Global Stereo in Polynomial Time

Carlo Tomasi
Duke University

1 Introduction

Stereo vision is one of the most thoroughly studied problems in computer vision. In a survey of the state of the art [23], and more recently on the accompanying Middlebury stereo vision web page, Scharstein and Szeliski categorize and compare a wide array of algorithms that follow approaches ranging from window-level correlation to dynamic programming, to combinatorial optimization, to belief propagation.

This survey shows that the problem of stereo matching is relatively easy to formulate, but hard to solve. More than thirty years after the first attempts on computers [12, 19], and in spite of the existence of several experimental or commercial hardware systems that match stereo image pairs [14, 17, 27, 2] most computer vision conferences still have entire sessions on this thriving topic. If the problem were solved, this would not be the case.

A public data benchmark like the Middlebury web site is a precious resource for comparing approaches, coalescing efforts, and eventually advancing the state of the art. We need more work in this direction. Because of this benchmark, whether one algorithm is better than another is not a matter of judgement, but can be measured by the number of correct matches for the two methods on the same image pairs.

One clear lesson learned from past work is that stereo is best formulated as a global problem, in which an entire image is matched to another one as the result of a single optimization problem. The two common alternatives are to match one point at a time (the favorite hardware solution), or two corresponding epipolar lines at a time. The latter is a reasonable compromise between the greediest solution and the completely global one, and has been traditionally addressed with dynamic programming (see [1, 20, 3, 9, 4, 26] and many more).

The great advantage of dynamic programming is its conceptual and algorithmic simplicity. This has allowed recasting stereo matching into a Bayesian formulation [3] and a Maximum-likelihood approach [9] as mathematical sophistication in computer vision progressed, to propose variations based on different similarity metrics [4], and to explore efficient multi-scale implementations [26]. Less tangibly, but perhaps not less importantly, when a problem that is simple to state has a solution that is simple to explain, we feel that we are on the right track: if something goes wrong it is easier to fix, and when things go right we know why and what to retain when attempting improvements. Good performance in the experiments is of course the ultimate indicator of a good idea, but until we can claim perfect performance an improvement that comes at the expense of understanding is of somewhat reduced heuristic value.

This is why I have not been able to experience the same sense of satisfaction with the completely global solutions that perform best on the Middlebury web page, whether I contributed to them [5, 18] or not [28, 16, 15, 24]. These solutions are all complex in two ways. First, the exact problem in all these formulations is NP hard, and approximations (still relatively expensive) must be used. Second, algorithms that use sophisticated heuristics to search for an approximate optimum in a large space are difficult to understand intimately. When something goes wrong, it is often hard to tell whether the culprit is in the necessary approximations introduced for tractability or rather an inappropriately defined optimization target. When things go well, it is not clear if a small modification to the algorithm, meant to improve some aspect of its solution, would retain good performance in other aspects.

Fortunately, the dilemma between a simple but line-by-line solution to stereo and a more global solution can be resolved in a different way, through a formulation based on maximum flows in networks. This solution has been shown in the literature in somewhat different versions [22, 13, 21, 6]. The version in [22] was even featured on the Middlebury web page, where it performed respectably (2.98 percent of pixels with disparity wrong by more than one pixel on the Tsukuba image), although not among the "stars."

In a nutshell, as shown most elegantly in [6], instead of finding a shortest path on the usual graph used in the

dynamic programming formulations, one first transforms this graph into its graph-theoretical dual, and *voilà*, the shortest path becomes a minimum cut. As well known, a minimum cut can be found by computing a maximum flow through the graph. If one considers one scan line at a time, this problem transformation is just an exercise in graph theory. However, if the two images in the stereo pair are considered in their entirety, the dynamic programming solution ceases to be applicable, while the minimum cut formulation still admits a polynomial-time solution. This is also practically very fast, thanks to Cherkassky and Goldberg's implementation [8] of the Goldberg-Tarjan max-flow algorithm [11].

The max-flow/min-cut solution to the stereo correspondence problem has not received the attention that it deserves. Few others work on it, and the the approach is rarely taught in computer vision courses or books. Part of the reason may be the method's good but not stellar performance on the benchmarks. Another part may have to do with the fact that the network flow formulation is apparently less flexible than the one based on dynamic programming, as it seems more difficult to incorporate some of the heuristic constraints that we are used to seeing in stereo vision algorithms. A third part of the explanation may be that previous formulations based on network flow may be more complex than they need to be.

The goal of this contribution is to introduce an even simpler max-flow/min-cut formulation of stereo correspondence than previously published, in the hope that the very simplicity of the approach will spur the types of studies of this method that have been devoted to dynamic programming for the line-by-line case. Specifically, Section 2 shows why dynamic programming cannot be applied to the global stereo correspondence problem. Section 3 then introduces the proposed formulation of global correspondence as a max-flow/min-cut problem, and compares it to previous ones. Finally, Section 4 shows anecdotal evidence of good performance, and Section 5 gives some indications for future work.

2 Dynamic Programming Stereo and Its Limits

Assume that pixels in a given scan line of the left image always correspond to pixels in the same scan line of the right image, and viceversa. This assumption entails no loss of generality, because stereo images can be always rectified so that this is the case [25]. Figure 1 shows the standard graph used in dynamic-programming formulations of the line-to-line stereo correspondence problem.

If the pixels of the right scan line are listed at integer positions along the horizontal axis of a Cartesian reference frame and those of the left scan line are listed at integer positions along the vertical axis, as shown in Figure 1, then a match between two pixels can be represented by a point on the plane with integer coordinates. Thus, the integer grid on the plane is the set of all possible matches. In the rectified camera configuration, a pixel in the left scan line can only match a pixel in the right scan line that has a lower horizontal image coordinate: points in the image from the right camera are shifted to the left relative to where they are in the left camera. This restriction eliminates all points of the grid that are above the main diagonal, labelled *disparity 0* in Figure 1. Points on this diagonal have a disparity of zero, which means that their horizontal image coordinates are equal. The ones on the next diagonal have a disparity of one, that is, their coordinates differ by one pixel, and so forth. Greater disparity corresponds to points that are closer to the camera, so disparities can also be bounded from above if one assumes that no object in the world can come closer than a given distance from the two cameras. This results in the trapezoidal shape of the grid in Figure 1.

A list of all the actual matches between the two scan lines yields a *path* on the grid, like the thick blue line in Figure 1, which is assumed to start at the top left of the grid, and end at the bottom right (dummy, compulsory matches can be added at the corners to enforce this). Not all paths correspond to valid solutions to the stereo matching problem. For instance, some paths may represent surfaces that hide themselves or each other from either camera, a situation that cannot occur in reality. To eliminate these impossible paths from the set of solution, valid paths are often constrained to only go monotonically from top left to bottom right. Thus, the blue path in Figure 1 is valid, because it is only made of segments that when the path is traversed from top to bottom go vertically down, horizontally to the right, or diagonally down and to the right. In contrast, the short, thick, orange segment would not be a valid path segment, because it goes diagonally down and to the left. Physically, this segment would connect two matches in which the order of the two points in the left image is opposite to the order of the two points in the right image (see the order of the thick and thin lines at the endpoints of the segment). Such pairs of matches are not impossible, but they are rare. Forbidding them results into the *ordering constraint*, which restricts the set of solution somewhat more than necessary,

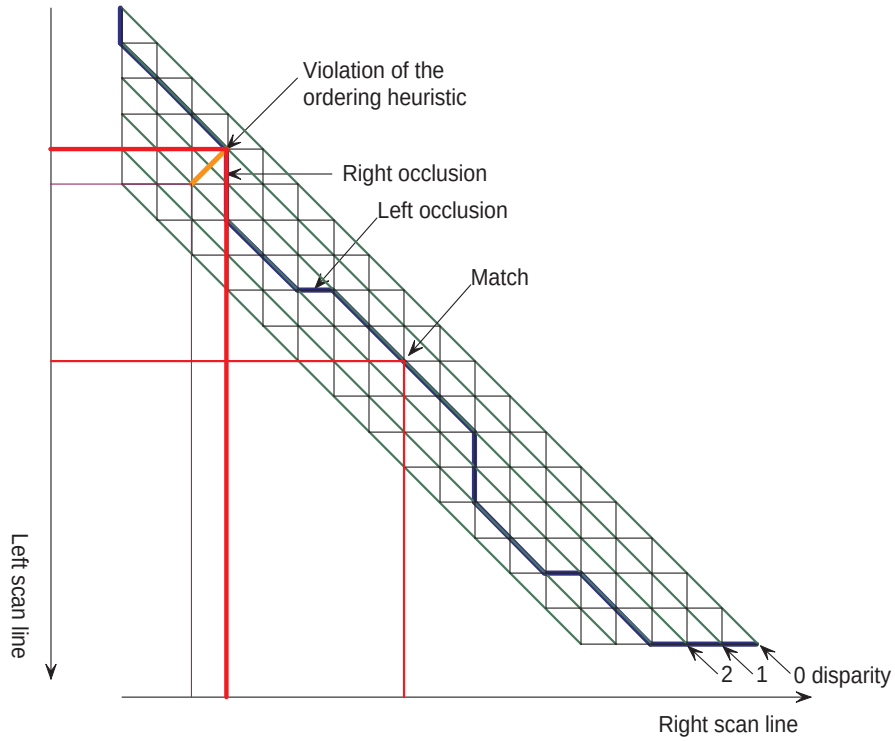

Figure 1: The standard graph used in dynamic-programming formulations of the line-to-line stereo correspondence problem. See text for explanations.

but in a way that is very easy to enforce, especially when using dynamic programming.

A horizontal segment of the path corresponds to right pixels that have no corresponding left pixels, that is, to a point in the left image where disparity changes abruptly. This is called a *left occlusion*. A *right occlusion* is defined similarly, with the role of the two images reversed.

When formulating the stereo correspondence problem within the framework of dynamic programming, one typically defines a *cost* for each candidate match, which is an increasing function of the difference in the pixel values at the two image points being matched: similar pixels are more likely to match than dissimilar ones. A cost is also typically charged to occlusions, both left and right, according to the observation that most surfaces in the world are smooth, that is, that disparity changes only slowly from point to point most of the time. One can then look for a path that satisfies the ordering constraint and has overall minimum cost. This is done very efficiently with *dynamic programming*.

In this style of solution, sketched in figure 2, a frontier (some curve that splits the start point from the end point of the path) is moved down the grid one step at a time. *For each position of the frontier, dynamic programming records the cost of the least expensive path up to each of the grid points on the frontier*. Because of the associative nature of the measure of the cost of a path (equal to the sum of the costs incurred along the path), the path costs recorded on the frontier can be updated efficiently when the frontier is moved. Once the frontier meets the bottom right of the grid, the cost of the best path from start to end is known. Additional information stored during the procedure allows to quickly reconstruct the best path that leads to this optimal cost.

The details of dynamic programming are not important for our purposes. The crucial observation is highlighted in the previous paragraph: a temporary variable is associated to each point on the frontier in order to store the cost of the best path up to that point. This is feasible because both the frontier and any path are polygonal curves on the plane, whose vertices are the integer coordinates of the grid points: two curves intersect at a point, and any frontier has a small number of possible intersection points.

If we now were to extend this principle of solution to full images, the match grid would become three-dimensional,

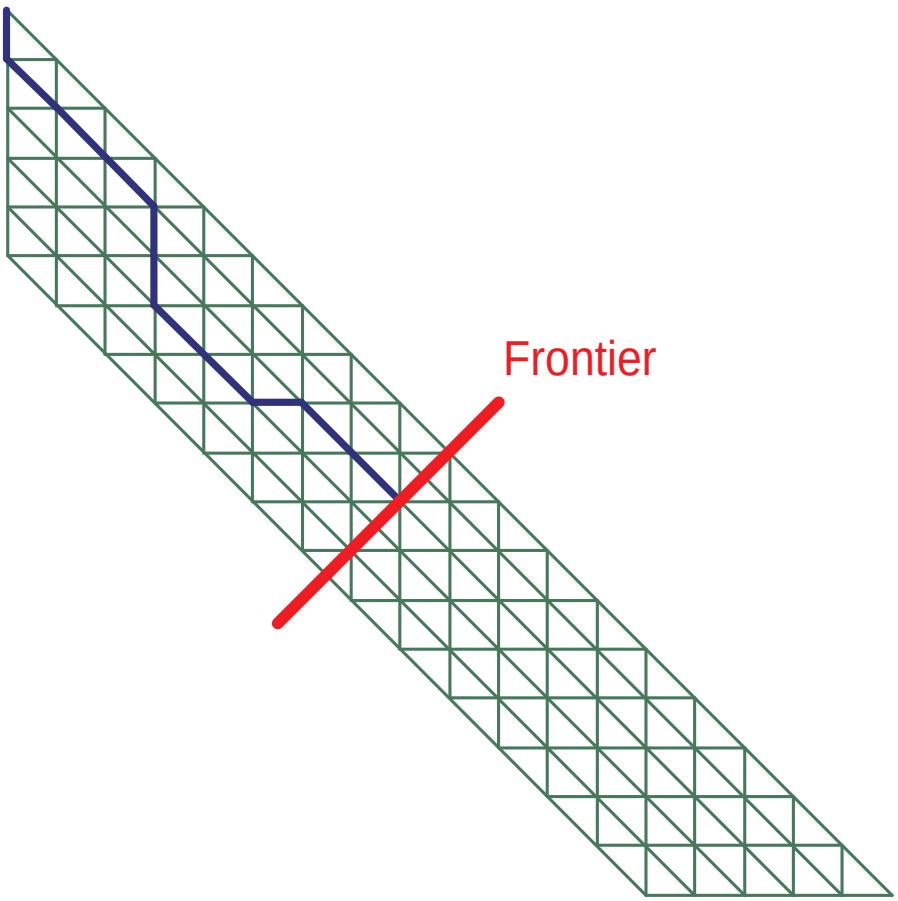

Figure 2: Dynamic programming moves a frontier (thick, red line) from top left to bottom right, and updates the costs of the best (blue) path up to each grid point on the frontier. Only a small number of such cost values must be maintained on the frontier.

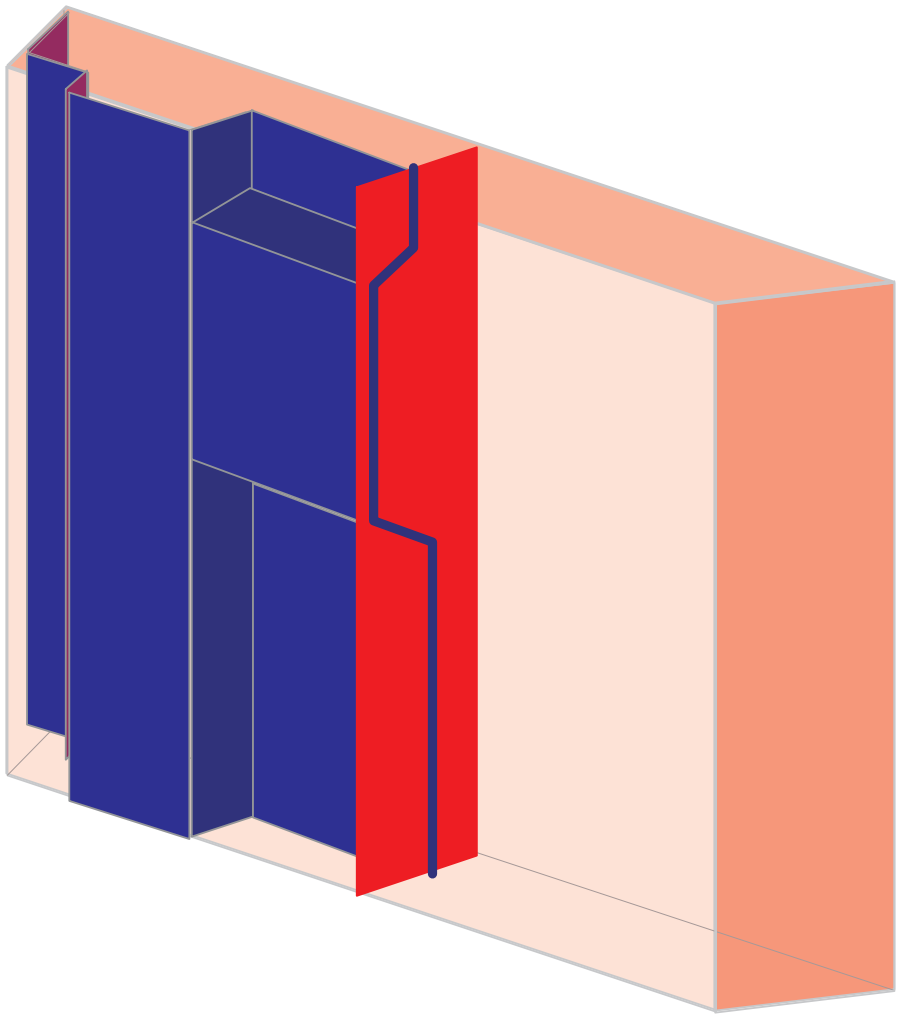

Figure 3: The blue path has now become a match *surface* that intersects the red frontier (another surface) along a curve.

since the two-dimensional grids for every scan line pair would have to be stacked on top of each other. Both curves involved in dynamic programming, the path and the frontier, would become surfaces: the path would be replaced by a family of paths, one per scan line pair, and this would form a match *surface*. Similarly, the frontier would have to be a surface as well in order to separate the top left from the bottom right of the three-dimensional grid. The new situation is illustrated in Figure 3.

This is why dynamic programming becomes infeasible for full image pairs: the intersection of a match surface and a frontier is no longer a point but a curve, and *the number of possible curves on any one frontier is too great*. We cannot store or update the cost of each possible match surface up to some frontier. The computational complexity makes dynamic programming infeasible.

3 The Max-Flow/Min-Cut Formulation

What is infeasible in Section 2 is the solution method, not the problem itself. In fact, to come up with a feasible solution, we just need to look at Figure 1 in a different way. The blue curve in that Figure has been described so far as a *path* that connects the top left corner of the match grid to its bottom right corner. Figure 4 shows the same curve interpreted differently, as a boundary or *cut* that separates the empty space in front of the surface, where disparities are higher, from that behind the surface, where disparities are lower.

Finding a good match then turns into the problem of separating front from back with a minimum cut, where the "size" of a cut is measured by the cost of the matches that lie along the cut. However, cutting the grid as drawn would entail removing nodes from it (the nodes on the solution cut). Minimum-cut problems are instead cast in terms of removing edges of a graph.

As explained in [6], since the graph is planar, all we need to do is to replace the original one with its dual, in which each region (triangle) in the original graph becomes a node, and edges are drawn between nodes of adjacent regions. Figure 5 shows this transformation. From here on, the details of graph construction are different from those in previous papers, and make the resulting graph simpler.

Since the original graph is undirected, so is its dual. The resulting graph is made to be directed by making all arrows point from high to low disparity. This will become useful a little later, when the ordering constraint is enforced.

There is one green, short edge for each match in the original (primal) grid, and the capacity of these edges equals the match cost that was charged to each node in the primal graph. The other, red edges have infinite capacity. Because of this, they cannot be cut without incurring an infinite penalty, so they will never be part of any minimum cut. Since the red edges are in series with the green ones, it is still possible to cut the graph by cutting only green edges.

To make the graph into a traditional flow network, we need to add a source and a sink node, and some plumbing to connect these to the rest of the graph. This is shown on a very small dual graph in Figure 6. From now on, we can forget the original (primal) graph.

The resulting graph can be cut with paths that are not valid, in that they do not satisfy the ordering constraint. To enforce the latter, we add one more set of edges, as shown in Figure 7. These also have infinite capacity, so they cannot be part of any minimum cut. The role of the ordering edges can be understood by referring to Figure 8, in which the small-scale details of a cut (blue curve) have been suppressed and only a few of the ordering edges have been drawn, for greater clarity. Note that whether the ordering edges traverse a cut from front to back of the cut or from back to front depends on the local orientation of the path. Solid arrows are edges that traverse the cut from front to back, and they occur in parts of the cut that are oriented in the direction forbidden by the ordering constraint, that is, generally from south-east to north-west when walking along the path starting from the top left of the diagram.

Dotted arrows, on the other hand, traverse parts of the cut that run in the valid direction, compatible with the ordering constraint, and they carry flow from back to front.

Because of this arrangement, any path that violates the ordering constraint must also cut ordering edges that carry flow forward (the solid arrows). Because these have infinite capacity, no minimum cost cut can have sections that run in the forbidden direction. On the other hand, cutting ordering edges where the cut has a valid direction is harmless, as these edges only carry flow from back to front, and do not contribute to the total flow from source to sink. Thus, these edges appropriately enforce the ordering constraint.

We are now ready for the last and crucial step, that is, the insertion of additional edges that connect adjacent pairs of scan lines for a completely global formulation of the stereo correspondence problem. The main constraint added by

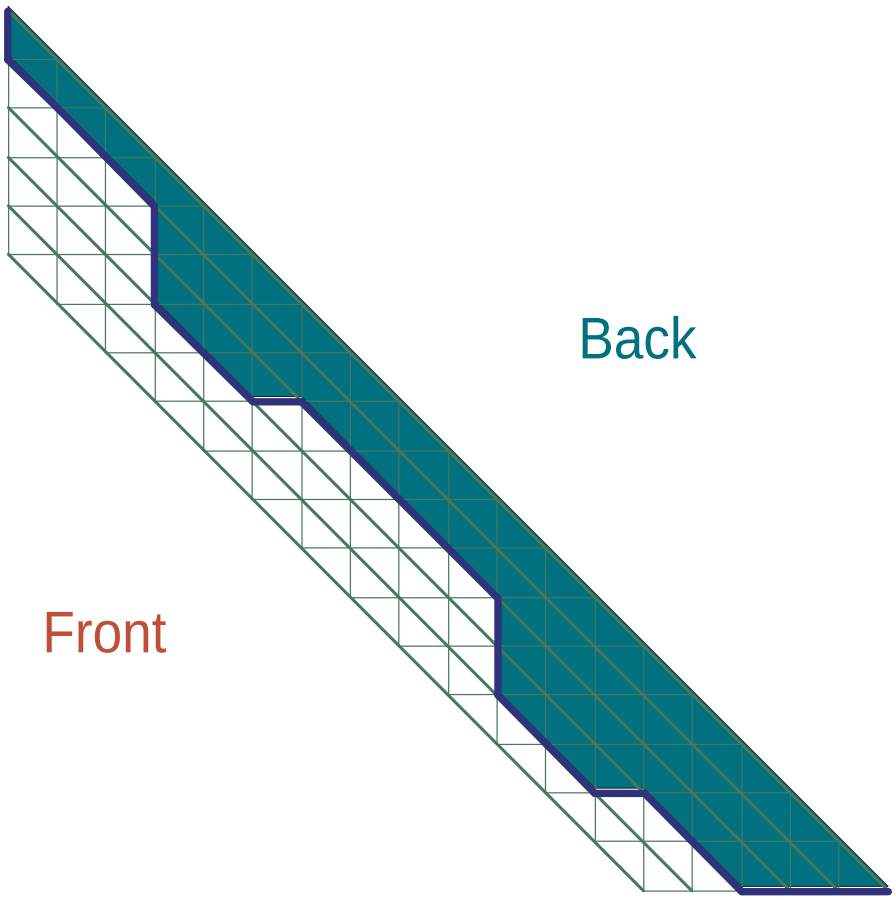

Figure 4: The path in Figure 1 can be viewed as a *cut* that separates what is in front of the surface (higher disparity) from what is behind.

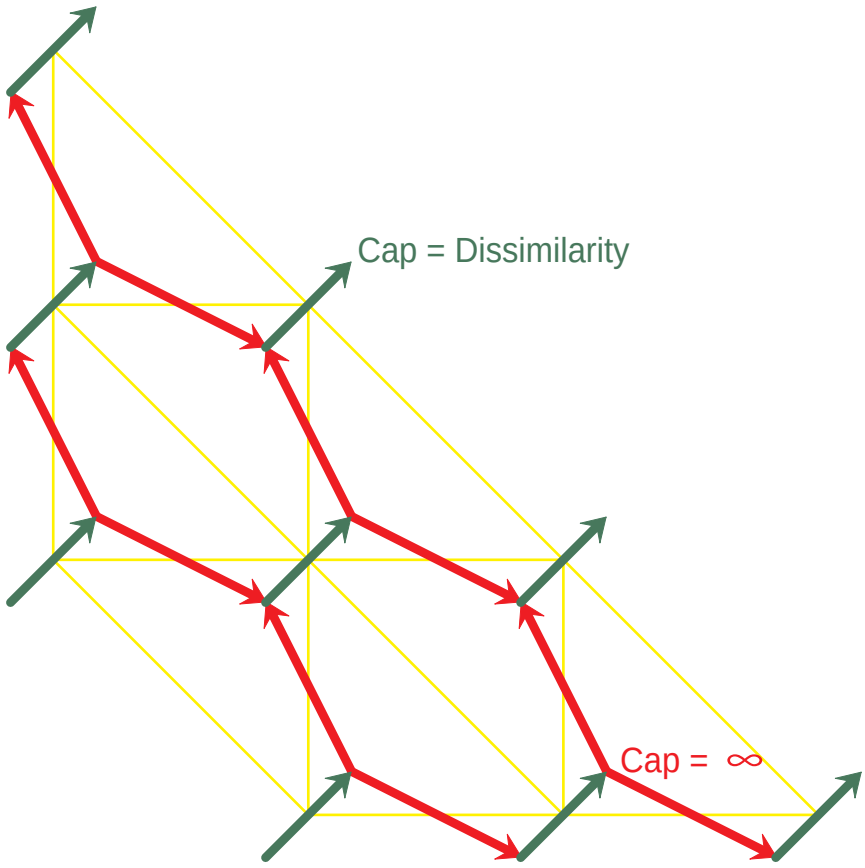

Figure 5: The thick, red and green arrows form the dual of the graph drawn in thin, yellow lines. We made the graph directed by having all arrows point from the front of the graph (higher disparities) to the back.

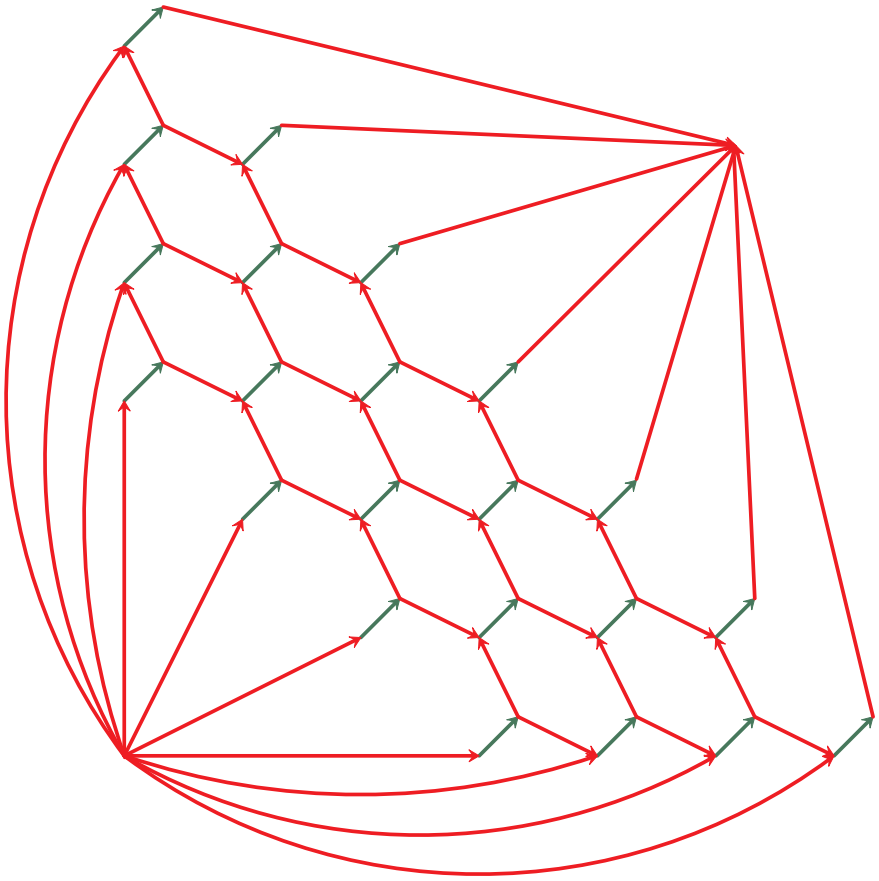

Figure 6: Adding a source and a sink, together with the necessary plumbing, makes a complete dual graph for a standard, single-source max-flow/min-cut problem.

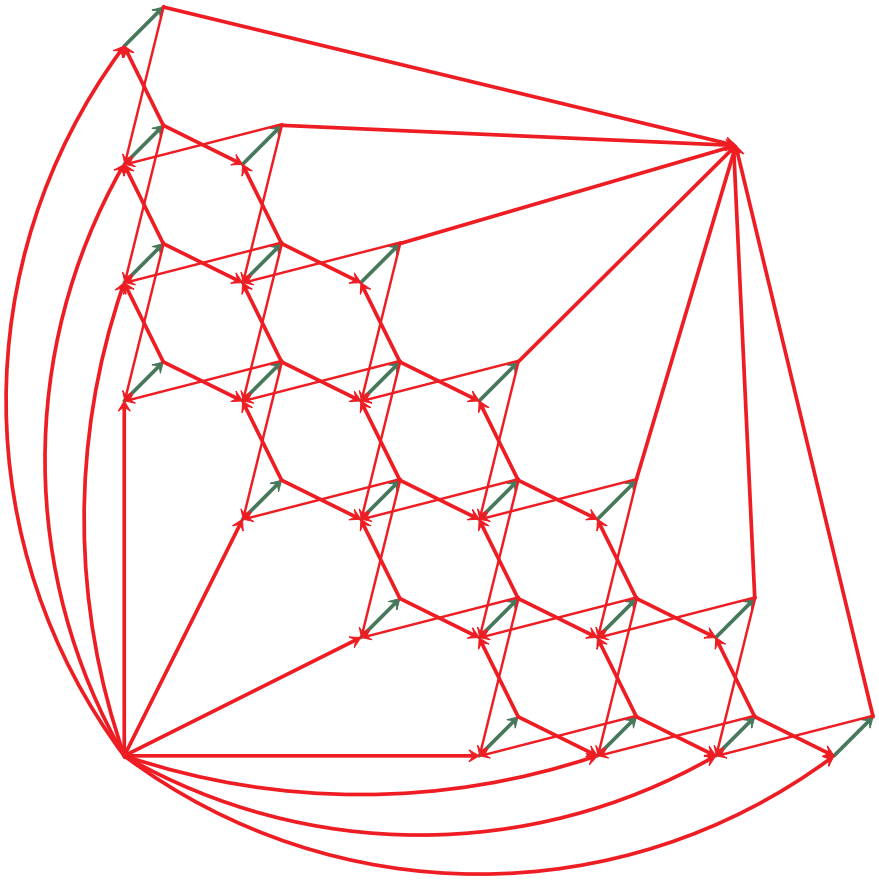

Figure 7: This is the complete graph for the two-dimensional matching problem. The extra, infinite-capacity edges enforce the ordering constraint, as explained in the text and in Figure 8.

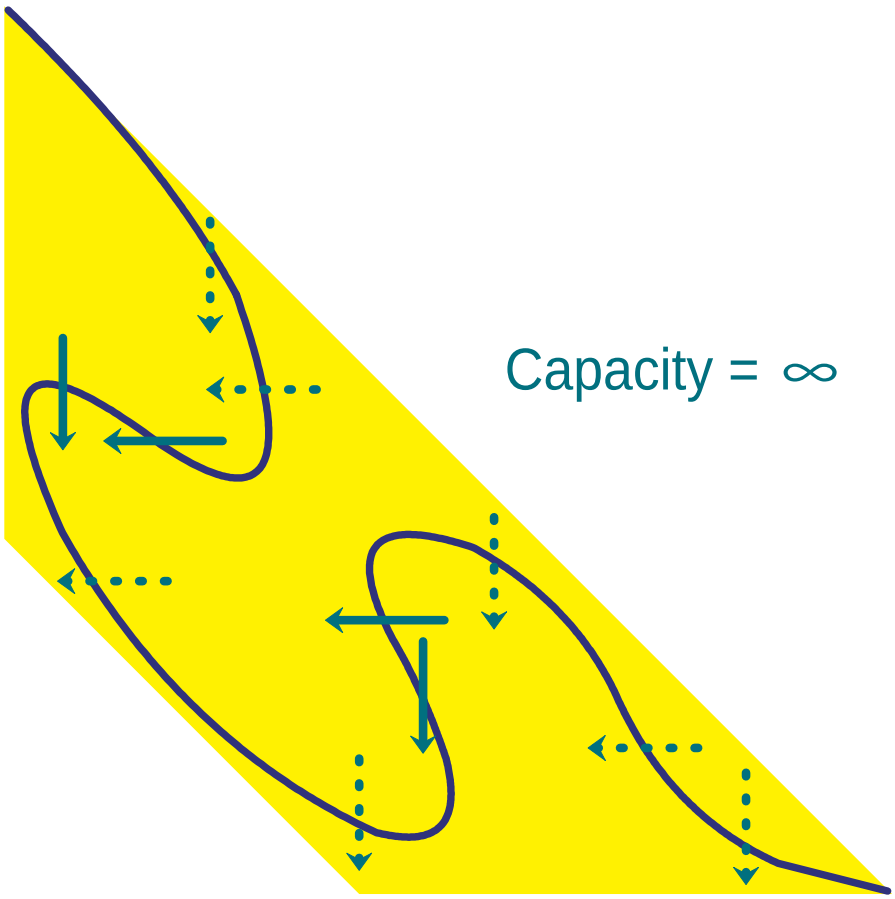

Figure 8: A conceptual view of the edges added in Figure 7. These enforce the ordering constraint by preventing cuts that would also have to cut infinite-capacity edges carrying flow from front to back of the cut. Solid arrows are examples of these. Dashed arrows are harmless because they carry flow from back to front, and therefore do not contribute to the cost of a cut.

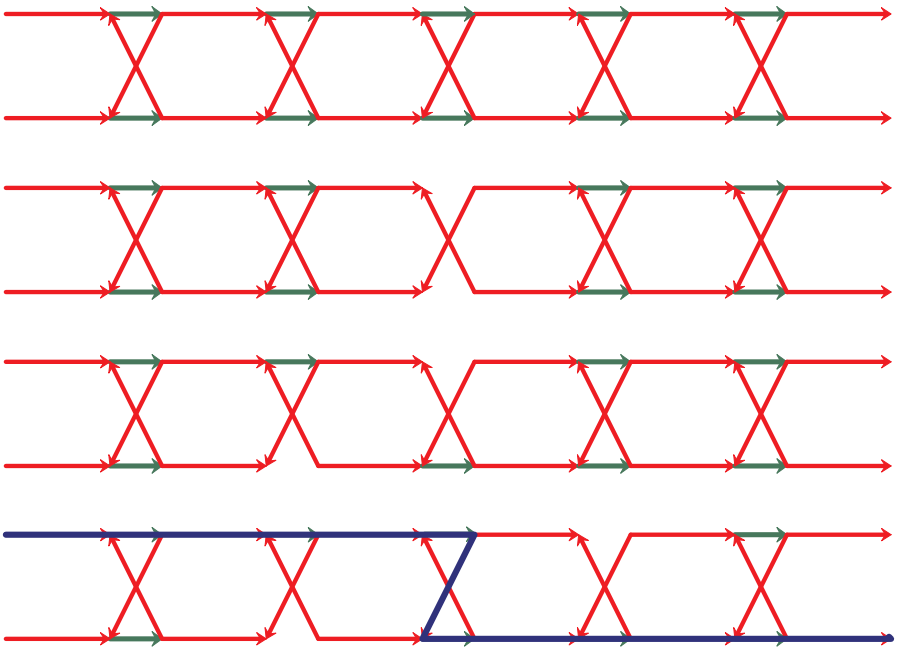

Figure 9: The top diagram shows the cross-edges added to enforce connectedness between rows for a slice of the complete, three-dimensional graph for the global stereo matching problem. To obtain a genuine cut, one must cut two vertically adjacent green edges (second diagram), or two diagonally adjacent green edges (third diagram), corresponding to disparity differences of zero or one pixel. The bottom diagram shows that removing green edges in violation of connectedness (two or more pixels of disparity difference) does not lead to a cut, as the blue flow can still make its way through the network.

a global solution is the *connectedness* of the resulting match surface. This requires vertically adjacent pixels to have similar disparities most of the time, a constraint of considerable heuristic power.

In previous work within the max flow/min cut framework, this constraint has invariably been implemented by the addition of tunable smoothness terms between vertically adjacent pixels. The resulting networks all have a large number of edges (17 for both [13] and [6]), and parameters. While parameters provide flexibility, they must be tuned, and it is hard to think of matching algorithms that can tune their parameters automatically as a function of the input images.

Instead, we propose a simpler version of connectedness, in which vertically adjacent pixels are merely required to have disparities that differ by at most one pixel. We handle discontinuities in depth by waiving this requirement whenever there are great changes in brightness between vertically adjacent pixels in either image, corresponding to the assumption that depth discontinuities come usually with brightness discontinuities.

Of course, this assumption is not always true. Also, constraining vertical disparity changes to at most one pixel might in some circumstances smooth over a very slanted surface. However, in both cases a (rare) violation of the heuristic merely costs a local blurring of the match surface with respect to truth. The great advantage of making this assumption is that the only parameter in the proposed method is a threshold implied by the edge detector that determines when a brightness discontinuity has occurred between two vertically adjacent pixels. Edge detection is well understood, including methods for selecting parameters [7].

Connectedness is enforced by a cross-connection, again with infinite-capacity edges, between vertically adjacent match edges (the green edges in Figure 7). Figure 9 shows these edges for a vertical slice through the three-dimensional graph obtained by stacking the two-dimensional graphs for all the scan line pairs. The slice is taken so that horizontally adjacent green edges correspond to disparities that differ by one pixel. The caption of this Figure explains why this arrangement enforces the proposed version of connectedness.

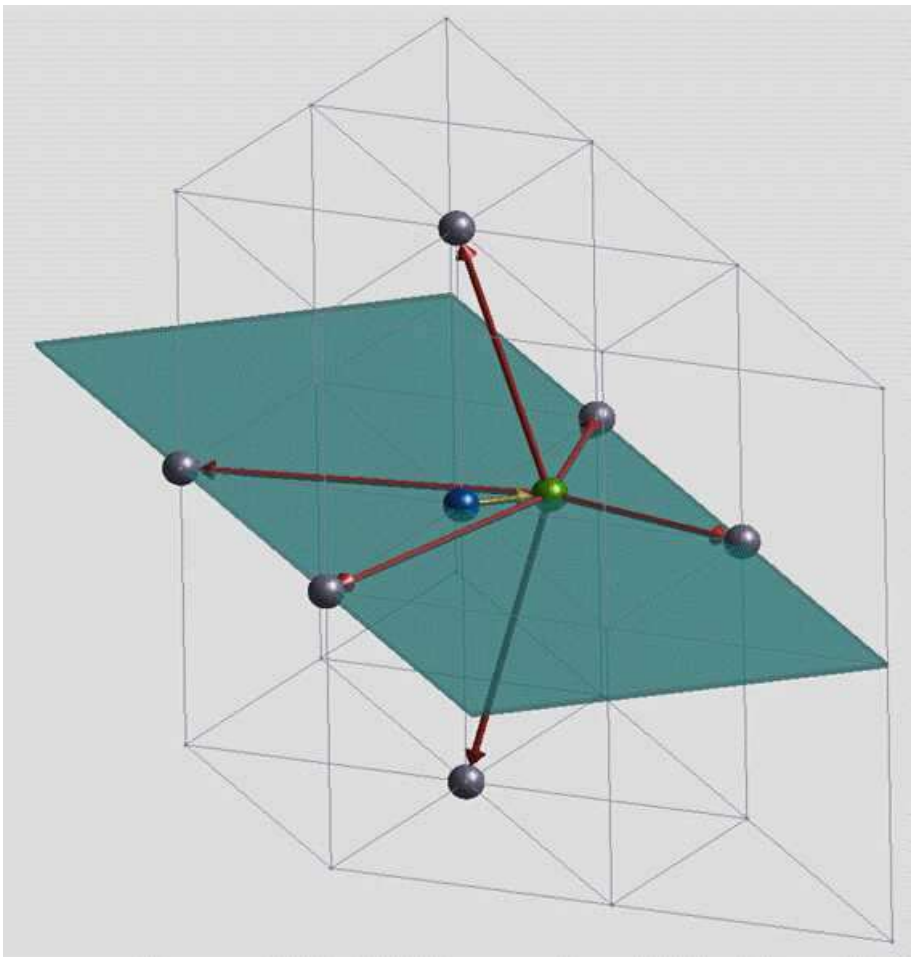

Figure 10: A node in the full network has seven edges. Of these, one (short, in gold) has a finite capacity that is an increasing function of the cost of the match corresponding to the node. The other six edges have infinite capacity.

In summary, the formulation of the stereo correspondence problem proposed here is essentially parameter-free (except for the well-understood parameters implied by the edge detector), and has seven edges per node, shown in Figure 10. Figure 11 shows a network for a very small matching problem.

The optimal disparity surface is found with an entirely off-the-shelf algorithm, Cherkassky and Goldberg's implementation [8] of the Goldberg-Tarjan max-flow algorithm [11]. This algorithm was designed to compute the maximum flow that can be pushed through a network. As well known [10], there exists a minimum cut of the network whose capacity is exactly the maximum flow, and finding the maximum flow also yields the minimum cut.

The asymptotic complexity of Goldberg and Tarjan's algorithm is $O(mn \log(n^2/m))$, where n is the number of vertices in the network and m is the number of edges. In our case, $m = 7n$ (see Figure 10), so m is $O(n)$, and the resulting bound is $O(n^2 \log n)$. To relate this to image size, let p be the number of pixels in the image and d the maximum disparity allowed. Then, $n = pd$. As image resolution increases, the maximum disparity must increase proportionally, so that d is $O(\sqrt{p})$. In summary, the asymptotic complexity is $O(p^3 \log p)$. In practice, the constants are small, and matching a 512 by 512 image takes of the order of one minute on a standard personal computer.

Both the goals of computational and conceptual simplicity have been achieved, albeit at the cost of a simplification of the underlying heuristics. The next Section shows that at least in one nontrivial case practical results are good. More experiments are left for future work.

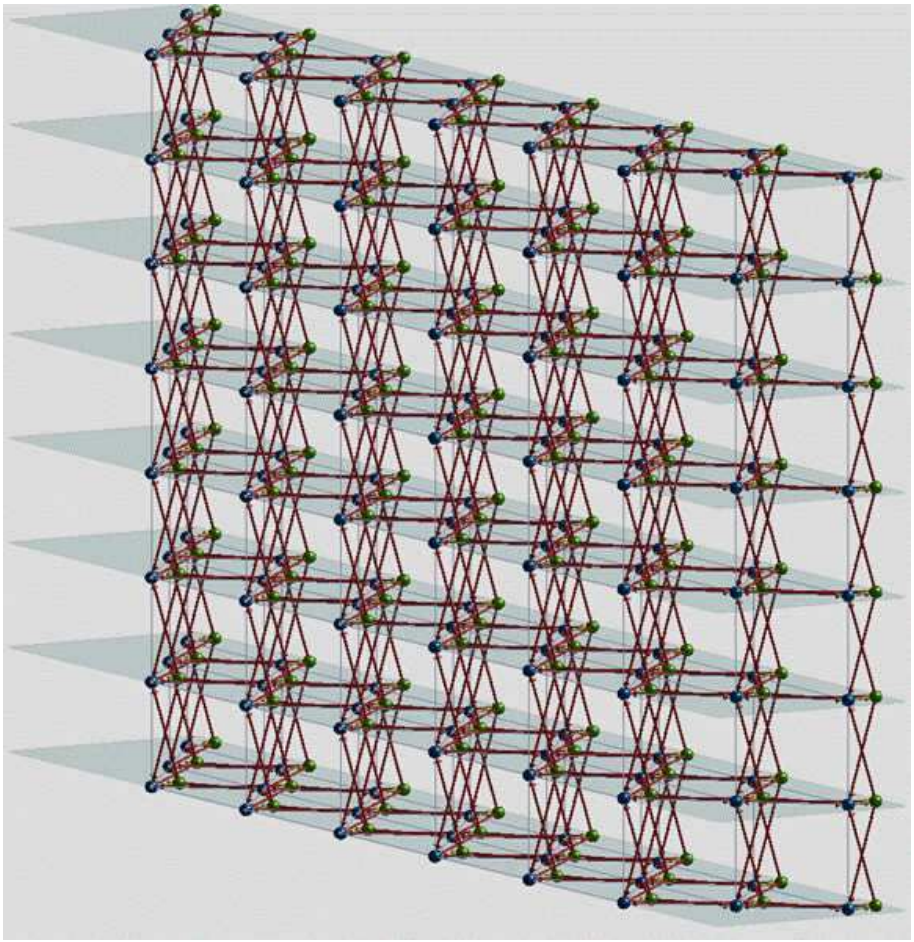

Figure 11: The three-dimensional network for a small matching problem. Source and sink edges have been omitted for clarity. Each of the shaded planes corresponds to a different pair of scan lines.

Figure 12: The Tsukuba stereo pair.

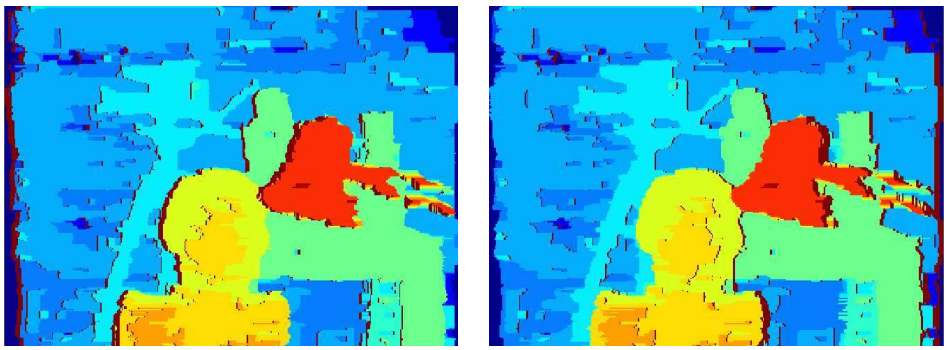

Figure 13: Color-coded view of the disparity surface mapped to the left and right image. Occlusions are in magenta (dark red).

4 An Experiment

The Middlebury web page provides among others the stereo pair shown in Figure 12, for which students at Tsukuba University have determined the correct matches by hand. Figure 13 shows the result computed by the maximum flow/minimum cut method described in the previous Section, and Figure 14 gives a different view of the left map of Figure 13. Images were processed at a resolution of 384 by 288 pixels, and matching required 40 seconds on a 1.4MHz Pentium personal computer. No parameters had to specified, except for the values for Canny's edge detector when determining intensity edges in the two images. These values are $\sigma = 1$ for the standard deviation of the detector's gradient operator, and $\theta_h = 0.14$ for the high threshold in the non-maximum suppression stage of the detector. The low threshold is then automatically set to $0.4\theta_h$. The two thresholds are referred to a range of image values normalized to $[0, 1]$. Please see [7] for the detailed meaning of these parameters. All these parameter values are the default values in the Matlab implementation of Canny's edge detector.

The fraction of correct disparity values is about 97%, similarly to [22]. This places this algorithm among the "relatively good" ones in the Middlebury competition, in which the absolute best reach just above 99% (if we take the hand-determined correspondences as the absolute truth). Note, however, that the proposed algorithm achieved these results without setting any parameters in the matcher. This out-of-the-box performance is rather unusual for stereo algorithms, which typically require extensive parameter tuning to yield good results.

5 Conclusions and Future Work

The main goal of this contribution has been to champion a global (*i. e.*, image-to-image) formulation of stereo matching that is equivalent to determining the minimum cut in a flow network. The resulting problem is essentially parameterless

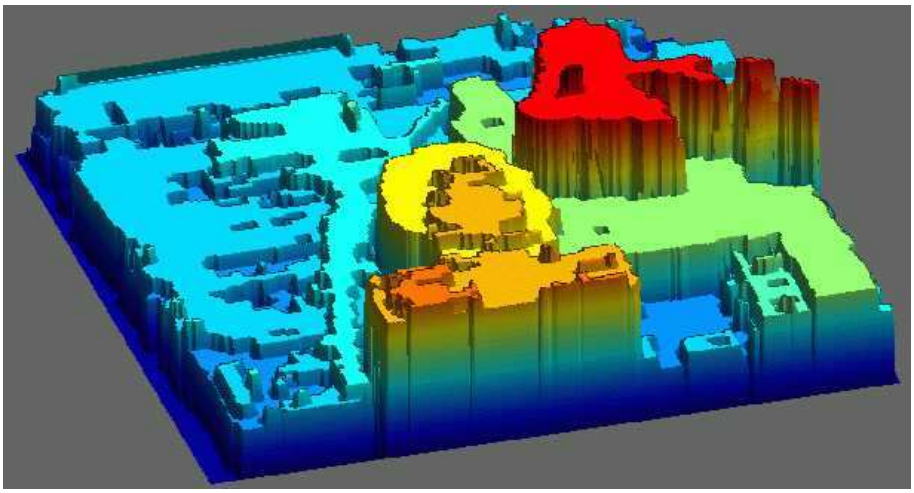

Figure 14: A three-dimensional display of the left view of the disparity surface.

and can be solved by efficient, off-the-shelf algorithms. This approach has been proposed a few times in the past eight years, and this paper provides a simpler instantiation of essentially the same idea.

The single experiment in the previous Section is not sufficient to establish the performance of the proposed algorithm. A systematic evaluation would run the algorithm on a series of image pairs of real scenes for which accurate ground truth is available. We are setting up an experimental facility with funding from the National Science Foundation of the USA for acquiring just this type of images.

All the same, the hope is that good performance achieved on a difficult image in the first run of the code will entice more researchers into studying and developing an approach to stereo that has the same features of simplicity of dynamic programming, but is applicable to entire images rather than just one pair of scan lines at a time.

In addition to working on alternative formulations within this framework, it may be useful to examine implementations that take advantage of the regular structure of the network in Figure 11 to achieve greater practical efficiency. This may be particularly useful for parallel implementations of the algorithm.

References

[1] R. D. Arnold and T. O. Binford. Geometric constraints in stereo vision. Technical report, Computer Science Department, Stanford University, 1980.

[2] Various Authors. The Digiclops stereo vision camera system. http://www.ptgrey.com.

[3] Peter N. Belhumeur and David Mumford. A Bayesian treatment of the stereo correspondence problem using half-occluded regions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 506–512, June 1992.

[4] S. Birchfield and C. Tomasi. Depth discontinuities by pixel-to-pixel stereo. In *Proceedings of the Sixth International Conference on Computer Vision (ICCV)*, pages 1073–1080, Bombay, India, January 1998.

[5] S. Birchfield and C. Tomasi. Multiway cut for stereo and motion with slanted surfaces. In *Seventh International Conference on Computer Vision (ICCV)*, pages 489–495, Kerkyra, Greece, September 1999.

[6] C. Buehler, S. J. Gortler, M. F. Cohen, and L. McMillan. Minimal surfaces for stereo. In *Computer Vision — ECCV 2002*, volume 3, pages 885–899, Berlin Heidelberg, 2002. Springer.

[7] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, November 1986.

[8] B. V. Cherkassky and A. V. Goldberg. On implementing the push-relabel method for the maximum flow problem. *Algorithmica*, 19:390–410, 1997.

[9] I. J. Cox, S. L. Hingorani, S. B. Rao, and B. M. Maggs. A maximum likelihood stereo algorithm. *Computer Vision and Image Understanding*, 63(3):542–567, May 1996.

[10] L. R. Ford, Jr. and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.

[11] A. V. Goldberg and R. E.Tarjan. A new approach to the maximum-flow problem. *Journal of the Association for Computing Machinery*, 35(4):921–940, October 1988.

[12] M. J. Hannah. *Computer matching of areas in stereo imagery*. PhD thesis, Computer Science Department, Stanford University, Stanford, CA, 1974.

[13] H. Ishikawa and D. Geiger. Occlusions, discontinuities, and epipolar lines in stereo. In *Proceedings of the European Conference on Computer Vision*, volume 1, pages 232–249, 1998.

[14] T. Kanade. Development of a video-rate stereo machine. In *Proceedings of the ARPA Image Understanding Workshop*, pages 549–557, Monterey, CA, November 1994.

[15] J. Kim, V. Kolmogorov, and R. Zabih. Visual correspondence using energy minimization and mutual information. In *International Conference on Computer Vision (ICCV)*, pages 1033–1040, October 2003.

[16] V. Kolmogorov and R. Zabih. Computing visual correspondence with occlusions using graph cuts. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 508–515, 2001.

[17] K. Konolige. Small vision systems: hardware and implementation. In *Eighth International Symposium on Robotics Research*, pages 111–116, 1997.

[18] M. H. Lin and C. Tomasi. Surfaces with occlusions from layered stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8):1073–1078, August 2004.

[19] D. Marr. A note on the computation of binocular disparity in a symbolic, low-level visual processor. Technical Report 327, MIT A.I. Lab., Cambridge, MA, 1974.

[20] Yuichi Ohta and Takeo Kanade. Stereo by intra- and inter-scanline search using dynamic programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-7(2):139–154, March 1985.

[21] S. Roy. Stereo without epipolar lines: a maximum-flow fomulation. *International Journal on Computer Vision*, 34(2/3):147–161, November 1999.

[22] S. Roy and I. Cox. A maximum-flow formulation of the N-camera stereo correspondence problem. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 492–499, 1998.

[23] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal on Computer Vision*, 47(1/2/3):7–42, April–June 2002. See also www.middlebury.edu/stereo.

[24] J. Sun, N. N. Zheng, and H. Y. Shum. Stereo matching using belief propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(7):787–800, July 2003.

[25] E. Trucco and A. Verri. *Introductory techniques for 3-D computer vision*. Prentice Hall, Upper Saddle River, NJ, 1998.

[26] G. Van Meerbergen, M. Vergauwen, M. Pollefeys, and L. Van Gool. A hierarchical symmetric stereo algorithm using dynamic programming. *International Journal on Computer Vision*, 47(1/2/3):275–285, 2002.

[27] J. Woodfill, G. Gordon, and R. Buck. Tyzx DeepSea high speed stereo vision system. In *Proceedings of the IEEE Computer Society Workshop on Real Time 3-D Sensors and Their Use, CVPR*, Washington, D.C., June 2004.

[28] O. Veksler Y. Boykov and R. Zabih. Fast approximate energy minimization via graph cuts. In *Proceedings of the International Conference on Computer Vision*, pages 377–384, September 1999.