

Belief Propagation for Panorama Generation

Alan Brunton
University of Ottawa
School of Information Technology and Engineering
Ottawa, Ontario K1N 6N5
abrunton@site.uottawa.ca

Chang Shu
National Research Council
Institute for Information Technology
Ottawa, Ontario
chang.shu@nrc-cnrc.gc.ca

Abstract

We present an algorithm for generating panoramic images of complex scenes from a multi-sensor camera. We further present a programmable graphics hardware implementation to process the large data sets more quickly. Because the sensors do not share the same center of projection, nearby objects may not be properly aligned, creating a ghosting or echoing effect in the generated panorama, unless correct depth information is taken into account. Taking a cue from the similar problem of dense stereo, we approximate our scene with a Markov random field and use belief propagation to estimate the maximum a posteriori panoramic image for that scene.

1. Introduction

This paper presents a Bayesian belief propagation approach to generating panoramic images of complex (indoor/outdoor) scenes by fusing the overlapping images from a multi-sensor camera system. Because there is parallax between adjacent sensors in this system, objects near the camera will introduce “ghosting” or “echo” effects if a simple blending and feathering approach is used to combine the images into a panorama. Panoramas can also be captured using mirror/lens combinations and a single CCD sensor, but we will concern ourselves only with multi-sensor or multi-image systems for this paper.

Panorama mosaicking has been well studied. Chen presented QTVR in 1995 [11], in which the input images are assumed to relate to one another by pure rotation, hence there is no parallax between them. In such a scenario feature-based stitching gives excellent results. Brown and Lowe used more advanced features to automatically stitch panoramic mosaics from a set of input images [12]. In their work, the input images were not assumed to be related only by rotation, but the scenes were distant from the camera

and therefore the images could be related by planar homographies.

These methods are effective because most panoramas are of outdoor scenes, or of a large, open indoor environment, where the geometry can be taken as planar. Our work, however, is motivated by the use of panoramic images for interactive walk-throughs of complex environments¹, where the geometry is likely to be near enough to the camera system that the parallax between sensors becomes a problem. We generate cubic panoramas, which are comprised of six square, 90 degree field-of-view, axis-aligned perspective images with a common center of projection. By reprojecting each pixel in the cubic panorama into the sensor images we obtain a color for that pixel as well as a photoconsistency, if it reprojects into more than one sensor image. For complex scenes, we must find the correct reprojection (i.e. depth) to avoid artifacts and sample the sensor images more effectively.

This makes our task one of image-based rendering (IBR) by dense correspondence. Obtaining dense correspondence is essentially the stereo problem, and we look to stereo methods which effectively combine data over large distances in the image to avoid local minima, which would cause artifacts. As discussed further in Section 3, we are only able to compute dense correspondence in the fractions of the input images that overlap, hence an additional need for a long-ranging method. Bayesian belief propagation (BP) is such a method, and has been applied to the stereo problem with success. Section 2 gives a brief overview of belief propagation as applied to low-level vision problems. Section 3 describes the multi-sensor camera system used in this research. Section 4 discusses the application of belief propagation to cubic panorama generation. Section 5 describes the implementation of this algorithm on programmable graphics hardware to speed the processing of large data sets. Section 6 presents results and Section 7 draws conclusions from them and considers areas for future work.

¹Learn more about the NAVIRE project at <http://www.site.uottawa.ca/school/research/viva/projects/ibr/>

2. Belief Propagation for Low-level Vision

We represent a Markov random field (MRF) Λ by a set Φ of potential functions over cliques in an undirected graph $G = (V, E)$ where each node in the set V represents a random variable and the edges E represent the dependencies between pairs of these variables.

Typically for low-level vision problems a pairwise MRF is used. That is, the potential functions are defined over pairs of nodes connected by a single edge. The set V is comprised of two subsets: $Y = \{y_p\}$ represent observed random variables Y_p , for every pixel p in the output image, and $X = \{x_p\}$ represent hidden quantities X_p about the scene, the values of which we wish to infer. Each random variable can take one of L possible values or labels. For every pixel p , $(x_p, y_p) \in E$. The hidden nodes X are connected in a rectangular grid lattice such that $(x_p, x_q) \in E$ iff pixels p and q are non-diagonal neighbors in the image. We now define the *local evidence*

$$\phi(x_p, y_p) = P(Y_p | X_p) \quad (1)$$

as the likelihood of observing Y_p (i.e. that the input images could have been observed) given that a particular labelling of X_p is the correct one. We define the *compatibility matrix*

$$\psi(x_p, x_q) = P(X_q | X_p) \quad (2)$$

as the probability of X_q given X_p , which is the *Markov blanket* property of Λ . The corresponding density functions are denoted as

$$\phi_p(f) = p(Y_p | X_p = f) \quad (3)$$

and

$$\psi_{pq}(f, g) = p(X_q = g | X_p = f) \quad (4)$$

for labels $f, g \in \{0, 1, \dots, L-1\}$. Abusing the notation slightly to allow X and Y to refer to both the sets of nodes and the corresponding sets of random variables, we can write the overall joint probability of all nodes in V as [2]

$$P(X, Y) = c_{X,Y} \prod_{(p,q) \in E} \psi(x_p, x_q) \prod_{p \in V} \phi(x_p, y_p) \quad (5)$$

where $c_{X,Y}$ is a normalization constant and (p, q) is shorthand for the edge (x_p, x_q) . We can also write the posterior as

$$P(X|Y) = c_{X|Y} \prod_{(p,q) \in E} \psi(x_p, x_q) \prod_{p \in V} \phi(x_p, y_p) \quad (6)$$

where $c_{X|Y} = \frac{c_{X,Y}}{P(Y)}$ is another normalization constant.

Maximizing either (5) or (6) is computationally intractable for reasonable sized graphs, and certainly for

panoramic images. Hence the need for approximation algorithms. We use the max-product version of belief propagation to estimate the maximum a posteriori (MAP) probability of X given Y .

Belief propagation is most commonly compared to graph cuts (eg. [7]). Tappen and Freeman performed a comparison of both algorithms as applied to the stereo problem [5] 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 occluded pixels that did not match any in the other image.

In our case, labels denote discrete depth levels. Belief propagation iteratively sends messages \mathbf{m}_{pq}^t from every hidden node x_p to each of its (hidden) neighbors x_q at each iteration t . Each message is a vector of length L , with each component being proportional to how likely node x_p “believes” it is that node x_q will have the corresponding label. In the max-product algorithm messages are updated in the following way [3, 5]

$$\mathbf{m}_{pq}^t(g) = \kappa \max_f \left(\psi_{pq}(f, g) \phi_p(f) \prod_{s \in N(p) \setminus q} m_{sp}^{t-1}(f) \right) \quad (7)$$

where κ is a normalization constant. After T iterations, the beliefs are computed [3, 2]

$$\mathbf{b}_p(f) = \kappa \phi_p(f) \prod_{q \in N(p)} m_{qp}^T(f) \quad (8)$$

and the MAP labelling for node x_p is

$$f_p^{MAP} = \arg \max_f \mathbf{b}_p(f). \quad (9)$$

Because BP is an iterative algorithm, finding optimizations and improving the time to convergence by either decreasing the number of iterations needed, or by decreasing the amount of time required for each iteration, can be very important. Sun et al. [3] 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 for 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.

Felzenszwalb and Huttenlocher [4] presented three algorithmic techniques to substantially improve the running time of BP for early vision problems. These optimizations were implemented in programmable graphics hardware for

this work, and their algorithm forms the basis of ours. The algorithm is further in Section 4.

First, they noted that for early vision problems, such as stereo, the compatibility matrix is 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 4, that is linear in L instead of quadratic, as is generally the case.

Second, a four-connected image grid graph is a bipartite graph. That is, X can be partitioned into two subsets A and B such that any node $x_p \in A$ has only neighbors $x_q \in B$. Coloring X in a checkerboard pattern and taking A to be one color 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 zero, 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.

All of these MRF formulations require the definition of parameters, such regularization weight, the values of which can dramatically affect the performance of the algorithm. These values often vary significantly from data set to data set, and must often be hard-coded by trial and error. In their comparison of graph cuts and belief propagation [5], 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 [6].

3. Multi-sensor Camera System

In this paper we use the Ladybug panoramic camera system [1] from Point Grey Research². The Ladybug, pictured in Figure 1, is comprised of six single-CCD (Bayer tiled) sensors. Each sensor is 1024×768 pixels. One sensor points vertically, while the other five point out radially.

Using wide-angle lenses the Ladybug’s sensors combine to view approximately three quarters of its surroundings (it does not have a sensor pointing down) with 80 to 100



Figure 1. The Point Grey Ladybug camera system.

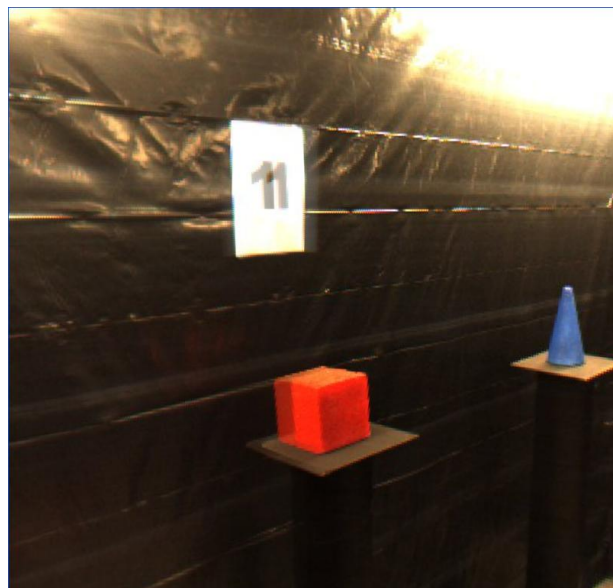


Figure 2. Simple blending and feathering produces ghosting or echoing effects, such as above, on objects that are near the camera and lie in the overlap region of adjacent Ladybug images.

²<http://www.ptgrey.com>

pixels overlapping in adjacent sensors. The wide-angle lenses induce substantial radial distortion, which make the images from the sensors impractical for use in image processing algorithms that rely on at least approximate pin-hole projection models, such as localization methods. Further, because the Ladybug's sensors do not share a common center of projection, nearby objects seen by adjacent sensors introduce parallax-related artifacts when the images are blended using simple alpha feathering, e.g. Figure 2.

Our objective is to resample the Ladybug images into a rectified, i.e. perspective correct, format without the parallax-induced artifacts. Kang et al. accomplish this using a method called multi-perspective plane sweep (MPPS) [13], which adjusts the center of projection from one sensor to the next while estimating the stereo disparity in the overlap region. This generates aligned, rectified images, but these images do not have a common center of projection. Instead we wish to generate a cubic panorama in the form of six axis-aligned, rectified images that do have a common center of perspective projection, as shown in Figure 3. This simplifies the task of determining the relative positions and orientations of two or more panoramas once they are generated. Also, our algorithm handles all images at once, whereas the MPPS operates pairs of images, and the top image from the Ladybug has to be handled as a special case.

Properly reprojecting the Ladybug images onto the faces of a cube requires accurate calibration information for the Ladybug's sensors. We use the Ladybug's API to retrieve the extrinsic calibration of each sensor (position and orientation relative to a common coordinate frame), but calibrate the intrinsic parameters ourselves. Calibrating the sensors' intrinsic parameters accurately is a challenging and interesting problem, however it is beyond the scope of this paper.

3.1. Image Sampling

To generate each side of a cubic panorama from the Ladybug images, we use a plane-sweeping technique: for $f = 0, \dots, L - 1$ we backproject each pixel p in the cube side to a depth $z(f)$ to obtain the 3D point \mathbf{P}_f , which is then reprojected into each input image from which it may be visible. The general idea is shown in Figure 3. Because of the overlapping sensor configuration, a point may project into one, two or three images. Due to the high distortion of the Ladybug images and the difficulty in properly calibrating such distortion, reprojections become less reliable the farther from the optical center of a sensor they get. Dropping the subscript f for clarity, let $\mathbf{s}_j(\hat{p})$ be the color sample taken from image j at the projection \hat{p} of \mathbf{P} onto that image. Let the visibility of \mathbf{P} from sensor j be $V_j(\mathbf{P}) = 1$ if it

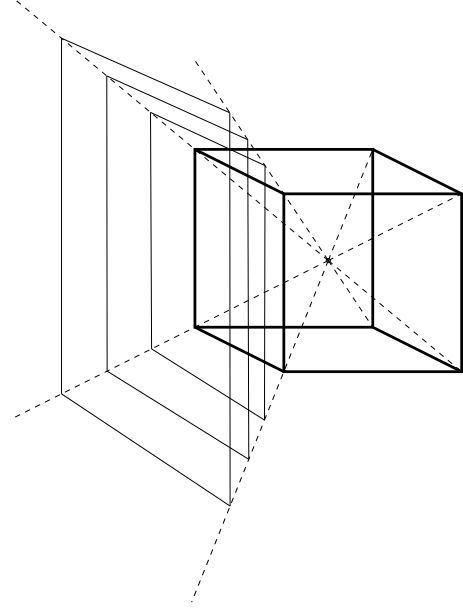


Figure 3. Plane sweeping for one side of a cubic panorama.

is visible, and 0 otherwise. Let the color of pixel p be the weighted average of samples from sensors from which \mathbf{P} is visible

$$\mathbf{c}_p(f) = \frac{1}{W} \sum_j V_j(\mathbf{P}) w_j(\hat{p}) \mathbf{s}_j(\hat{p}) \quad (10)$$

where $w_j(\hat{p}) = r_{bound}(\hat{p}) - r(\hat{p})$ is a weight based on the distance $r(\hat{p})$ of the reprojection from the optical center of sensor j and the distance from the same to the edge of the image, through \hat{p} , $r_{bound}(\hat{p})$; and $W = \sum_j V_j(\mathbf{P}) w_j(\hat{p})$.

We now define the reprojection cost of pixel p at depth $z(f)$ to be the photoconsistency measure

$$D_p(f) = \frac{1}{W} \sum_j V_j(\mathbf{P}) w_j(\hat{p}) (\bar{s}_j(\hat{p}) - \bar{c}_p(f))^2 \quad (11)$$

where \bar{s} and \bar{c} are the scalar luminances of RGB colors \mathbf{s} and \mathbf{c} respectively. Note that since all input images are used the same way and the photoconsistency cost (11) has linear complexity in the number of images, if this method was generalized to any number of images it would be considered a "true multi-image" matching technique as defined by Collins [14].

4. BP for Cube Generation

For numerical stability [4] we convert the MAP estimation from max-product to min-sum form and minimize the

energy function

$$\Gamma(\mathbf{F}) = \sum_{(p,q) \in E} U_{pq}(f_p, f_q) + \sum_p D_p(f_p) \quad (12)$$

where \mathbf{F} is a configuration or labelling with a label f_p for every node x_p , and $D_p(f) \propto -\ln(\phi_p(f))$ and $U_{pq}(f, g) \propto -\ln(\psi_{pq}(f, g))$ are the *data cost* and *discontinuity cost*, respectively. The message vector \mathbf{m}_{pq}^t is defined over each label g by

$$\mathbf{m}_{pq}^t(g) = \min_f \left(U_{pq}(f, g) + D_p(f) + \sum_{s \in N(p) \setminus q} \mathbf{m}_{sp}^{t-1}(f) \right) \quad (13)$$

where $N(p) \subset X$ is the first-order neighborhood of x_p .

After T iterations the belief vector \mathbf{b}_p is defined over each label f by

$$\mathbf{b}_p(f) = D_p(f) + \sum_{q \in N(p)} \mathbf{m}_{qp}^T(f). \quad (14)$$

The label f_p corresponding to the minimal component of \mathbf{b}_p is taken as the MAP solution for x_p .

For the data cost $D_p(f)$ we use the photoconsistency measure (11) described in Section 3.1. In the hierarchical scheme, for a node x_b^k in the level- k MRF corresponding to block b of 2^{2k} pixels, $D_b^k(f) = \sum_{p \in b} D_p(f)$.

For the discontinuity cost $U_{pq}(f, g)$ we follow the truncated linear model

$$U_{pq}(f, g) = \min(\lambda \|f - g\|, d_{pq}) \quad (15)$$

where λ is a scale factor and d_{pq} is the truncation threshold for discontinuity between pixels p and q . A discontinuity cost function like this one allows messages to be computed in two passes over the set of labels, making the complexity of computing a single message linear in L as opposed to quadratic in general [4].

The message passing iterations are divided into a forward pass and a backward pass over the set of possible labels. In the forward pass, for each label $f = 0, 1, \dots, L-1$ we compute

$$\mathbf{m}_{pq}^t(f) = \begin{cases} h_{pq}(f) & \text{If } f = 0 \\ \min(h_{pq}(f), \mathbf{m}_{pq}^t(f-1) + \lambda) & \text{otherwise} \end{cases} \quad (16)$$

where $h_{pq}(f) = D_p(f) + \sum \mathbf{m}_{sp}^{t-1}(f)$. In the backward pass, for each label $f = L-2, L-3, \dots, 0$ we compute

$$\mathbf{m}_{pq}^t(f) = \min(\mathbf{m}_{pq}^t(f), \mathbf{m}_{pq}^t(f+1) + \lambda) \quad (17)$$

$$\mathbf{m}_{pq}^t(f) = \min\left(\mathbf{m}_{pq}^t(f), \min_g h_{pq}(g) + d_{pq}\right). \quad (18)$$

At each iteration, we update in the above fashion only

the messages for the appropriate subset A or B of X based on whether the iteration number t is even or odd, respectively, as per the bipartite graph optimization described in Section 2. In the multi-level or hierarchical MRF setup, after T iterations at level k , the messages must be inherited to corresponding child nodes in the level- $k-1$ MRF.

5. Implementation

Belief propagation is well suited for parallel execution. While conceptually message updates are performed in parallel, a single CPU will perform the computations sequentially. Graphics processing units (GPUs) are highly parallel single-instruction-multiple-data (SIMD) processors built into modern graphics cards along with up to 256 or even 512 MB of high-speed memory. Vertex and fragment programs, or shaders, allow developers to perform custom, complex arithmetic and texturing operations in hardware. GPUs and their programmable interface have become so versatile that much research has been done on performing general purpose computation in graphics hardware (GPGPU). For many such examples, the interested reader is referred to <http://www.gpgpu.org>. Core computer vision algorithms, such the Canny edge detector and feature extraction, have been implemented for the GPU [8]. Gong and Yang use image gradients in their real-time stereo algorithm for the GPU [9] and Yang et al. presented a plane-sweep algorithm on the GPU for real-time view synthesis [10] that is similar to the process presented here for generating a side of a cubic panorama.

By storing the message data and the data cost in textures, we can use fragment programs to perform the message update scheme described in Section 4 in a much more parallel way than in a CPU implementation. (Newer GPUs can process as many as 16 or 24 fragments in parallel.) Each message is a vector of length L , and each node x_p sends a message to each of its four neighbors. By component-wise interleaving the messages from x_p to each of its neighbors we can store corresponding components of those messages in a four-channel floating-point texture element. Thus, the message data for a given label is stored in a single texture. A one-channel floating-point texture is used to store the data cost for each label.

The storage complexity for this algorithm is $O(ML)$, where M is the number of output pixels. For most binocular stereo data sets, modern graphics cards have enough on-board memory to store all the required data. However, for a cube side that is 1024×1024 pixels, the message data alone requires $L \times 4 \times 1024 \times 1024$ 32-bit floats. For 16 labels this is 256 MB. Hence, we keep in graphics memory the message data for only a cached subset of labels, while maintaining the full message data in main memory. We swap

data to and from main memory as needed by the algorithm, thus greatly improving the scalability of this approach. The bipartite graph optimization described in Section 2 allows us to further cut in half the amount of message data that must be kept in graphics memory by packing the message data as shown in Figure 4, which also halves the amount of message data that must be transferred to and from main memory each iteration. In the hierarchical scheme, at level k , message data for 2^{2k} labels can be stored in the same size texture as stores the message data for a single label at level 0. The same goes for the data cost. This in turn cuts the data transferred each iteration by a factor of 2^{2k} . Packing the data in texture memory like this also improves the texture cache performance of the GPU for $k > 0$.

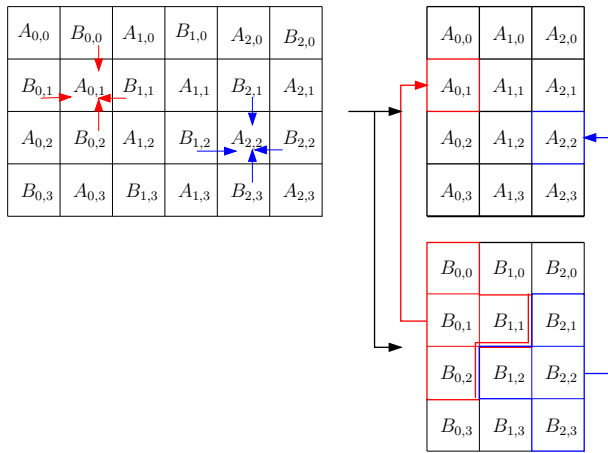


Figure 4. Bipartite partitioning of the image grid. Red arrows indicate messages incoming to $A_{0,1}$ and blue arrows indicate messages incoming to $A_{2,2}$. The left side shows how the nodes are located relative to one another in the MRF, while the right side shows how incoming messages from the previous iteration are read with the bipartite optimization in effect.

We compute the forward and backward passes of the message passing iterations described in Section 4 by rendering a screen-aligned rectangle textured with the necessary input fields. For each label in the forward pass, for every pixel p and each of its neighbors q , textures storing $\mathbf{m}_{pq}^{t-1}(f)$, $\mathbf{m}_{pq}^t(f-1)$ and $D_p(f)$ are applied and a fragment program computes $\mathbf{m}_{pq}^t(f)$ per (16) and stores it in another texture. Figure 5 illustrates how the fragment program computes the intermediate quantity $h_{pq}(f)$ in (16).

Similarly, a fragment program is used to compute (17) and (18) for each label in the backward pass from textures storing $\mathbf{m}_{pq}^t(f)$, $\mathbf{m}_{pq}^t(f+1)$, $\min_g h_{pq}(g)$ and optionally d_{pq} , which is otherwise constant. The texture stor-

ing $\min_g h_{pq}(g)$ is written in the forward pass fragment program using the ability of the GPU to render to multiple textures at once.

Figures 4 and 5 show how the forward pass fragment program handles the texture addressing for $\mathbf{m}_{sp}^{t-1}(f)$ under the bipartite grid packing scheme by switching texture coordinate offsets based on whether p is in an even or odd row in the image. Other calculations are made in both the forward and backward pass fragment programs to offset the texture look-ups to handle multiple labels within the same texture for $k > 0$. By setting the viewport, we ensure that only the appropriate rectangular region of the output texture is updated for a given level k and label f .

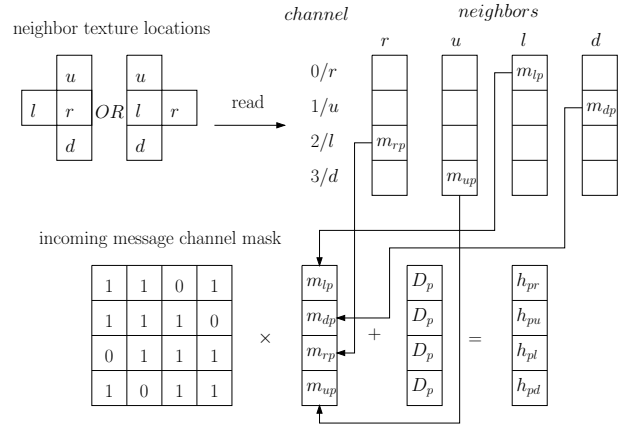


Figure 5. Combining incoming messages to compute outgoing message. Beginning at the top-left with the bipartite neighbor pattern for the given node x_p , messages from right, up, left and down neighbors are read into four separate registers, combined into one incoming message register, which is then multiplied by the incoming channel mask and then added to the data cost.

We also make use of the GPU for the image sampling described in Section 3.1, with fragment programs to compute $\mathbf{c}_p(f)$ and $D_p(f)$ according to (10) and (11) respectively.

6. Results

Figures 6 and 7 are the faces of cubic panoramas generated for an indoor scene and an outdoor scene respectively. The overlapping regions of the input images have not been blended so the reader can see where they are. The front face in Figure 6 is generated from the same input images as were blended and feathered to produce Figure 2. Note that

the echo artifacts on the number 1 and on the red block in front of it have been removed.

Figure 7 illustrates the results of applying the method to an outdoor scene. Blending and feathering performs nearly as well in this case since objects in the scene are generally far from the view. Space constraints prevent a comparison figure, but Figure 7 is intended only to show the the proposed method handles such scenes without modification from indoor scenes.

Notable remaining artifacts, particularly the light in the rear-view of Figure 6, result from a lack of overlap (i.e. correspondence information) in the view and some combination of numerical instability, calibration errors and noise. Work is ongoing to rectify such errors.

We also applied GPU accelerated BP to standard stereo data sets, which require less storage and can therefore be executed entirely on the GPU without the need to swap data to and from main memory. The GPU implementation shows a promising speed-up, averaging 0.489 s on a NVidia GeForce 6800 GT to produce a MAP disparity estimate comparable to that produced by [4] on a Pentium 4 3.4 GHz in an average of 1.189 s. Both averages were computed over 20 trials under similar conditions. It should be noted that the method of Felzenszwalb and Huttenlocher runs many times faster than other stereo algorithms that produce comparable results.

For generating cube sides, data must be swapped between graphics and main memory, and the GPU implementation performs much more comparably to the original CPU version. To perform the message passing iterations for a cube side, the CPU implementation takes around 20 s, while the GPU implementation takes more like 17 s for the message passing iterations, and up to 25 s to generate the data cost *and* execute the message passing iterations. The main bottleneck is reading back data from the graphics card to main memory. In fact, our experiments show this to be up to 10 times slower than sending data to the graphics card.

7. Summary and Conclusions

We have presented a novel approach to generating panoramic images from a multi-sensor camera system, borrowing the success of Markov random fields and belief propagation as applied to the similar problem of stereo matching. We have shown a significant running time improvement, nearly 60 percent, over the CPU implementation of belief propagation algorithm of Felzenszwalb and Huttenlocher for stereo, which in turn is an order of magnitude faster than any other published stereo method (to our knowledge) that produces comparably accurate results. The speed advantage of the GPU decreases on the larger data sets for panorama generation, when data must be swapped

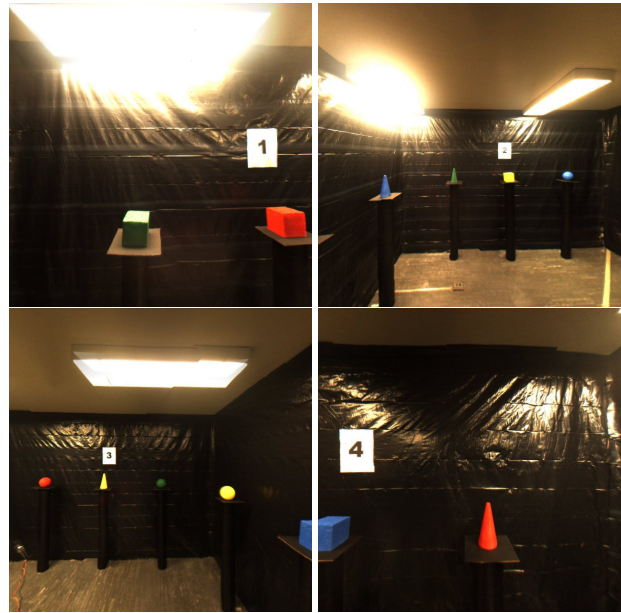


Figure 6. The front, right, back and left faces of a cubic panorama of an indoor environment generated using the proposed method (the top and bottom faces are omitted to save space since they are not very interesting).



Figure 7. The front, right, back and left faces of a cubic panorama generated for an outdoor scene (the top and bottom of the cube are omitted to save space since they are not very interesting).

to and from main memory.

The remaining artifacts may be fixed by improved calibration, or by a more connected model. Currently, each side is considered separately, and since photoconsistency information is available only in the overlap regions, the sides are not guaranteed to align perfectly. A possible improvement would be to consider a fully connected panoramic MRF, for example a cubic MRF. A cubic MRF is topologically different than a planar MRF and the bipartite optimization could not be used. Research into this method is on-going.

Further considerations also include assuring temporal consistency in a panoramic sequence, as well as other types panoramic images. This method could be modified to generate spherical or cylindrical panoramas.

Acknowledgments

We would like to thank Dr. Eric Dubois of the University of Ottawa for his expertise and suggestions regarding Markov random fields. This work was funded through the NAVIRE project of the VIVA Lab of the School of Information Technology and Engineering at the University of Ottawa.

References

- [1] *Ladybug Spherical Digital Video Camera System User Manual and API Reference*, Version 1.1, 2002-2003, Point Grey Research, Inc.
- [2] Jonathan S. Yedidia, William T. Freeman and Yair Weiss, "Understanding Belief Propagation and its Generalizations", *International Joint Conference on Artificial Intelligence IJ-CAI 2001*, 2001.
- [3] 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.
- [4] 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.
- [5] 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 (ICCV 2003)*, pp. 900–906, 2003.
- [6] 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.
- [7] 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.
- [8] James Fung, "Computer Vision on the GPU", *GPU Gems 2*, edited by Matt Phar, NVidia/Addison-Wesley, Toronto, pp. 649–666, 2005.
- [9] Minglun Gong and Ruigang Yang, "Image-gradient-guided Real-time Stereo on Graphics Hardware", *Proceedings of 3DIM 05*, pp. 548–555, 2005.
- [10] Ruigang Yang, Greg Welch and Gary Bishop, "Real-Time Consensus-Based Scene Reconstruction using Commodity Graphics Hardware", *Pacific Graphics 2002*, pp. 255–234, 2002.
- [11] Shenchang Eric Chen, "QuickTime VR - An Image-Based Approach to Virtual Environment Navigation" *Proceedings of ACM SIGGRAPH 95*, pp. 29–38, 1995.
- [12] M. Brown and D. G. Lowe, "Recognising Panoramas" *International Conference on Computer Vision (ICCV 2003)*, pp. 1218–1225, Nice, France, 2003.
- [13] Sing Bing Kang, Richard Szeliski, Matthew Uyttendaele, "Seamless Stitching using Multi-Perspective Plane Sweep", *Microsoft Research Technical Report*, MSR-TR-2004-48.
- [14] R.T. Collins, "A space-sweep approach to true multi-image matching", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 358–363, San Francisco, CA, June, 1996.