Belief Propagation on the GPU for Stereo Vision

Alan Brunton
School of Information Technology and Engineering (SITE), University of Ottawa
Ottawa, Ontario
e-mail: abrunton@site.uottawa.ca

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

Gerhard Roth
Institute for Information Technology, National Research Council of Canada
Ottawa, Ontario
e-mail: gerhard.roth@nrc-cnrc.gc.ca

Abstract

The power of Markov random field formulations of low-level vision problems, such as stereo, has been known for some time. However, recent advances, both algorithmic and in processing power, have made their application practical. This paper presents a novel implementation of Bayesian belief propagation for graphics processing units found in most modern desktop and notebook computers, and applies it to the stereo problem. The stereo problem is used for comparison to other BP algorithms.

Keywords: stereo vision, image-based rendering, belief propagation, graphics processing unit.

1 Introduction

This paper describes a programmable graphics hardware implementation of belief propagation algorithms for stereo vision. By stereo vision we refer to the classical dense stereo problem of computing the depth or disparity of each pixel in a reference image, by matching them with pixels in supporting images of the same scene from different positions.

We represent a Markov random field (MRF) by a set Φ of potential functions over cliques in an undirected graph $G = (V, E)$ where the set of nodes V each represent 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. The set V is comprised of two subsets: $Y = \{y\}$ represent observed quantities of the scene and $X = \{x\}$ represent hidden quantities about the scene, the value of which we wish to infer. For each pixel p in the image there exists an observed node y_p and a corresponding hidden node x_p, with $(x_p, y_p) \in E$. We assume there are equal numbers of hidden and observed nodes. The hidden nodes $\{x\}$ are connected in a 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* $\phi(x_p, y_p)$ and the *compatibility matrix* $\psi(x_p, x_q)$ as the joint probability of an observed node and its corresponding hidden node, and the joint probability of two neighboring hidden nodes, respectively. We can write the overall joint probability of all nodes in V as [3]

$$P(X, Y) = c_{X,Y} \prod_{(p,q)\in E} \psi(x_p, x_q) \prod_{p\in V} \phi(x_p, y_p) \quad (1)$$

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

$$P(X|Y) = c_{X|Y} \prod_{(p,q)\in E} \psi(x_p, x_q) \prod_{p\in V} \phi(x_p, y_p) \quad (2)$$

where $c_{X|Y} = \frac{c_{X,Y}}{P(Y)}$ is another normalization constant. From here there are two possible objectives: maximizing the marginal probabilities for each hidden node x using the minimum mean squared error (MMSE) estimator or computing the maximum a posteriori (MAP) estimator.

MRF formulations of stereo and other early vision problems have existed for some time, but the computational and storage complexity make exact solutions infeasible; exact solutions to both the MMSE and MAP estimators are NP-complete, hence the need for approximation algorithms such as graph cuts and belief propagation. As processing power has increased, the approximation algorithms have become practical for stereo.

Both graph cuts (e.g. [9]) and belief propagation (BP) solve for the MAP estimator by energy minimization or direct calculation of probability densities. Belief propagation can also be used for the MMSE esimator [2, 7]. Tappen and Freeman performed a comparison of both algorithms as applied to the stereo problem [7] 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

Proceedings of the 3rd Canadian Conference on Computer and Robot Vision (CRV'06)
0-7695-2542-3/06 $20.00 © 2006 IEEE

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.

Each random variable X_p represented by the node x_p can take any one of L values or *labels*. In the case of stereo, labels are disparity or depth levels. Let $\phi_p(f) = \phi(X_p = f, y_p)$ be the probability that node x_p is labelled f and let $\psi_{pq}(f,g) = \psi(X_p = f, X_q = g)$ be the probability that node x_p is labelled with f and neighboring node x_q is labelled with g, for $f, g \in \{0, 1, ..., L-1\}$. BP estimates a MAP labelling for the MRF by sending a message $\mathbf{m}_{pq}^t$ from every hidden node x_p to each of its (hidden) neighbors x_q at every 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 be have the corresponding label. BP Algorithms that perform MAP or MMSE estimation are referred to as "max-product" and "sum-product" respectively because of how the messages are updated. In the max-product algorithm messages are updated in the following way [4, 7]

$$\mathbf{m}_{pq}^t(g) = \kappa \max_f \left(\psi_{pq}(f,g)\, \phi_p(f) \prod_{s \in N(p) \setminus q} m_{sp}^{t-1}(f) \right) \tag{3}$$

where κ is a normalization constant. After T iterations, the beliefs are computed [4, 3]

$$\mathbf{b}_p(f) = \kappa \phi_p(f) \prod_{q \in N(p)} m_{qp}^T(f) \tag{4}$$

and the MAP labelling for node x_p is

$$f_p^{MAP} = \arg\max_f \mathbf{b}_p(f). \tag{5}$$

Messages in the sum-product algorithm are computed as follows [3]

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

and the belief is computed the same as in the MAP estimator.

Sun et al [4] achieve a speed-up 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 presented [5] three algorithmic techniques to substantially improve the running time of BP for early vision problems. 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 2.1, 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 were initialized closer to their point of convergence, they should take less time 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 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, 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 these paramters [8].

Belief propagation is well suited for parallel execution and hardware implementation [4]. 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 progams, 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 [10]. Gong and Yang use image gradients in their real-time stereo algorithm for the GPU [11] and Yang et al. presented a plane-sweep algorithm on the GPU for real-time view synthesis [12].

Section 2 describes the proposed BP algorithm and its implementation. Section 3 discusses the results obtained and compares them to similar methods. Section 4 draws some conclusions based on the results and discusses areas for future work.

2 BP on the GPU

We propose a modification of the hierarchical or "Multiscale" BP algorithm presented by Felzenszwalb and Huttenlocher [5] for solving the stereo problem. Using the stereo problem as illustrative example, we incorporate edge and occlusion information to demonstrate the use of additional cues and intermediate results within the hierarchical BP algorithm. To preserve clarity, we consider here only binocular, narrow-baseline stereo of rectified images. To simplify the implementation slightly and aide numerical stability we convert the problem from max-product to min-sum by working with the negative logarithm of the probabilities. Section 2.1 describes the modified hierarchical BP algorithm while Section 2.2 details the GPU implementation.

2.1 BP Algorithm

In min-sum form we minimize the energy function

$$\Gamma(\mathbf{F}) = \sum_{(p,q)\in E} U_{pq}(f_p, f_q) + \sum_p D_p(f_p) \qquad (6)$$

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 $\mathbf{m}^t_{pq}$ is defined over each label g by

$$\mathbf{m}^t_{pq}(g) = \min_f \left(U_{pq}(f,g) + D_p(f) + \sum_{s\in N(p)\setminus q} \mathbf{m}^{t-1}_{sp}(f) \right) \qquad (7)$$

where $N(p)$ is the first-order neighborhood of x_p (not including y_p).

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

$$\mathbf{b}_p(f) = D_p(f) + \sum_{q\in N(p)} \mathbf{m}^T_{qp}(f). \qquad (8)$$

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 sum of squared differences between the two images or SSD over a 3×3 window centered on the pixel p. In the hierarchical scheme, for a node x^k_b in the level-k MRF corresponding to block b of 2^{2k} pixels, $D^k_b(f) = \sum_{p\in b} D_p(f)$.

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

$$U_{pq}(f,g) = \min(\lambda \|f-g\|, d_{pq}) \qquad (9)$$

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 as is the case in the standard BP algorithm [5].

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 $f = \{0, 1, ..., L-1\}$ we compute

$$\mathbf{m}^t_{pq}(f) = \begin{cases} h_{pq}(f) & \text{If } f = 0 \\ \min\left(h_{pq}(f), \mathbf{m}^t_{pq}(f-1) + \lambda\right) & \text{otherwise} \end{cases} \qquad (10)$$

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

$$\mathbf{m}^t_{pq}(f) = \min\left(\mathbf{m}^t_{pq}(f), \mathbf{m}^t_{pq}(f+1) + \lambda\right) \qquad (11)$$

$$\mathbf{m}^t_{pq}(f) = \min\left(\mathbf{m}^t_{pq}(f), \min_g h_{pq}(g) + d_{pq}\right). \qquad (12)$$

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 1.

It has been observed [4, 1] that occlusion is a long-ranging interaction beyond the capacity of a pairwise MRF to model. Hierarchical BP was proposed [5] with the express purpose of facilitating long-range flow of information across the Markov network, and we exploit this for the purpose of occlusion modelling in addition to faster convergence. Tappen and Freeman found [7] that between sixty and eighty percent of the energies of ground-truth disparity maps for four standard stereo data sets were due to occluded pixels. Since the correct labelling for these nodes adds significant energy to the system, using the data cost for these nodes is likely to adversely affect the solution.

The BP algorithm generally proceeds by iterating for some number of iterations T and then selecting the label f_p that minimizes the belief vector $\mathbf{b}_p$ for node x_p. However, following the backward pass we have all the information needed to compute the belief vector and the minimizing label. Thus, for an even iteration, after computing the messages from nodes $x_q \in A$ for each label f, we compute the belief for nodes $x_p \in B$ by (8) and the label f_p with minimal belief for x_p. For an odd iteration, we analogously

Proceedings of the 3rd Canadian Conference on Computer and Robot Vision (CRV'06)
0-7695-2542-3/06 $20.00 © 2006 IEEE

compute the messages from $x_p \in B$ and then the belief for nodes $x_q \in A$ and f_q.

We thus have a intermediate disparity map after each iteration, and we can use this to test for occlusion before computing the next iteration of messages. If a pixel is occluded (in the right image), we ignore its data cost by multiplying by a visibility mask O. If pixel p is occluded, $O(p) = 0$; otherwise $O(p) = 1$. Drouin et al observed experimentally [1] that many stereo algorithms have a tendency to overestimate the disparity of occluded pixels, making close objects larger, and causing a misclassification of pixels: occludees as occluders, occluders as regular pixels and regular pixels as occludees. Drouin et al note that this observation discourages the direct use of visibility as a mask and compensate for this bias by labelling both occluders and occludees as invisible. Hence, $O(p) = 0$ if p either occludes another pixel or is occluded by another pixel.

Given O is computed from an intermediate disparity map, which contains errors, we use a simple, non-exhaustive occlusion test, and recompute O every iteration. Also note that as we perform this occlusion masking at higher levels of the hierarchy as well as the full-resolution grid, we do not want pixels to be masked-out permanently as a result of occlusion involving a higher-level block to which those pixels belong. The occlusion test is performed for every hidden node x_p at the current level of the hierarchy. The change in disparity from x_p to its left neighbor x_l and to its right neighbor x_r is computed as $\Delta_- = f_p - f_l$ and $\Delta_+ = f_p - f_r$ where f_l and f_r are the disparity levels computed at x_l and x_r in the previous iteration, respectively. Let x_- and x_+ be the nodes $|\Delta_-|$ to the left and $|\Delta_+|$ to the right of x_p, respectively and let their disparities be f_- and f_+. If $f_- + \Delta_- = f_p$ and $\Delta_- \geq 1$, then pixel p is an occluder and $O(p) = 0$; if $f_+ + \Delta_+ = f_p$ and $\Delta_+ \leq -1$ then pixel p is an occludee and $O(p) = 0$. Otherwise, $O(p) = 1$.

2.2 Implementation

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 2.1 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 of this algorithm is $O(ML)$, where M is the number of pixels, and for standard dense stereo data sets this fits on newer graphics cards. The bipartite graph optimization described in Section 1 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 2.2.

We compute the forward and backward passes of the message passing iterations described in Section 2.1 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 (10) and stores it in another texture. Figure 2.2 illustrates how the fragment program computes the intermediate quantity $h_{pq}(f)$ in (10).

Similarly, a fragment program is used to compute (11) and (12) 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 storing $\min_g h_{pq}(g)$ is written in the forward pass fragment program using the ability of the GPU to render to multiple textures at once.

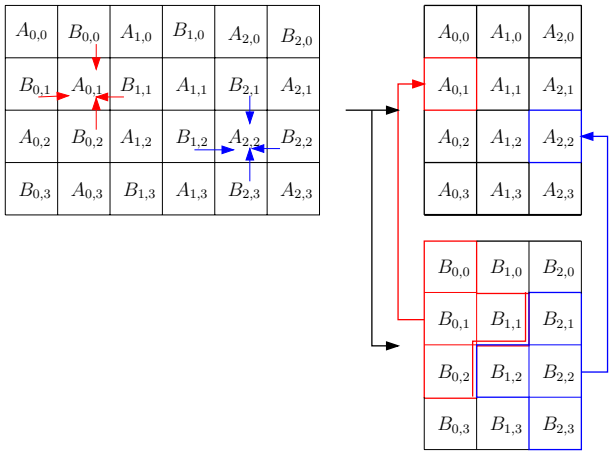

Figure 1. Bipartite partitioning of the image grid. Red arrows indicate messages incoming to $A_{0,1}$ and blue arrows indicate messages incoming to $A_{2,2}$. The left side shows how the nodes are located relative to one another in the MRF, while the right side shows how incoming messages from the previous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

Figures 2.2 and 2.2 show how the forward pass fragment

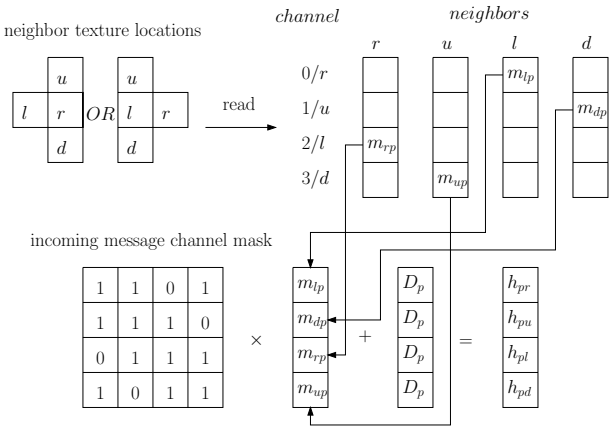

Figure 2. Combining incoming messages to compute outgoing message. Beginning at the top-left with the bipartite neighbor pattern for the given node x_p, messages from *r*ight, *u*p, *l*eft and *d*own neighbors are read into four separate registers, combined into one incoming message register, which is then multiplied by the incoming channel mask matrix and then added to the data cost.

Figure 3. Disparity map generated for the Tsukuba data set using a GPU implementation of the algorithm by Felzenzwalb and Huttenlocher.

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 multi-level MRFs within the same texture in the hierarchical version. 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.

3 Results

Figure 3 shows the results of using a five-level hierarchy with six iterations at each level to compute the disparity map for the Tsukuba data set [14], without incorporating occlusion or edge information. Figure 4 shows results for the same trial with occlusion and edge information incorporated.

The running time for the GPU version of the modified BP algorithm was compared to the original version without edge and occlusion information [5] on a 3.4 GHz Pentium 4, using an NVidia GeForce 6800 GT with 256 MB of video memory and a PCI-Express bus. Over twenty trials, the original [5] averaged 1.189 s, while the GPU version averaged 0.610 s. The GPU version with occlusion handling averaged 1.661 s. Using the error measure $B_{\hat{O}}$ [13], the GPU version achieve an error rate of 3.6 (original algorithm) and 3.3 (incorporating occlusion information) percent erroneous disparities.

4 Conclusion

We have presented a novel implementation of a belief propagation algorithm to run on programmable graphics hardware, which applied to stereo vision produces fast, accurate results. The algorithm incoporates side information and intermediate results, in the form of edge and occlusion information for stereo. The scalability of the algorithm presented allows it to be applied other applications.

Areas of interest for future work are numerous. They include the incorporation of information about the prior distribution, and enumerating local minima using local optimization method such as randomized gradient descent and using the local minima as the labels for the belief propagation algorithm. Incorporating parameter estimation [8], improving the edge information used, using the sum-product BP algorithm to compute the MMSE estimator and applying belief propagation on the GPU to other image processing and computer vision problems, such as superresolution, would also be interesting areas of investigation. On going work involves applying this technique to generating panoramic images from multi-sensor cameras.

Figure 4. Disparity map generated for the Tsukuba data set using occlusion and edge information. Computed entirely on the graphics card.

References

[1] 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, June 2005.

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

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

[4] 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, July 2003.

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

[6] 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, 2004.

[7] Marshal F. Tappen and William T. Freeman, *"Comparison of Graph Cuts with Belief Propagation for Stereo, using Identical MRF Parameters", Internation Conference on Computer Vision*, 2003.

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

[9] 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, November 2001.

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

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

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

[13] 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.

[14] 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.