A Maximum-Flow Formulation of the N-camera Stereo Correspondence Problem

Sébastien Roy* Ingemar J. Cox
NEC Research Institute
4 Independence Way
Princeton, NJ 08540, U.S.A.

Abstract

This paper describes a new algorithm for solving the N-camera stereo correspondence problem by transforming it into a maximum-flow problem. Once solved, the minimum-cut associated to the maximum-flow yields a disparity surface for the whole image at once. This global approach to stereo analysis provides a more accurate and coherent depth map than the traditional line-by-line stereo. Moreover, the optimality of the depth surface is guaranteed and can be shown to be a generalization of the dynamic programming approach that is widely used in standard stereo. Results show improved depth estimation as well as better handling of depth discontinuities. While the worst case running time is $O(n^2d^2log(nd))$, the observed average running time is $O(n^{1.2}\ d^{1.3})$ for an image size of n pixels and depth resolution d.

1 Introduction

It is well known that depth related displacements in stereo pairs always occur along lines associated to the camera motion, the epipolar lines. These lines reduce the stereo correspondence problem to one dimension and the ordering constraint allows dynamic programming to be applied [1–4]. However, it is clear that this reduction to 1-d is an oversimplification of the problem that is primarily necessary for computational efficiency. The solutions obtained on consecutive epipolar lines can vary significantly and create artifacts across epipolar lines, especially affecting object boundaries that are perpendicular to the epipolar lines (e.g. vertical boundary with horizontal epipolar lines).

In this paper, we address the full 2-d problem, replacing the traditional ordering constraint with the more general local coherence constraint. To perform the global 2-d optimization, we cast the stereo correspondence problem as a maximum-flow problem in a graph and show how the associated minimum-cut can be interpreted as a disparity surface. While the theoretical computational complexity is significantly higher for maximum-flow than dynamic programming, in practice, the average case performance is similar. We also show how this new paradigm can support both binocular and N-camera stereo configurations.

*Sébastien Roy is visiting from Université de Montréal, Département d'informatique et de recherche opérationnelle, C.P. 6128, Succ. Centre-Ville, Montréal, Québec, H3C 3J7

There have been several earlier attempts to relate the solutions of consecutive epipolar lines matched with dynamic programming. In [2], dynamic programming is used to first match epipolar lines and then iteratively improve the solutions obtained by using vertical edges as reference. In [3], a probabilistic approach is used to relate the individual matchings obtained by dynamic programming to improve the depth map quality. As a first approach, the current line matching uses the previous epipolar line solution to improve its own solution. However, this introduces a non-desirable vertical asymmetry. A second approach is to iteratively improve each epipolar line solutions with its neighboring lines solution. While this *local* approach is not globally optimal, it provides an efficient way to introduce smoothness constraint across epipolar lines. In [5], a Bayesian approach to the stereo correspondence problem is described. The resulting optimization problem can be solved efficiently by using dynamic programming along epipolar lines, resulting in the same problem as [2, 3] of relating the independent solutions. It proposes a heuristic method called *iterated stochastic dynamic programming* that uses previously computed adjacent epipolar line solutions to iteratively improve randomly selected solutions. This approach is not optimal and further more introduce a large amount of smoothness that tends to blur depth discontinuities.

Some multiple camera algorithms have been presented (see [4, 6–8]). In [6], a pair of camera is used as a *reference* or base pair. Other cameras provide extra information to enrich the matching cost function of the reference camera pair. The matching then proceed using dynamic programming as in [3]. In [7] and [8], a multiple-camera real-time stereo system is presented. They use a single *reference* camera to perform the matching. All the other cameras provide the information pertinent to each possible depth of points in the reference image. While each pixel is independently solved for depth, an implicit smoothness constraint is enforced by smoothing the images before processing them.

Section 2 describes a general N-camera stereo framework to be used with multiple images from arbitrary viewpoints. In Section 3, the stereo problem is extended from matching single epipolar lines to solving for a full disparity map. The generalization of the *ordering* constraint to *local coherence* constraint is also described there. In Section 4, the stereo match-

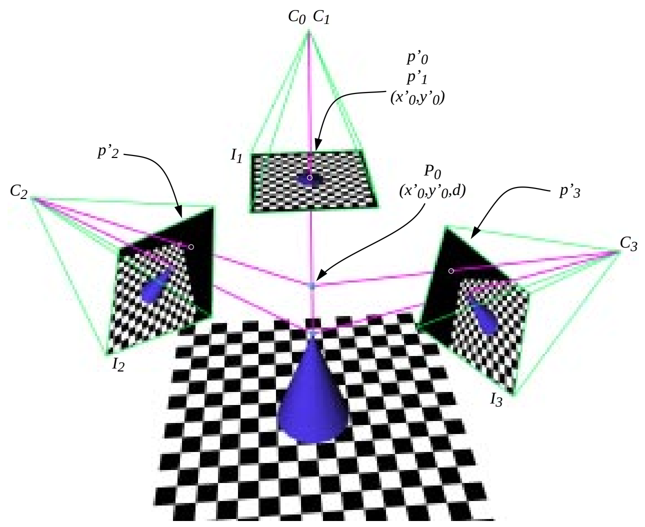

Figure 1: Multiple-camera stereo setup. For any disparity d of point $\mathbf{p}'_0$, you can back-project (x'_0, y'_0, d) in each inspection camera (C_1, C_2, C_3), obtaining the set of points p'_1, p'_2, p'_3.

ing problem is formulated as a maximum-flow problem. Details of the maximum-flow algorithm and performance issues are presented in Section 4.3. Experiments on both classic two-image and multiple-image stereo sequence are presented and discussed in Section 5.

2 Stereo Framework

In this section, we present a general framework to handle stereo in the context of multiple images taken under arbitrary camera geometries. It naturally extends the traditional two-image, single-baseline framework for stereo.

A set of n *inspection* cameras $C_1, \ldots, C_n$ provides n images $I_1, \ldots, I_n$ of a scene, as depicted in Figure 1 (with $n = 3$). A *base* camera C_0 provides the view for which we wish to compute the disparity map (or equivalently depth map) for every image point. The base camera does not have to provide an image; only the inspection cameras do. In the case of Figure 1, the base camera C_0 is identical to inspection camera C_1. A 3d point $\mathbf{P}_w$ expressed in the world coordinate system with homogeneous coordinates

$$\mathbf{P}_w = \begin{bmatrix} x_w & y_w & z_w & 1 \end{bmatrix}^T$$

can be transformed to the homogeneous point $\mathbf{P}_i$ in the coordinate system of camera i by the relation

$$\mathbf{P}_i = \mathbf{W}_i \, \mathbf{P}_w$$

where

$$\mathbf{W}_i = \begin{bmatrix} \mathbf{R}_i & \mathbf{T}_i \\ \mathbf{0}^T & 1 \end{bmatrix}$$

and $\mathbf{R}_i$ and $\mathbf{T}_i$ are, respectively, the rotation and translation matrices defining the position and orientation of camera i. Assuming the pinhole camera model, a point $\mathbf{P}_i$ is projected onto the image plane into the projective point $\mathbf{p}_i$ by the relation

$$\mathbf{p}_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = \mathbf{J} \, \mathbf{P}_i$$

where J is the projection matrix defined as

$$\mathbf{J} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

From a transformed and projected point $\mathbf{p}_i$, the corresponding image coordinates $\mathbf{p'}_i$ are obtained from the relation

$$\mathbf{p'}_i = H(\mathbf{p}_i)$$

where H is an homogenizing function

$$H(\begin{bmatrix} x \\ y \\ h \end{bmatrix}) = \begin{bmatrix} x/h \\ y/h \end{bmatrix}$$

During the process of stereo matching, each point $\mathbf{p}'_0$ of image I_0 is attributed a depth z or equivalently a disparity d (defined as $d = 1/z$) and can be expressed as

$$\mathbf{P}_0 = \begin{bmatrix} \mathbf{p}'_0 \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x'_0 \\ y'_0 \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x'_0 \\ y'_0 \\ 1 \\ d \end{bmatrix}$$

in the base coordinate system C_0. While these two formulations are equivalent, using the disparity formulation allows one to express naturally points that reach an infinite depth. Therefore, we use disparity d instead of depth z.

From this point $\mathbf{P}_0$, it is possible to project back to any camera image $\mathbf{p'}_i$ using the previously defined equations as

$$\begin{aligned} \mathbf{p'}_i &= H(\mathbf{p}_i) \\ &= H(\mathbf{J} \, \mathbf{P}_i) \\ &= H(\mathbf{J} \, \mathbf{W}_i \, \mathbf{P}_w) \\ &= H(\mathbf{J} \, \mathbf{W}_i \, \mathbf{W}_0^{-1} \, \mathbf{P}_0) \\ \begin{bmatrix} x'_i \\ y'_i \end{bmatrix} &= H(\mathbf{J} \, \mathbf{W}_i \, \mathbf{W}_0^{-1} \begin{bmatrix} x'_0 \\ y'_0 \\ 1 \\ d \end{bmatrix}) \end{aligned}$$

and therefore obtain pixel intensity information from inspection cameras in order to perform the matching.

During the stereo matching, each base image point $\mathbf{p}'_0 = [x'_0, y'_0]^T$ and its disparity value d generates a set of reprojected pixel values that form a pixel intensity vector $\mathbf{v}$ defined as

$$\mathbf{v}(\mathbf{p}'_0, d) = \{ I_i(H(\mathbf{J} \, \mathbf{W}_i \, \mathbf{W}_0^{-1} \begin{bmatrix} \mathbf{p}'_0 \\ 1 \\ d \end{bmatrix})) \}, \forall i \in [1, \ldots, n] \tag{1}$$

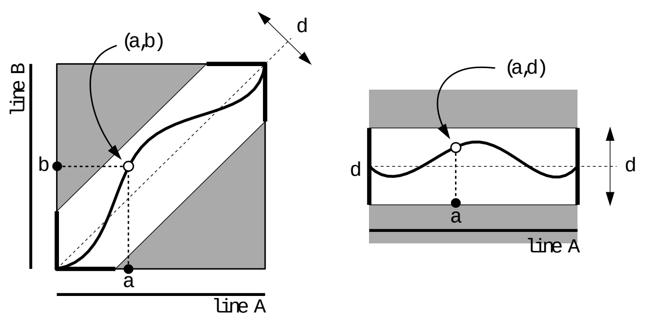

Figure 2: Epipolar Matching. Left, grid of all possible matches between line A and B. Right, equivalent formulation of the problem, where B does not appear directly.

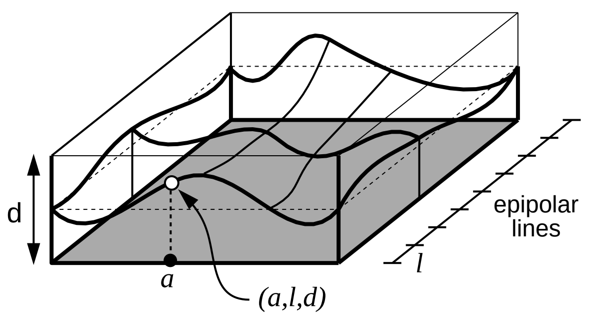

Figure 3: Matching whole images. All epipolar lines l are stacked together so that the whole image A is matched with disparity range d. A point has depth d and position a along epipolar line l.

This vector contains all the pixel intensity information from the inspection cameras for a particular match.

In order to perform the actual stereo matching, a *matching cost* function is required. Ideally, it is minimum for a likely match and large for an unlikely one. Deriving a meaningful matching cost is a non trivial task. Since this is not the primary purpose of this paper, we will use the simple form described next.

If we assume that surfaces are Lambertian (i.e. their intensity is independent of viewing direction) then the pixel intensity values of $\mathbf{v}(\mathbf{p}'_0, d)$ should be identical when $(\mathbf{p}'_0, d)$ is on the surface of an object and thus a valid match. Then, we can define the matching cost $cost(\mathbf{p}'_0, d)$ as the variance of the pixel intensity vector $\mathbf{v}(\mathbf{p}'_0, d)$ as

$$cost(\mathbf{p}'_0, d) = \frac{1}{n} \sum (\mathbf{v}(\mathbf{p}'_0, d) - \overline{\mathbf{v}}(\mathbf{p}'_0, d))^2 \qquad (2)$$

2.1 Epipolar geometry and Matching

It is a well known fact that for a given camera geometry, each image point is restricted to move along a single line called the epipolar line [4]. This reduces the matching process to a 1-D search along corresponding epipolar lines.

A very important additional constraint is the *ordering constraint*. It states that the order of points along corresponding epipolar lines is preserved. In fact, this corresponds to enforcing a *smoothness constraint* along epipolar lines (also noted in [4]).

In the traditional approach to stereo matching, a single epipolar line A is matched with its corresponding epipolar line B in the other image. The established matching between the two lines is a path in the grid of all possible matches (a, b), as shown on the left of Figure 2. The allowed starting and ending positions of the path are shown as thick black lines. By assuming that the ordering constraint is satisfied along epipolar lines, it is possible to solve this path problem very efficiently via dynamic programming [1–4].

In order to be able to use multiple cameras, the matching grid between lines A and B can be transformed into the equivalent formulation on the right of Figure 2, where only line A appears directly. For that case, each potential match has the form (a, d), where a is a position along line A and d is its associated disparity. The coordinates in image B corresponding to the match (a, d) are easy to compute from Eq. 1, while the cost function is directly obtained from Eq. 2. Given a match (a, d) or (a, b), it is straightforward to map it to any number of cameras with known geometries and therefore use extra information from multiple cameras. However, the representation using (a, d) is favored over one using (a, b) because we do need two base camera (A and B) as in [6] but only one (A).

3 Recovering a full disparity map

A natural extension to matching a single pair of epipolar lines at a time would be to extend it to the whole image at once, as depicted in Figure 3, by matching all pairs of epipolar lines simultaneously. Every minimum-cost path defining the matching of a single epipolar line are now assembled into a single minimum-cost surface. This surface contains all the disparity information of the base image. The goal of this construction is to take advantage of one very important property of disparity fields, *local coherence*, suggesting that disparities tend to be locally very similar, in any and all directions. As discussed previously, this property is exploited along epipolar lines by enforcing the ordering constraint. However, local coherence occurs in all directions and thus *across* epipolar lines. By putting all the epipolar lines together and solving globally for a disparity surface, it becomes possible to take full advantage of local coherence and improve the resulting depth map.

Note that each potential match (a, l, d) in Figure 3 is four-connected since it is part of a 2-D matching grid as presented in Figure 2. To take full advantage of local coherence, they have to be be six-connected to relate each individual epipolar line. Unfortunately, doing this makes dynamic programming unusable since there is no strict order for building the solution surface.

Many solutions for global disparity surface matching have been proposed [2, 5, 6]. Typically, these algorithm propose an iterative approach in which a solution is improved by using the previous matching obtained for neighboring epipolar lines. While this can sometimes work in practice, these solutions are not very efficient and not optimal.

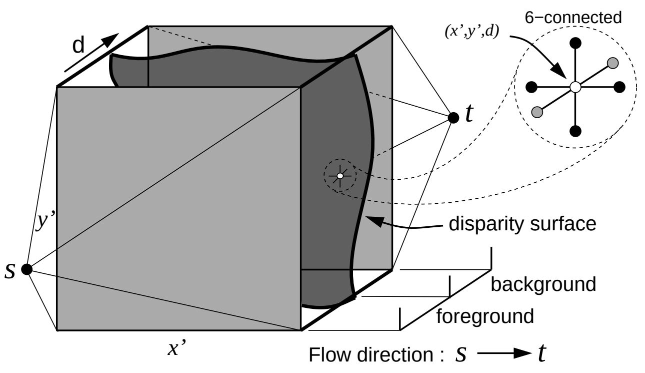

Figure 4: Image Matching as a Maximum Flow problem.

3.1 Avoiding direct use of epipolar geometry

An important distinction has to be made between the stereo matching problems depicted in Figures 3 and 4. In the first case, the epipolar lines are simply stacked up one after the other. While this might work for binocular stereo, it does not extend well to the case of multiple image stereo since the epipolar lines are specific to a single pair of cameras and arbitrary camera geometries will yield arbitrary set of epipolar lines.

To alleviate this problem, we discard the ordering constraint altogether, replacing it with the local coherence property mentioned in Section 3, which is similar but more general. In this new formulation, we can pick *any* set of lines in the image to be stacked together. The obvious choice is to take the set of horizontal lines since this is the natural image layout. This explains why we can refer to a point in Figure 4 by its image coordinates (x', y') instead of the epipolar line index l and position a in Figure 3.

The epipolar geometry is now only indirectly used in computing the matching cost for points with given disparity values (in Equation 1) but does not contribute as an explicit constraint to the matching process.

4 Stereo matching as a Maximum Flow problem

We propose to solve globally for the disparity surface by adding a source and a sink to the formulation of Figure 3, and treat it as a flow problem in a graph, as depicted in Figure 4. Consider the graph $G = (V, E)$ forming a 3-D mesh as in Figure 4. The vertex set V is defined as

$$V = V^* \cup \{s, t\}$$

where s is the source, t is the sink, and V^* is the 3d mesh

$$\begin{aligned} V^* &= \{(x', y', d) : x' \in [0 \ldots x'_{max}], \\ &\qquad y' \in [0 \ldots y'_{max}],\ d \in [0 \ldots d_{max}]\} \end{aligned}$$

where $(x'_{max} + 1, y'_{max} + 1)$ is the base image size and $d_{max} + 1$ is the depth resolution. Internally the mesh is six-connected and the source s connects to the front plane while the back plane is connected to the sink t. We have

$$E = \left\{ \begin{array}{lcl} (u, v) \in V^* \times V^* & : & \|u - v\| = 1 \\ (\ s\ ,\ (x', y', 0)\ ) & : & x' \in [0 \ldots x'_{max}] \\ (\ (x', y', d_{max})\ ,\ t\ ) & & y' \in [0 \ldots y'_{max}] \end{array} \right.$$

Being six-connected instead of four-connected, each vertex of the new problem is not only connected to its neighbors along the epipolar line (in depth), but also across adjacent epipolar lines (see Figure 4). Since dynamic programming is not possible in this situation, we can instead compute the maximum-flow between the source and sink. The set of edges that are saturated by the maximum-flow represent a *minimum-cut* of the graph. This cut separates the source and sink and effectively represents the disparity surface sought.

We define the edge capacities in the graph in a straightforward way. The matching cost is used directly as a capacity. Since a likely match has a low matching cost, the corresponding edge capacity will be low and that edge is likely to be saturated by the maximum-flow. Inversely, a high matching cost yields a high capacity edge which is unlikely to be saturated.

Since a vertex in the graph correspond to a potential match, we can use Equation 2 to derive its matching cost. The capacity of an edge is derived from the matching cost of the two vertices that it links. We arbitrarily define the edge capacity function $c(u, v)$ between vertices u and v from Equation 2 as

$$c(u, v) = \frac{\text{cost}(u) + \text{cost}(v)}{2} \tag{3}$$

where cost(u) is used for simplicity instead of cost(p'_0, d) since u is a match and defined by its associated point p'_0 and disparity d. In fact, since an edge links to vertices that each represent a specific 3-D match, it corresponds itself to a line segment in each inspection image. The obvious improvement to the edge capacity function is to derive it directly from these line segments. The average of two vertices matching cost is just a heuristic that works quite well in practice.

4.1 Expressing smoothness through edge capacity

In order to control the level of smoothness of the disparity map, it is important to differentiate between two kind of edges. As depicted in Figure 5, an edge oriented along the disparity axis is called a *disparity* edge while all other edge orientation are called *occlusion* edge. It will be shown later that the capacity of occlusion edges directly controls the level of smoothness. Edges adjacent to the source or sink are not classified and have infinite capacities. We have

$$c(u, v) = \left\{ \begin{array}{ll} 0 & \text{if } (u, v) \notin E \\ \infty & \text{if } u = s \text{ or } v = t \\ c_{disp}(u, v) & \text{if } (u - v) = (0, 0, \Delta_d) \\ c_{occ}(u, v) & \text{if } (u - v) = (\Delta_x, \Delta_y, 0) \end{array} \right.$$

where $c_{disp}(u, v)$ is the capacity of a disparity edge (oriented along the d axis) while $c_{occ}(u, v)$ is an occlusion edge (oriented along the x or y axis). In Figure 5,

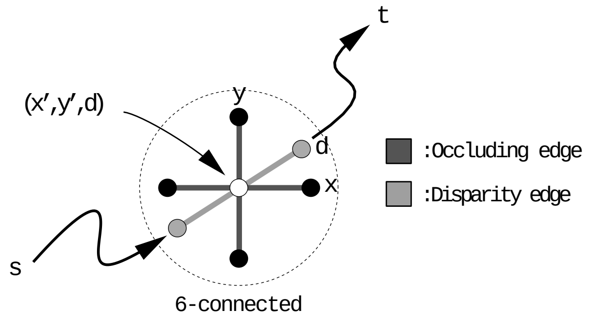

Figure 5: Expressing smoothness through edge capacity.

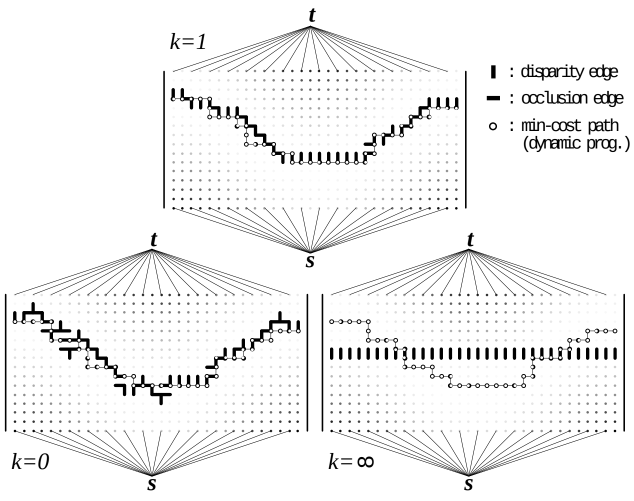

Figure 6: Example cuts for different smoothness values. $k = 0$, maximal discontinuity. $k = 1$, intermediate smoothness. $k = \infty$, infinite smoothness.

the darker edges (connecting the black vertices) are occlusion edge while lighter edges disparity edges. We define these costs from Equation 3 as

$$\begin{array}{rcl} c_{disp}(u,v) & = & \frac{\text{cost}(u)+\text{cost}(v)}{2} \\ c_{occ}(u,v) & = & k\, c_{disp}(u,v) \;\; (0 \le k \le \infty) \end{array}$$

where k is a smoothness parameter. A higher occlusion cost (i.e. larger k) increases the smoothness of recovered surfaces while, inversely, a lower occlusion cost facilitate depth discontinuities.

To illustrate the effect of the smoothness parameter k, we created an example 2-D problems with a simple cost function, as shown in Figure 6. For reference purposes, a minimum-cost path linking the left and right sides of the graph was computed using standard dynamic programming and is displayed as a chain of white dots. The maximum-flow was computed in this graph for smoothness values 0, 1, and ∞ and the corresponding minimum-cut are displayed as sets of thick black edges. These extreme values of the smoothness parameter k have intuitive consequences. When $k = \infty$, the resulting disparity surface is flat (maximally smooth) and features a single disparity value for the whole image. Setting $k = 0$, each column of the graph is independently given a disparity, therefore achieving maximal discontinuity in the disparity surface. For $k = 1$, at the top of Figure 6, a balance is reached and the minimum-cut corresponds very well to the minimum-cost path computed by dynamic programming.

4.2 From a cut to a disparity surface

It is well known that once the maximum flow is found, a minimum-cut C separates the source and sink in such a way that the sum of edge capacities of C is minimized. This cut is therefore the optimal way to separate the source and the sink for the particular cost function. Since the source is connected to the *closest* points while the sink is connected to the *deepest* points, the cut effectively separates the view volume into a foreground and background and yields the depth map of the scene. The minimum cut is also guaranteed to provides a depth estimate for each image point, as demonstrated by Property 1.

Property 1 (cut as a depth map)
Consider a cut C associated with some flow in the graph $G = (V, E)$. For all (x, y), there exist at least one d such that the edge $(x, y, d) - (x, y, d + 1)$ is part of C.

Proof. For any (x, y), there is a path $s \rightsquigarrow t$ in G of the form

$$s \to (x, y, 0) \to (x, y, 1) \to \ldots \to (x, y, d_{max}) \to t$$

therefore containing the set of edges

$$\left\{ \begin{array}{ll} s \to (x,y,0) & \\ (x,y,d) \to (x,y,d+1) & d \in [0, d_{max} - 1] \\ (x,y,d_{max}) \to t & \end{array} \right\}$$

Any cut of G must break this path and thus contain at least one edge of the form $(x, y, d) - (x, y, d + 1)$ since the edges $s \to (x, y, 0)$ and $(x, y, d_{max}) \to t$ have infinite capacities. □

According to property 1, a depth map can be constructed from the minimum-cut C of graph G as follow. For each point (x, y), the disparity is the largest d such that the edge $(x, y, d) - (x, y, d + 1)$ belongs to C. This results in the desired global disparity surface.

4.3 Solving the Maximum Flow problem

There is an abundant literature on algorithms to solve the maximum-flow problem [9, 10]. For this paper, we implemented a well known algorithm, *preflow-push lift-to-front* (see [9]). Currently, the best maximum-flow algorithm is presented in [10] and is particularly well suited for sparse graphs like the ones built for stereo matching.

The number of vertices v in the graph is equal to the number of image pixels multiplied by the depth resolution. For an image of size n pixels, i.e. of dimension approximately $\sqrt{n} \times \sqrt{n}$, and a depth resolution of d

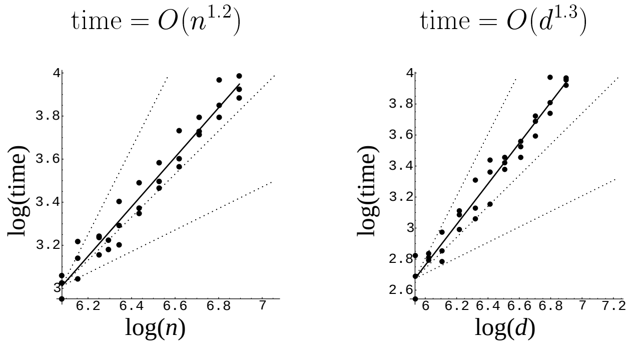

Figure 7: A) Performance as a function of image size n in pixels, for fixed depth resolution. B) Performance as a function of depth resolution d for a fixed size n. Three dotted lines show performance levels of $O(\sqrt{n})$, $O(n)$, and $O(n^2)$.

steps, we have $v = nd$. Since the graph is a three-dimensional mesh where each vertex is six-connected, the number of edge e is $e = O(V) = nd$.

This implies that the preflow-push algorithm used, with a running time

$$O(ve \log(v^2/e))$$

yields a running time of

$$O(n^2 d^2 \log(nd))$$

The new best bound [10] runs in

$$O(e^{\frac{3}{2}} \log(v^2/e) \log(U))$$

where U is the largest edge capacity, yields a running time of

$$O(n^{1.5} d^{1.5} \log(nd) \log(U))$$

The dynamic programming approach on separate epipolar lines [3] requires a total running time of $\Theta(nd)$, which might seem much better than the maximum-flow algorithm. However, the topology of the graph, the position of the source and sink, and the structure of edge capacities all tend to make the problem easier to solve, making the average running time much better than the worst case analysis. Figure 7 shows the typical performance as a function of total image size n (in pixels) and depth resolution d. The average running time is $O(n^{1.2}\ d^{1.3})$, which is almost linear with respect to image size n (in pixels) and compares favorably with the dynamic programming approach. The typical running time for 256×256 images is anywhere between 1 to 30 minutes, on a 160Mhz pentium machine, depending on the depth resolution used. While this is considerably slower than [3], the algorithm was not optimized for speed. Performance improvement are expected in the future.

5 Experiments and results

In this sections, results of binocular and N-camera stereoscopic matching from maximum-flow are presented and compared with two other algorithms.

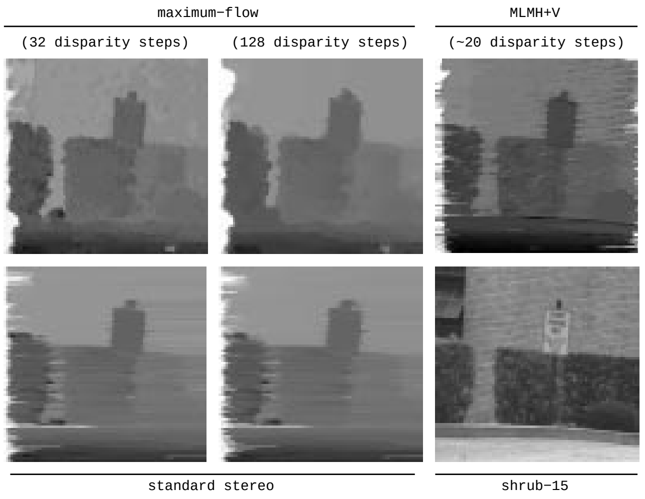

Figure 8: Disparity maps for the Shrub a two precision level (32 and 128 disparity steps). On top, the maximum-flow and MLMH+V results. At bottom, the original image shrub-15 and results for standard stereo.

First, the algorithm referred to as standard stereo uses line-by-line dynamic programming on N-camera with variable depth resolutions. It differs from the maximum-flow algorithm only in the way it solve the disparity surface. They are otherwise identical and their results use the same disparity scale and are not equalized.

Second, the algorithm referred to as MLMH+V is the efficient dynamic programming implementation from [3] (for the binocular version) and from [6] (for the N-camera version). It performs an iterative optimization of its disparity solution to enforce smoothness across disparity lines. It should be noted that the results from this algorithm use a different disparity scale (gray levels) than maximum-flow or standard stereo and are equalized to improve their contrast.

Shrub

Figure 8 shows one image of a pair of the Shrub image sequence (courtesy of T. Kanade and T. Nakahara of CMU), along with some matching results. These results show how maximum-flow tends to extract sharp and precise depth discontinuities, while standard stereo and MLMH+V produce many artifacts along vertical depth discontinuities. Two level of depth resolutions are shown (32 and 128 steps) with different level of smoothness. It is notable that even at high smoothness levels, maximum-flow does not produce spurious horizontal links across the gap between the two larger shrubs. The results of multiple-camera analysis is shown in Figure 9. All the images of this sequence share a common horizontal baseline. Even if the algorithms use different number of images (4 and 7), the total spanned camera displacement is the same and therefore provide about the same depth discrimination. Some image normalization is performed for MLMH+V prior to matching. None was used for the other two algorithms.

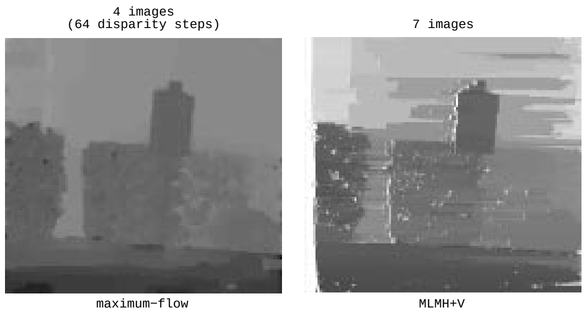

Figure 9: Disparity maps for the 4 and 7 images Shrub sequence. Both sequences span the same total horizontal displacement and should yield similar results. White points on the right denote detected occlusions.

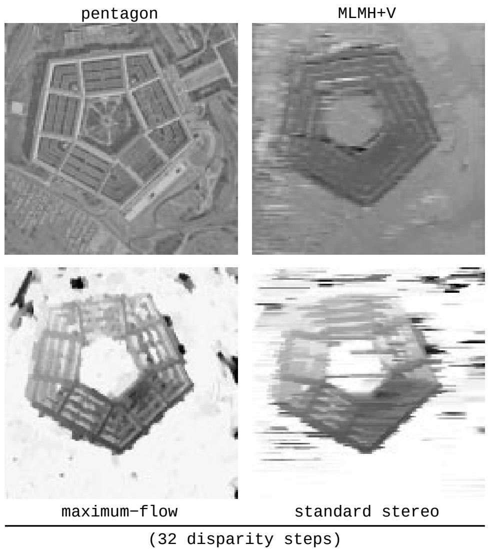

Figure 10: Disparity maps for the Pentagon stereo pair.

Pentagon

The left image of the stereo pair Pentagon is shown in Figure 10, along with the matching results. This stereo pair presents some challenge since the true camera motion is not exactly horizontal and contain some rotation, creating image motions that violates the epipolar constraint. Fortunately, algorithms like MLMH+V resist better to these misalignment since they allow negative disparities as well as positive. This explains how the highway structures at the top left are well recovered for MLMH+V while the other algorithms produced some noticeable spurious mismatch. A predicted, maximum-flow does produce a more symmetric result, with less spurious horizontal streaks.

Park meter

The image sequence Park meter shown in Figure 11 was analyzed for different number of images. Here a number of vertical objects put in evidence the difficulties that standard stereo and MLMH+V have to relate horizontal epipolar lines solutions together. No horizontal streaks are present in maximum-flow. Using 4 images (horizontally displaced along a single baseline), the results at the bottom of Figure 11 improve sensibly from those at the top. No results were available for MLMH+V.

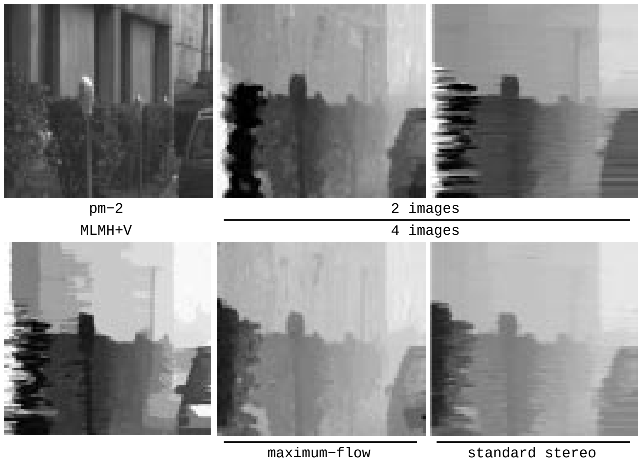

Figure 11: Disparity maps for the Park meter sequence. Results are shown for 2 and 4 image sequence. The MLMH+V result is shown for 2 images.

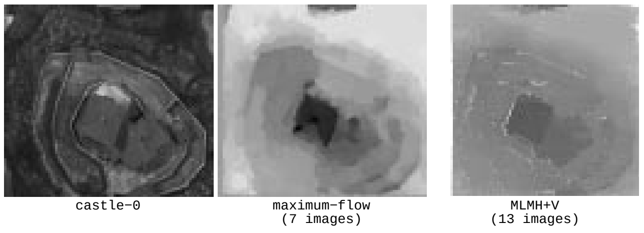

Figure 12: Disparity maps for the Roof sequence. Results are shown for 7 and 13 images, respectively. White points on the right denote detected occlusions.

Roof

The image sequence "Roof" (courtesy of T. Kanade and E. Kawamura of CMU) is shown on the left of Figure 12. It contains 13 images featuring either horizontal or vertical translations. The results for maximum-flow and MLMH+V are presented at the right. The disparity map obtained by maximum-flow is very detailed. In particular, the structure of the roof is well reconstructed. Note that only 7 horizontally separated images were used by maximum-flow because the exact amount of vertical displacement of the remaining 6 images was not available.

Castle

The sequence Castle from CMU is shown on the left of Figure 13 and contains 11 images with various combinations of horizontal, vertical and forward camera

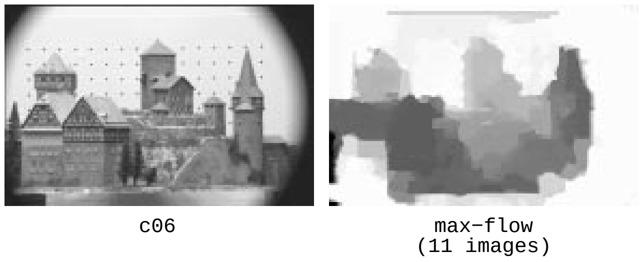

Figure 13: The Castle image stereo sequence. On the left, one of the 11 images. On the right, the resulting maximum-flow disparity map.

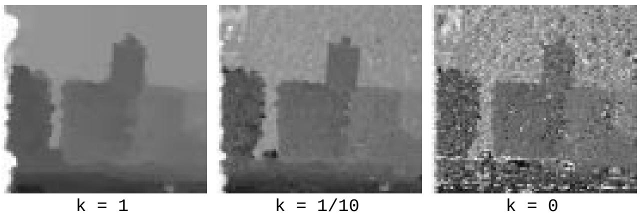

Figure 14: Disparity maps for the Shrub sequence for 3 smoothness levels. On the left, $k = 1$ enforce high smoothness. In the middle, $k = 1/10$ is medium smoothness. On the right, $k = 0$ enforce no smoothness.

motion. The 11 images were used to create the disparity map shown on the right. A high level of detail and very few spurious matches are present. Notice that the white background is recovered correctly regardless of its lack of texture.

It is important to note that this sequence represent a challenge since the actual disparity range, i.e. the difference in disparity between the closest and the farthest object, is only 2.7 pixels. Performed at a depth resolution of 96 steps, this implies that the disparity precision achieved is 0.03 pixels.

5.1 Level of Smoothness

In this section, we wish to illustrate how the level of smoothness, represented by the parameter k of Section 4.1, can affect the quality of the disparity map recovered. Figure 14 illustrates this for three level of smoothness, namely $k = 1$, $k = 1/10$ and $k = 0$. For $k = 0$, the capacity of occlusion edges is zero and therefore each pixel is given a disparity independently of its neighbors. It is essentially equivalent to finding the best disparity by correlation over a single pixel window (on the right of Figure 14).

As expected, lowering the occlusion capacities favors depth discontinuities and therefore creates sharper object edges, at the expense of surface smoothness.

It is observed that large depth discontinuities tend to stay sharp as the level of smoothness increases. This is probably due to the fact that the smoothness is expressed in all direction instead of only along epipolar line. This result differs strongly from most other methods where a high level of smoothness induces blurred or missing depth discontinuities.

6 Conclusion

We presented a new algorithm for establishing N-camera stereo correspondence, based on a reformulation of the stereo matching problem to finding the maximum-flow in a graph. Representing a generalization of dynamic programming along epipolar lines to the global matching space, it is able to solve optimally for the full disparity surface in a single step, therefore avoiding the usual disparity inconsistencies across neighboring epipolar lines. The *ordering* constraint, required for dynamic programming, is replaced with a more general *local coherence* property that applies in all directions instead of along epipolar lines. The new stereo problem formulation naturally supports multiple arbitrary cameras and can estimate depth for an arbitrary virtual camera. For any desired level of smoothness, depth discontinuities are well preserved since smoothness is applied in all directions instead of only along epipolar lines.

As for future research, there are many avenues open to improve the maximum-flow formulation proposed in this paper. In particular, a multi-resolution approach as well as local smoothness variations could be directly embedded in the graph, improving performance and depth map quality. The edge capacity computation can also be improved (as discussed at then end of Section 4) by directly comparing image line segments instead of single pixels.

References

[1] H. H. Baker. *Depth from Edge and Intensity Based Stereo.* PhD thesis, University of Illinois at Urbana-Champaign, 1981.

[2] Y. Ohta and T. Kanade. Stereo by intra- and inter-scanline using dynamic programming. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 7(2):139–154, 1985.

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

[4] O. Faugeras. *Three-dimentional computer vision.* MIT Press, Cambridge, 1993.

[5] P. N. Belhumeur. A Bayesian approach to binocular stereopsis. *Int. J. Computer Vision*, 19(3):237–260, 1996.

[6] I. J. Cox. A maximum likelihood N-camera stereo algorithm. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pages 733–739, 1994.

[7] S. B. Kang, J. A. Webb, C. L. Zitnick, and T. Kanade. An active multibaseline stereo system with real-time image acquisition. Technical Report CMU-CS-94-167, School of Computer Science, Carnegie Mellon University, 1994.

[8] T. Kanade, A. Yoshida, K. Oda, H. Kano, and M. Tanaka. A stereo machine for video-rate dense depth mapping and its new applications. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, San Francisco, 1996.

[9] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* McGraw-Hill, New York, 1990.

[10] A. V. Goldberg and S. B. Rao. Length functions for flow computations. Technical Report 97-055, NEC Research Institute, Princeton NJ, 1997.