s^2 + t^2 - 2st\cos A$, and then solving a quadratic equation. Implementing this and making use of the manufacturer's calibration is definitely an interesting avenue for future work.

The extrinsic parameters of the sensors, their position and orientation within a common coordinate frame, can be obtained directly from the manufacturers API, in the form of a vector containing three translation components and three Euler angles, which was converted to matrix form for use in this thesis.

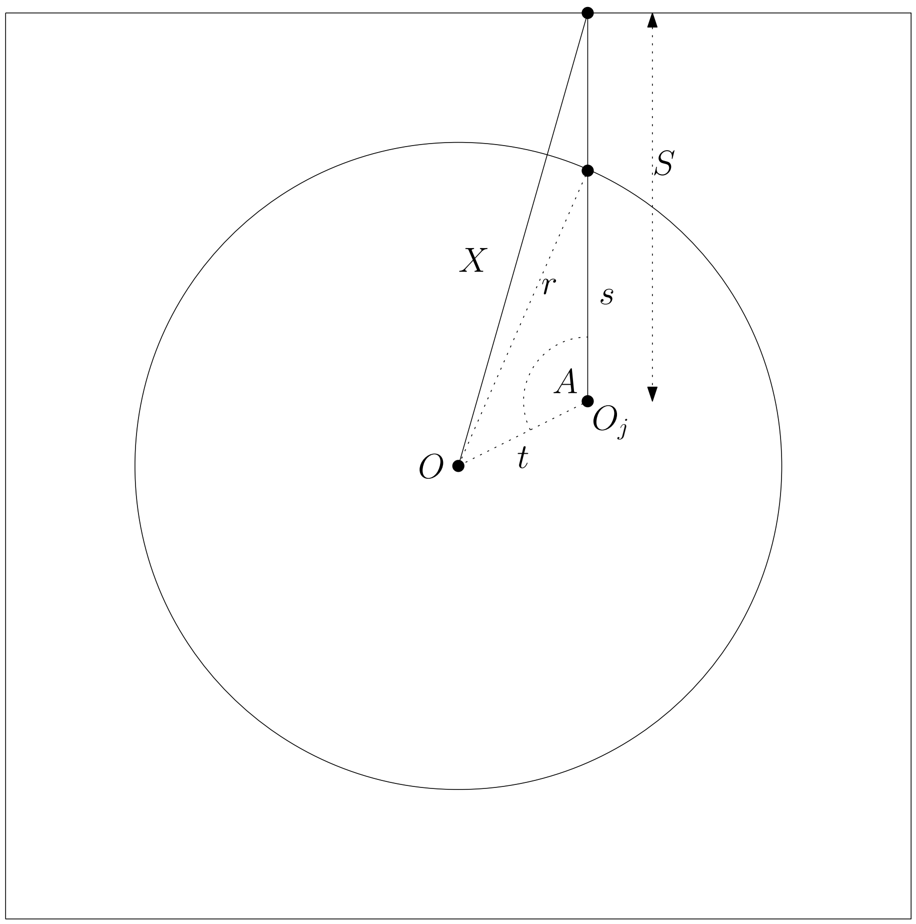


Figure 5.2: A point on a cube is projected onto a sphere through sensor j's center of projection (O_j). The lengths X, S and t are known, and by solving for s one has the corresponding point on the sphere.

5.2 Cubic Panoramas

This section presents cubic panoramas generated using the method described in Chapters 3 and 4, as well as those generated using direct blending and feathering, and direct minimization of photoconsistency (3.5) over a uniform sampling in depth. Figures 5.3, 5.6 and 5.9 show flattened cubic panoramas of an indoor scene generated by blending and feathering, uniform sampling, and belief propagation, respectively. Figures 5.4 and 5.5 are perspective views (1024×768 pixels) of the blended and feathered cubic panorama, Figure 5.3, showing objects near the capture location and in the overlap of two adjacent input images. Notice the ghosting on the "1" and the cube, and the blurriness of the greenball. Figure 5.7 likewise is a perspective view of Figure 5.6, with poor matching on the "1" and the cube, and deformation surrounding the boundary between overlapping and non-overlapping regions. The uniformly sampled depth cube underwent the same smoothing as is applied to those generated by belief propagation, as described in Section 3.3. Figure 5.8 shows direct minimization of a set of depth planes chosen specifically to suit the scene and the camera configuration. The matching in the overlap region is better, but discontinuities along the boundaries of the overlap region are more apparent. Figures 5.10 and 5.11, are similar perspective views of the same scene generated with the method presented in this thesis. Both ghosting and discontinuity artifacts are virtually eliminated. In Figure 5.10, echo artifacts are removed from the “1” and the cube below it. In Figure 5.11, the blurriness of the cube in Figure 5.5 is removed. In Figure 5.9, some ghosting remains on the floor and some discontinuity remains along the edges of the cube where the top face meets one or two side faces. The former is due to calibration error, and the latter is due to a lack of continuity in the MRF model and illumination differences in the input images. A side-by-side comparison of details from Figures 5.4, 5.7 and 5.10 is shown in Figure 5.12.

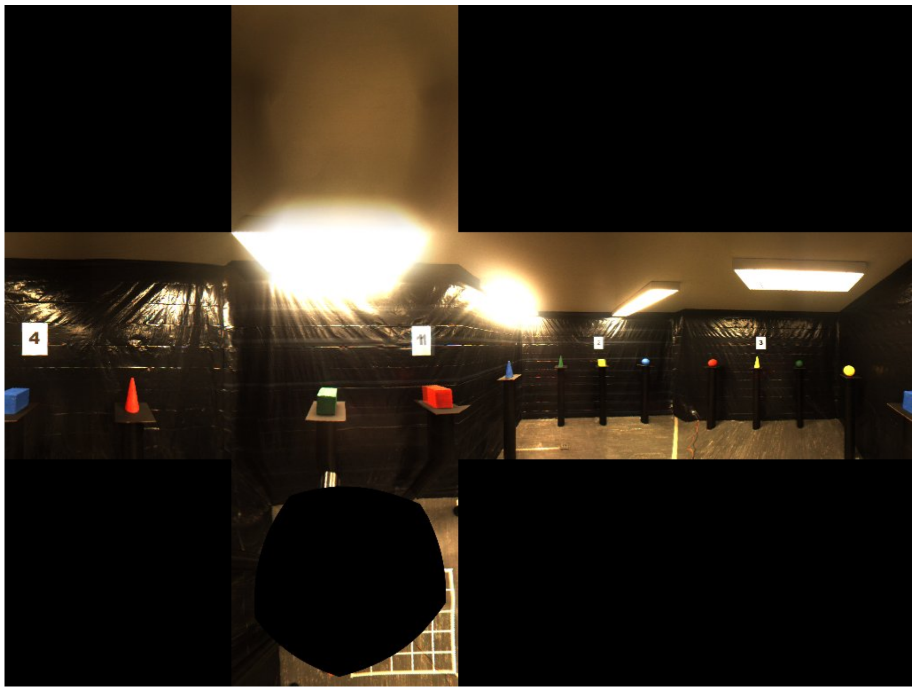


Figure 5.3: Flattened or unfolded cubic panorama of an indoor scene generated by blending and feathering overlap regions.

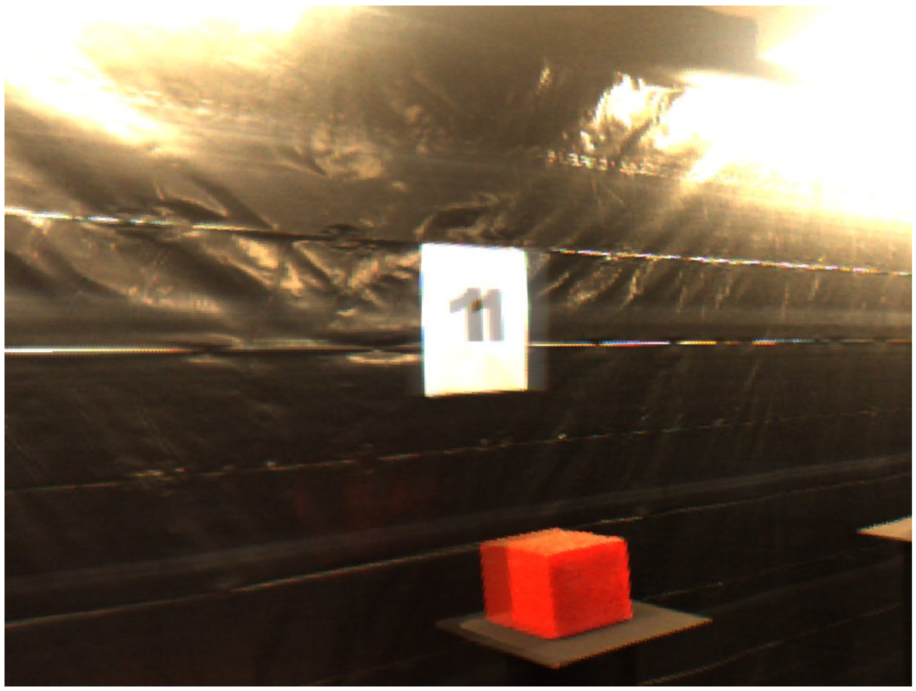


Figure 5.4: Perspective detail of cubic panorama in Figure 5.3.

5.3 Performance

Table 5.1 shows the running times for belief propagation on a CPU and GPU for a 512×512 pairwise MRF corresponding to a face of a cubic panorama. The CPU implementation was that of Felzenszwalb and Huttenlocher [13], and the GPU implementation was that presented in Chapter 4. The same data cost was used for both, hence the same number of labels (16). Also, the same number of hierarchy levels (5) and iterations at each level (12) were used for both implementations. The running times were computed by running each implementation ten times under similar conditions (background processes, etc.) and taking the average. Table 5.2 displays the same information for a 1024×1024 pairwise MRF corresponding to the face of a cubic panorama. Newer processors are used, but in addition to the $(4\times)$ greater number of nodes, more of iterations (20) are used at each MRF level, so the run-times are longer for both implementations. More iterations are required to distribute the information across the larger MRF. Note that the GPU implementation presented in Chapter 4 provides a reduction in processing time by a factor of between 4 and 5.

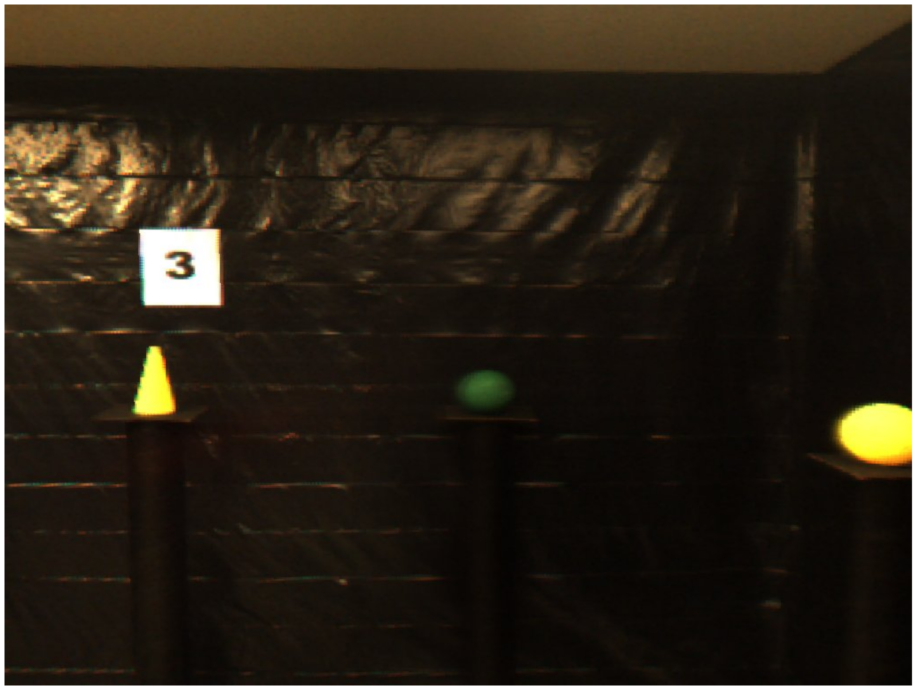


Figure 5.5: Perspective detail of cubic panorama in Figure 5.3.

Processor	AMD Athlon XP 1.6 GHz	ATI Radeon 9500 Pro
Run-time (s)	9.550	1.862

Table 5.1: Run-times for belief propagation on the CPU and GPU for a 512×512 cube side grid.

Processor	Intel P4 3.4 GHz	NVidia GeForce 6800
Run-time (s)	30.073	6.965

Table 5.2: Run-times for belief propagation on the CPU and GPU for a 1024×1024 cube side grid.

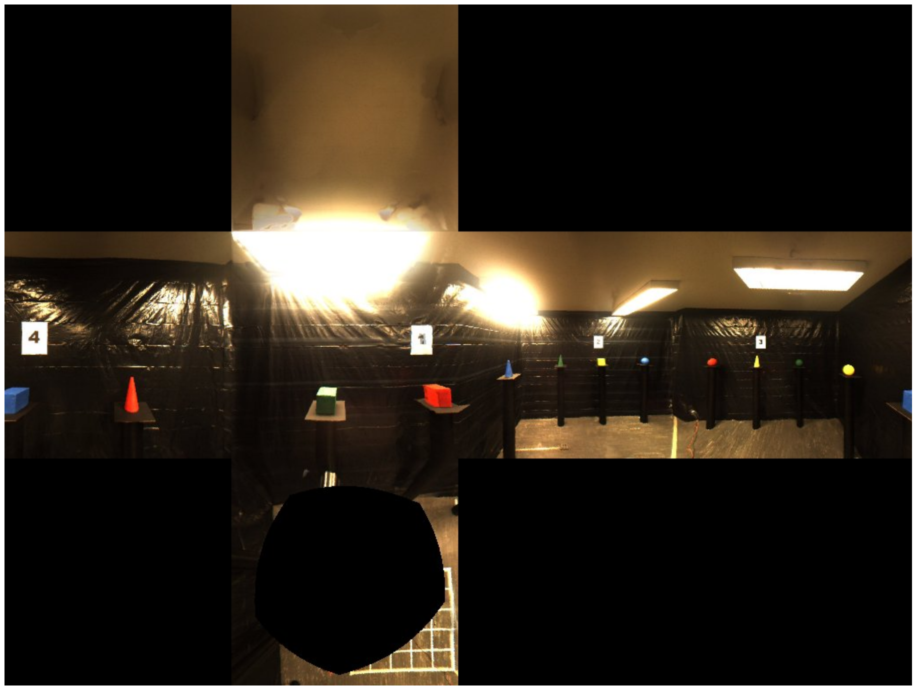


Figure 5.6: Flattened or unfolded cubic panorama of an indoor scene generated by uniform depth stereo matching.

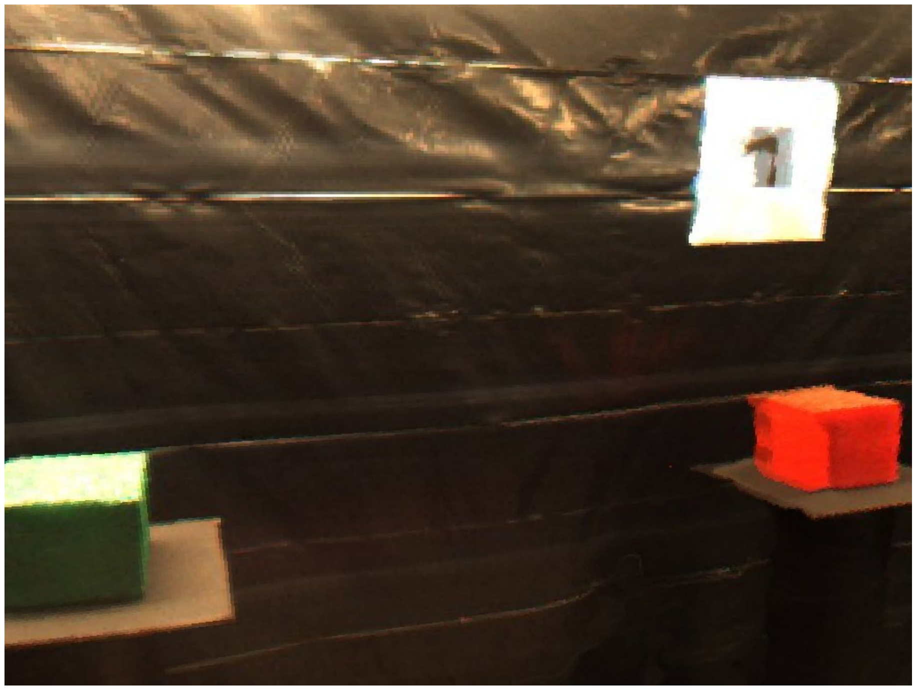


Figure 5.7: Perspective detail of cubic panorama in Figure 5.6.

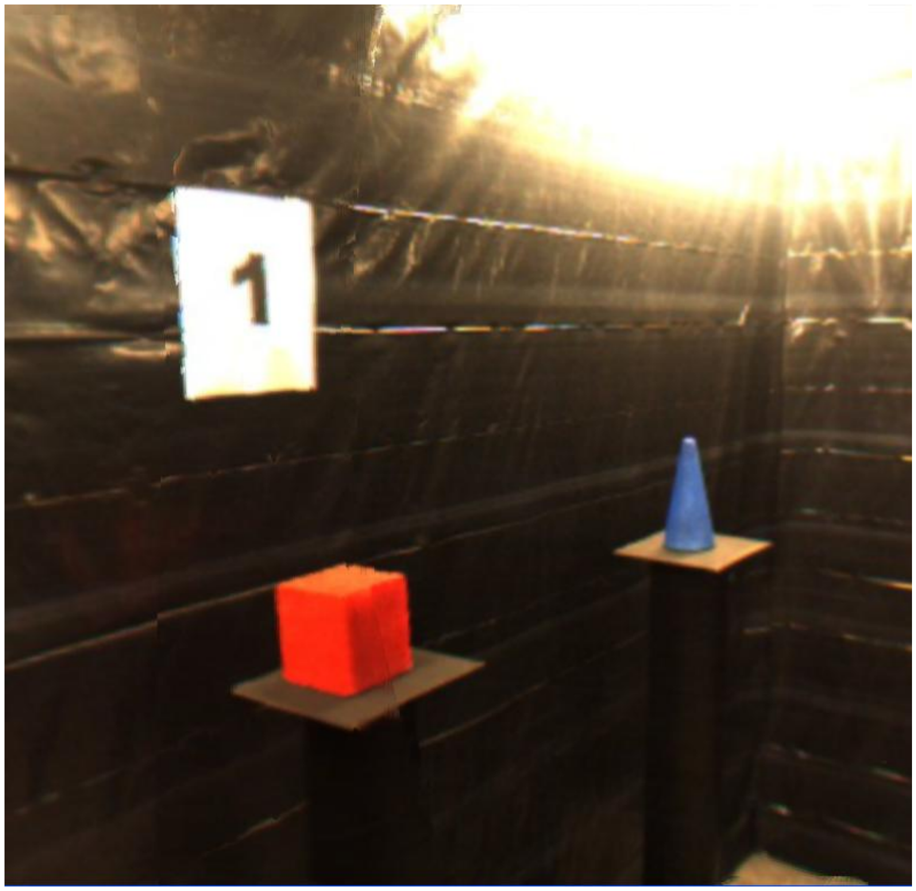


Figure 5.8: Perspective detail of cubic panorama generated by direct minimization of manually selected depths.

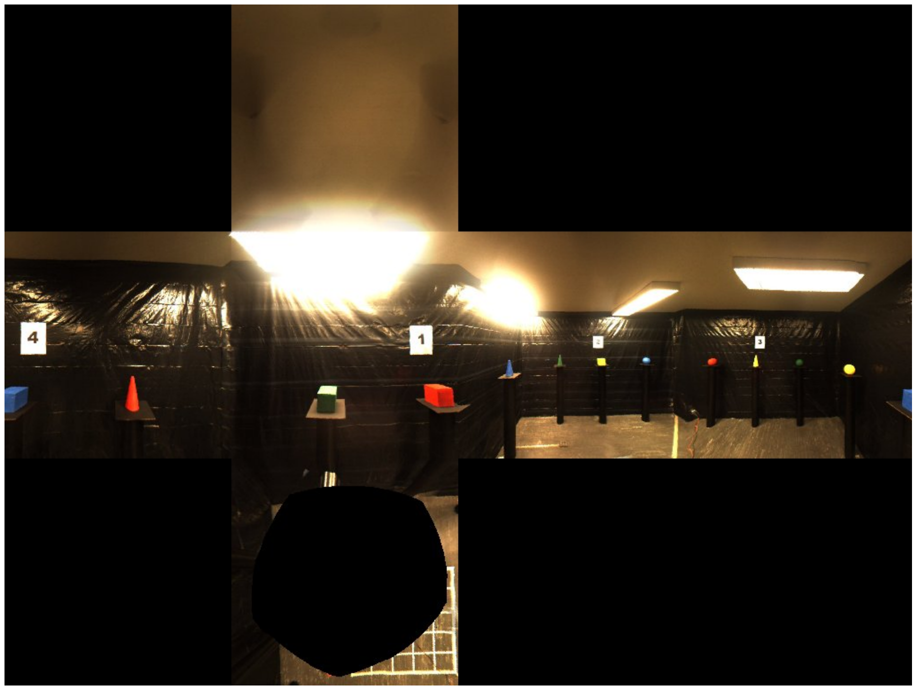


Figure 5.9: Flattened or unfolded cubic panorama generated using this method.

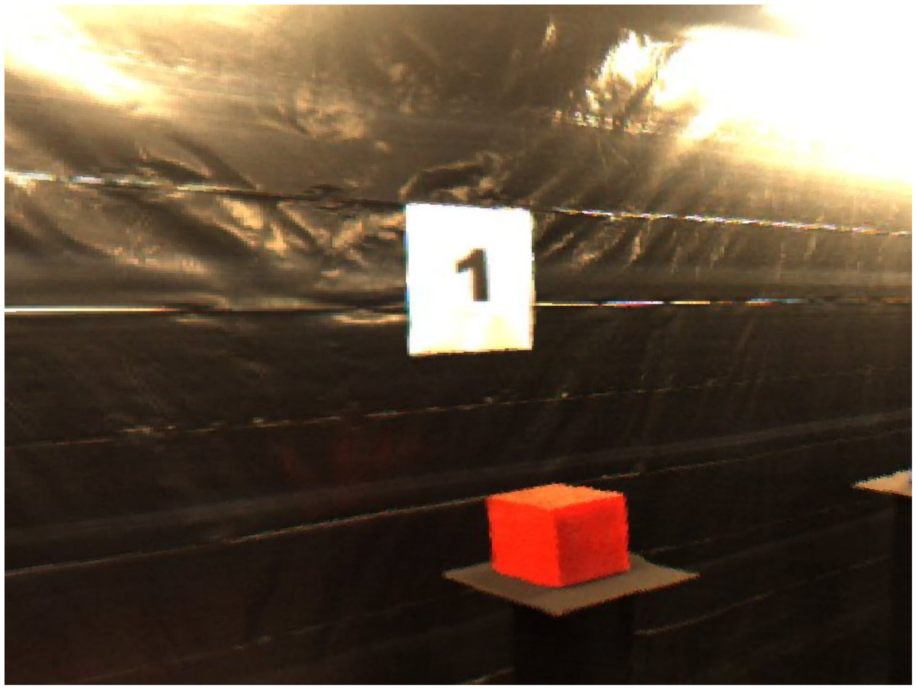


Figure 5.10: Perspective detail of cubic panorama in Figure 5.9.

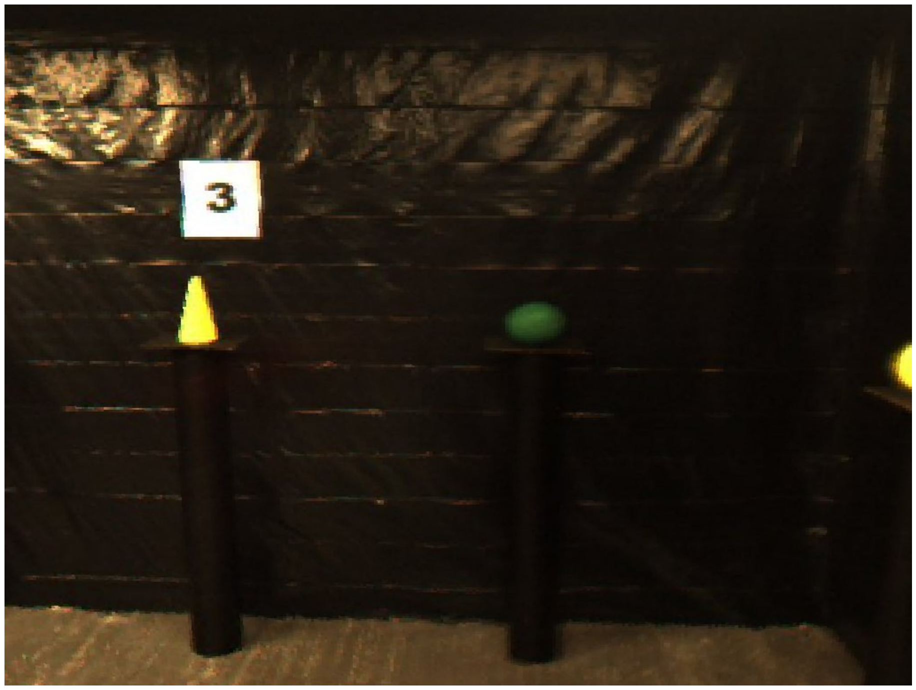


Figure 5.11: Perspective detail of cubic panorama in Figure 5.9.

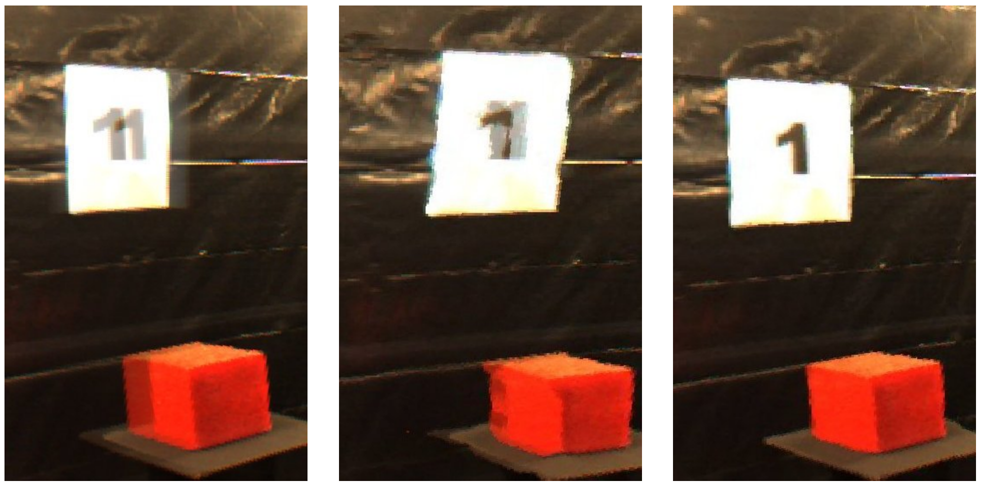


Figure 5.12: Side-by-side comparison of perspective detail from Figures 5.4, 5.7 and 5.10.

Chapter 6

Conclusions

This thesis has presented a Bayesian framework for generating panoramic images of complex, cluttered scenes. The primary contribution is the application of loopy belief propagation for this purpose. Specifically, a very efficient hierarchical belief propagation algorithm [13] is used to compute a dense sampling of the input images, or dense correspondence between the input images, that avoids seams and "ghosting" artifacts in the resulting panorama. As a further contribution, to compensate for the computational expense required for this technique, a programmable graphics hardware implementation is presented that provides significant speed-up of an already efficient belief propagation algorithm.

Experimental results show this technique to be an improvement over simpler techniques for input image configurations that have small amounts of overlap between adjacent images. While the technique of Kang et al. [48] is effective in the same setting, it is my hypothesis that the technique presented here is sufficiently flexible to also be effective in more diverse settings, and would be an effective technique for image-based rendering in general.

6.1 Future Work

There are many potential avenues for future work following on this thesis. Given the stated hypothesis, an obvious one is to conduct experiments on other camera configurations, such as hand-held snapshots in radial configuration or circular video. One area in need of improvement is preventing seams along the edges of the cubic panorama. The ideal solution for this would be to use a full cubic MRF. This, however, has some drawbacks, including computational and storage complexity issues. Also, the hidden nodes of a cubic MRF would not form a bipartite graph, meaning the bipartite optimization could not be used. While a cubic MRF could be implemented in graphics hardware using cubic environment maps, the hierarchical data packing optimization (see Section 4.2) could not be used. More practical would be a hybrid solution, for example using a full cubic MRF at lower resolution and using the results to seed the hierarchical BP algorithm for each face. Another option would to use another type of panoramic image, such as spherical, cylindrical or octahedral, that can be mapped onto a planar MRF in a topologically preserving way (i.e. without cuts).

There is significant future work involved in simply improving the quality of data input to the belief propagation algorithm. This includes using the manufacturer's calibration, which is more accurate than that used for this thesis, as described in Section 5.1. Also, the algorithm should perform better if the labels are spaced linearly with respect to the inverse depth instead of the depth itself. This would result in a more efficient sampling, with more labels closer to the centre of the panorama, because changes in depth make less of difference as depth increases. Further, a more appropriate colour space than RGB should be used. For example, using the perceptually uniform CIELAB colour space should improve matching performance.

Other future work of interest includes incorporating multiple exposure settings for generating high-dynamic range panoramas, and extending the process to a panoramic sequence. It may also profit to perform local minimization in colour space, as in Fitzgibbon et al. [14], and to perform Bayesian estimation of colour rather than geometry, or to represent correspondence differently than depth.

Bibliography

[1] Judea Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, San Mateo, CA, 1988.

[2] Stuart Geman and Donald Geman, *"Stochastic Relaxation, Gibbs Distributions, and the Bayesian restoration of Images", IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 6, No. 6, pp.721–741, November 1984.

[3] Davi Geiger and Federico Girosi, *"Parallel and Deterministic Algorithms from MRF's: Surface Reconstruction", IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 13, No. 5, pp. 401–412, May 1991.

[4] Michael J. Black and Anand Rangarajan, *"On the Unification of Line Processes, Outlier Rejection, and Robust Statistics with Applications in Early Vision", International Journal of Computer Vision*, Vol. 19, No. 1, pp. 57–91, 1996.

[5] Janusz Konrad and Eric Dubois, *"Bayesian Estimation of Motion Vector Fields", IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 14, No. 9, pp. 910–927, September 1992.

[6] Sunil K. Kopparapu and Uday B. Desai, *Bayesian Approach to Image Interpretation*, Kluwer Academic Publishers, Boston, MA, USA, 2001.

[7] Yair Weiss, *"Belief Propagation and Revision in Networks with Loops"* MIT Technical Report, AI Lab Memo 1616, MIT Press, Cambridge, MA, 1997.

[8] Yair Weiss and William T. Freeman, *"On the Optimality of Solutions of the Max-Product Belief-Propagation Algorithm in Arbitrary Graphs", IEEE Transactions on Information Theory*, Vol. 47, No. 2, pp. 736–744, February 2001.

[9] William T. Freeman, Egon C. Pasztor and Owen T. Carmichael, *"Learning Low-Level Vision", International Journal of Computer Vision*, Vol. 40, No. 1, pp. 25–47, 2000.

[10] Jonathan S. Yedidia, William T. Freeman and Yair Weiss, *"Generalized Belief Propagation", Mitsubishi Electric Research Laboratory Technical Report*, TR-2000-26, June 2000.

[11] Jonathan S. Yedidia, William T. Freeman and Yair Weiss, *"Understanding Belief Propagation and its Generalizations", International Joint Conference on Artificial Intelligence IJCAI 2001*, 2001.

[12] Jian Sun, Nan-Ning Zheng and Heung-Yeung Shum, *"Stereo Matching Using Belief Propagation", IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 25, No. 7, pp. 787–800, July 2003.

[13] Pedro F. Felzenszwalb and Daniel P. Huttenlocher, *"Efficient Belief Propagation for Early Vision", IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Washington, DC, pp. 261–268, 2004.

[14] Andrew Fitzgibbon, Yonatan Wexler and Andrew Zisserman, *"Image-based rendering using image-based priors" International Conference on Computer Vision 2004*, ICCV 2004.

[15] M. F. Tappen, B. C. Russell, and W. T. Freeman, *"Efficient graphical models for processing images", IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Washington, DC, pp. 673–680, 2004.

[16] Marshal F. Tappen and William T. Freeman, *"Comparison of Graph Cuts with Belief Propagation for Stereo, using Identical MRF Parameters", International Conference on Computer Vision*, pp. 900–906, 2003.

[17] Talya Meltzer, Chen Yanover and Yair Weiss, *"Globally optimal solutions for energy minimization in stereo vision using reweighted belief propagation", IEEE International Conference on Computer Vision and Pattern Recognition 2005*, CVPR 2005.

[18] Li Zhang and Steven M. Seitz, *"Parameter Estimation for MRF Stereo", IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, San Diego, pp. 288–295, June 2005.

[19] Gang Hua, Ming Hsuan Yang and Ying Wu, *"Learning to Estimate Human Pose with Data Driven Belief Propagation", IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, San Diego, June 2005.

[20] Oliver Williams, Michael Isard and John MacCormick, *"Estimating Disparity and Occlusions in Stereo Video Sequences", IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, San Diego, June 2005.

[21] Yan Li, Yanghai Tsin, Yakup Genc and Takeo Kanade, *"Object Detection Using 2D Spatial Ordering Constraints", IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, San Diego, June 2005.

[22] Mithun Das Gupta, Shyamsundar Rajaram, Nemanja Petrovic and Thomas S. Huang, *"Restoration and Recognition in a Loop", IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, San Diego, June 2005.

[23] Kam-Lun Tang, Chi-Keung Tang and Tien-Tsin Wong, *"Dense Photometric Stereo Using Tensorial Belief Propagation", IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, San Diego, June 2005.

[24] Qingxiong Yang, Liang Wang, Ruigang Yang, Henrik Stewenius and David Nister, *"Stereo Matching with Color-Weighted Correlation, Hierarchical Belief Propagation and Occlusion Handling", IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, New York, June 2006.

[25] Jingdan Zhang, Leonard McMillan and Jingyi Yu, *"Robust Tracking and Stereo Matching under Variable Illumination", IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, New York, June 2006.

[26] Nikos Komodakis and Georgios Tziritas, *"Image Completion Using Global Optimization", IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, New York, June 2006.

[27] Gang Li and Steven W. Zucker, *"Surface Geometric Constraints for Stereo in Belief Propagation", IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, New York, June 2006.

[28] Yuri Boykov, Olga Veksler and Ramin Zabih, *"Fast Approximate Energy Minimization via Graph Cuts", IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 23, No. 11, pp. 1222–1239, November 2001.

[29] James Fung, *"Computer Vision on the GPU", GPU Gems 2*, editted by Matt Phar, NVidia/Addison-Wesley, Toronto, pp. 649–666, 2005.

[30] Edward H. Adelson and James R. Bergen, "The Plenoptic Function and the Elements of Early Vision" in M. Landy and J. A. Movshon (eds), *Computational Models of Visual Processing*, pp. 3–20, MIT Press, Cambridge, MA, 1991.

[31] Shenchang Eric Chen, *"QuickTime VR - An Image-Based Approach to Virtual Environment Navigation" Proceedings of ACM SIGGRAPH 1995*, pp. 29–38, 1995.

[32] Leonard McMillan and Gary Bishop, *"Plenoptic Modeling: An Image-Based Rendering System", Proceedings of ACM SIGGRAPH 1995*, pp. 39-46, 1995.

[33] Richard Szeliski and Heung-Yeung Shum, *"Creating Full View Panoramic Image Mosaics and Environment Maps", Proceedings of ACM SIGGRAPH 1997*, pp. 251–258, 1997.

[34] Heung-Yeung Shum and Richard Szeliski, *"Construction of Panoramic Image Mosaics with Global and Local Alignment", International Journal of Computer Vision (IJCV)*, Vol. 32, No. 2, pp. 101–130, 2000.

[35] M. Brown and D. G. Lowe, *"Recognising Panoramas" International Conference on Computer Vision (ICCV 2003)*, pp. 1218–1225, Nice, France, 2003.

[36] David G. Lowe, *"Distinctive Image Features from Scale-Invariant Keypoints", International Journal of Computer Vision (IJCV)*, Vol. 60, No. 2, pp. 91–110, 2004.

[37] Shmuel Peleg and Joshua Herman, *"Panoramic Mosaics by Manifold Projection", IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 338–343, 1997.

[38] Paul Rademacher and Gary Bishop, *"Multiple-center-of-projection images", In Proceedings of SIGGRAPH*, pages 199206, July 1998.

[39] R. T. Collins, *"A space-sweeping approach to true multi-image matching", IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 358–363, San Francisco, CA, June 1996.

[40] S. B. Kang, R. Szeliski and J. Chai, *"Handling occlusions in dense multi-view stereo", IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 103–110, Kauai, HI, December 2001.

[41] Z. Zhu, E.M. Riseman and A.R. Hanson, *"Parallel-perspective stereo mosaics", International Conference on Computer Vision (ICCV)*, pp. 345–352, Vancouver, BC, 2001.

[42] Heung-Yeung Shum, Sing Bing Kang and Shing-Chow Chan, *"Survey of Image-Based Representations and Compression Techniques", IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 13, No. 11, pp. 1020–1037, November 2003.

[43] S. Chen and L. WIlliams, *"View interpolation for image synthesis", ACM Annual Computer Graphics Conference*, pp. 279–288, August 1993.

[44] Steven M. Seitz and Charles M. Dyer, *"View morphing", ACM Annual Computer Graphics Conference*, pp. 21–30, New Orleans, LA, August 1996.

[45] H.-Y. Shum and L.-W. He, *"Rendering with concentric mosaics", ACM SIGGRAPH '99*, pp. 299–306, Los Angeles, CA, August 1999.

[46] Minglun Gong and Ruigang Yang, *"Image-gradient-guided Real-time Stereo on Graphics Hardware", Proceedings of 3DIM 05*, pp. 548–555, 2005.

[47] Ruigang Yang, Greg Welch and Gary Bishop, *"Real-Time Consensus-Based Scene Reconstruction using Commodity Graphics Hardware", Pacific Graphics 2002*, pp. 255–234, 2002.

[48] Sing Bing Kang, Richard Szeliski, Matthew Uyttendaele, *"Seamless Stitching using Multi-Perspective Plane Sweep", Microsoft Research Technical Report*, MSR-TR-2004-48.

[49] Matthew Uyttendaele, Antonio Criminisi, Sing Bing Kang, Simon Winder, Richard Szeliski and Richard Hartley, *"Image-Based Interactive Exploration of Real-World Environments", IEEE Computer Graphics and Applications*, Vol. 24, No. 3, May/June 2004.

[50] D. Scharstein and R. Szeliski, *"A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms", IJCV 47(1/2/3):7-42*, April-June 2002.

[51] D. Scharstein and R. Szeliski, *"High-accuracy stereo depth maps using structured light", In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2003)*, volume 1, pages 195-202, Madison, WI, June 2003.

[52] Stan Birchfield and Carlo Tomasi, *"A Pixel Dissimilarity Measure That Is Insensitive to Image Sampling", IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 4, April 1998.

[53] Vladimir Kolmogorov and Ramin Zabih, *"Multi-camera Scene Reconstruction via Graph Cuts", Proceedings of European Conference on Computer Vision, 2002* (ECCV 2002).

[54] Marc-Antoine Drouin, Martin Trudeau and Sebastien Roy, *"Geo-consistency for Wide Multi-Camera Stereo", IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, San Diego, pp. 351–358, June 2005.

[55] Roger Y. Tsai, *"A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses", IEEE Journal of Robotics and Automation*, Vol. 3, No. 4, pp. 323–344, August 1987.

[56] Zhengyou Zhang, *"A Flexible New Technique for Camera Calibration", IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 11, pp. 1330–1334, November 2000.

[57] Zhengyou Zhang, *"A Flexible New Technique for Camera Calibration"* Microsoft Technical Report, MSR-TR-98-71, Microsoft Research, Redmond, WA, 1998.

[58] Janne Heikkila and Olli Silven, *"A Four-step Camera Calibration Procedure with Implicit Image Correction", IEEE Conference on Computer Vision and Pattern Recognition*, CVPR 1997.

[59] J. Besag, *"On the Statistical Analysis of Dirty Pictures", Journal of the Royal Statistical Society, Series B*, Vol. 48, No. 3, pp. 259–302, 1986.

[60] Alan Brunton and Chang Shu, *"Belief Propagation for Panorama Generation", Third International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)*, Chapel-Hill, NC, June 2006.

[61] Alan Brunton, Chang Shu and Gerhard Roth, *"Belief Propagation on the GPU for Stereo Vision", Third Canadian Conference on Computer and Robot Vision (CRV06)*, Quebec City, QC, June 2006.

[62] Qingxiong Yang, Liang Wang, Ruigang Yang, Shengnan Wang, Miao Liao and David Nister, *"Real-time Global Stereo Matching Using Hierarchical Belief Propagation", British Machine Vision Conference (BMVC06)*, pp. 989–998, 2006.

[63] *Ladybug Spherical Digital Video Camera System User Manual and API Reference*, Version 1.1, 2002-2003, Point Grey Research, Inc.