A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms

Daniel Scharstein
Dept. of Math and Computer Science
Middlebury College
Middlebury, VT 05753
schar@middlebury.edu

Richard Szeliski
Microsoft Research
Microsoft Corporation
Redmond, WA 98052
szeliski@microsoft.com

Abstract

Stereo matching is one of the most active research areas in computer vision. While a large number of algorithms for stereo correspondence have been developed, relatively little work has been done on characterizing their performance. In this paper, we present a taxonomy of dense, two-frame stereo methods. Our taxonomy is designed to assess the different components and design decisions made in individual stereo algorithms. Using this taxonomy, we compare existing stereo methods and present experiments evaluating the performance of many different variants. In order to establish a common software platform and a collection of data sets for easy evaluation, we have designed a stand-alone, flexible C++ implementation that enables the evaluation of individual components and that can easily be extended to include new algorithms. We have also produced several new multi-frame stereo data sets with ground truth and are making both the code and data sets available on the Web. Finally, we include a comparative evaluation of a large set of today's best-performing stereo algorithms.

1. Introduction

Stereo correspondence has traditionally been, and continues to be, one of the most heavily investigated topics in computer vision. However, it is sometimes hard to gauge progress in the field, as most researchers only report qualitative results on the performance of their algorithms. Furthermore, a survey of stereo methods is long overdue, with the last exhaustive surveys dating back about a decade [7, 37, 25]. This paper provides an update on the state of the art in the field, with particular emphasis on stereo methods that (1) operate on two frames under known camera geometry, and (2) produce a *dense* disparity map, i.e., a disparity estimate at each pixel.

Our goals are two-fold:

1. To provide a **taxonomy** of existing stereo algorithms that allows the dissection and comparison of individual algorithm components design decisions;
2. To provide a **test bed** for the quantitative evaluation of stereo algorithms. Towards this end, we are placing sample implementations of correspondence algorithms along with test data and results on the Web at `www.middlebury.edu/stereo`.

We emphasize calibrated two-frame methods in order to focus our analysis on the essential components of stereo correspondence. However, it would be relatively straightforward to generalize our approach to include many multi-frame methods, in particular multiple-baseline stereo [85] and its plane-sweep generalizations [30, 113].

The requirement of dense output is motivated by modern applications of stereo such as view synthesis and image-based rendering, which require disparity estimates in all image regions, even those that are occluded or without texture. Thus, sparse and feature-based stereo methods are outside the scope of this paper, unless they are followed by a surface-fitting step, e.g., using triangulation, splines, or seed-and-grow methods.

We begin this paper with a review of the goals and scope of this study, which include the need for a coherent taxonomy and a well thought-out evaluation methodology. We also review *disparity space* representations, which play a central role in this paper. In Section 3, we present our taxonomy of dense two-frame correspondence algorithms. Section 4 discusses our current test bed implementation in terms of the major algorithm components, their interactions, and the parameters controlling their behavior. Section 5 describes our evaluation methodology, including the methods we used for acquiring calibrated data sets with known ground truth. In Section 6 we present experiments evaluating the different algorithm components, while Section 7 provides an overall comparison of 20 current stereo algorithms. We conclude in Section 8 with a discussion of planned future work.

2. Motivation and scope

Compiling a complete survey of existing stereo methods, even restricted to dense two-frame methods, would be a formidable task, as a large number of new methods are published every year. It is also arguable whether such a survey would be of much value to other stereo researchers, besides being an obvious catch-all reference. Simply enumerating different approaches is unlikely to yield new insights.

Clearly, a comparative evaluation is necessary to assess the performance of both established and new algorithms and to gauge the progress of the field. The publication of a similar study by Barron *et al.* [8] has had a dramatic effect on the development of optical flow algorithms. Not only is the performance of commonly used algorithms better understood by researchers, but novel publications have to improve in some way on the performance of previously published techniques [86]. A more recent study by Mitiche and Bouthemy [78] reviews a large number of methods for image flow computation and isolates central problems, but does not provide any experimental results.

In stereo correspondence, two previous comparative papers have focused on the performance of sparse feature matchers [54, 19]. Two recent papers [111, 80] have developed new criteria for evaluating the performance of dense stereo matchers for image-based rendering and tele-presence applications. Our work is a continuation of the investigations begun by Szeliski and Zabih [116], which compared the performance of several popular algorithms, but did not provide a detailed taxonomy or as complete a coverage of algorithms. A preliminary version of this paper appeared in the CVPR 2001 Workshop on Stereo and Multi-Baseline Vision [99].

An evaluation of competing algorithms has limited value if each method is treated as a "black box" and only final results are compared. More insights can be gained by examining the individual components of various algorithms. For example, suppose a method based on global energy minimization outperforms other methods. Is the reason a better energy function, or a better minimization technique? Could the technique be improved by substituting different matching costs?

In this paper we attempt to answer such questions by providing a taxonomy of stereo algorithms. The taxonomy is designed to identify the individual components and design decisions that go into a published algorithm. We hope that the taxonomy will also serve to structure the field and to guide researchers in the development of new and better algorithms.

2.1. Computational theory

Any vision algorithm, explicitly or implicitly, makes assumptions about the physical world and the image formation process. In other words, it has an underlying computational theory [74, 72]. For example, how does the algorithm measure the evidence that points in the two images match, i.e., that they are projections of the same scene point? One common assumption is that of Lambertian surfaces, i.e., surfaces whose appearance does not vary with viewpoint. Some algorithms also model specific kinds of camera noise, or differences in gain or bias.

Equally important are assumptions about the world or scene geometry and the visual appearance of objects. Starting from the fact that the physical world consists of piecewise-smooth surfaces, algorithms have built-in smoothness assumptions (often implicit) without which the correspondence problem would be underconstrained and ill-posed. Our taxonomy of stereo algorithms, presented in Section 3, examines both matching assumptions and smoothness assumptions in order to categorize existing stereo methods.

Finally, most algorithms make assumptions about camera calibration and epipolar geometry. This is arguably the best-understood part of stereo vision; we therefore assume in this paper that we are given a pair of rectified images as input. Recent references on stereo camera calibration and rectification include [130, 70, 131, 52, 39].

2.2. Representation

A critical issue in understanding an algorithm is the representation used internally and output externally by the algorithm. Most stereo correspondence methods compute a univalued disparity function $d(x, y)$ with respect to a reference image, which could be one of the input images, or a "cyclopian" view in between some of the images.

Other approaches, in particular multi-view stereo methods, use multi-valued [113], voxel-based [101, 67, 34, 33, 24], or layer-based [125, 5] representations. Still other approaches use full 3D models such as deformable models [120, 121], triangulated meshes [43], or level-set methods [38].

Since our goal is to compare a large number of methods within one common framework, we have chosen to focus on techniques that produce a univalued *disparity map* $d(x, y)$ as their output. Central to such methods is the concept of a *disparity space* (x, y, d). The term *disparity* was first introduced in the human vision literature to describe the difference in location of corresponding features seen by the left and right eyes [72]. (Horizontal disparity is the most commonly studied phenomenon, but vertical disparity is possible if the eyes are verged.)

In computer vision, disparity is often treated as synonymous with inverse depth [20, 85]. More recently, several researchers have defined disparity as a three-dimensional projective transformation (collineation or homography) of 3-D space (X, Y, Z). The enumeration of all possible matches in such a generalized disparity space can be easily achieved with a *plane sweep* algorithm [30, 113], which for every disparity d projects all images onto a common plane using

a perspective projection (homography). (Note that this is different from the meaning of plane sweep in computational geometry.)

In general, we favor the more generalized interpretation of disparity, since it allows the adaptation of the search space to the geometry of the input cameras [113, 94]; we plan to use it in future extensions of this work to multiple images. (Note that plane sweeps can also be generalized to other sweep surfaces such as cylinders [106].)

In this study, however, since all our images are taken on a linear path with the optical axis perpendicular to the camera displacement, the classical inverse-depth interpretation will suffice [85]. The (x, y) coordinates of the disparity space are taken to be coincident with the pixel coordinates of a *reference image* chosen from our input data set. The correspondence between a pixel (x, y) in reference image r and a pixel (x', y') in matching image m is then given by

$$x' = x + s\,d(x, y), \quad y' = y, \tag{1}$$

where $s = \pm 1$ is a sign chosen so that disparities are always positive. Note that since our images are numbered from leftmost to rightmost, the pixels move from right to left.

Once the disparity space has been specified, we can introduce the concept of a *disparity space image* or DSI [127, 18]. In general, a DSI is any image or function defined over a continuous or discretized version of disparity space (x, y, d). In practice, the DSI usually represents the confidence or log likelihood (i.e., *cost*) of a particular match implied by $d(x, y)$.

The goal of a stereo correspondence algorithm is then to produce a univalued function in disparity space $d(x, y)$ that best describes the shape of the surfaces in the scene. This can be viewed as finding a surface embedded in the disparity space image that has some optimality property, such as lowest cost and best (piecewise) smoothness [127]. Figure 1 shows examples of slices through a typical DSI. More figures of this kind can be found in [18].

3. A taxonomy of stereo algorithms

In order to support an informed comparison of stereo matching algorithms, we develop in this section a taxonomy and categorization scheme for such algorithms. We present a set of algorithmic "building blocks" from which a large set of existing algorithms can easily be constructed. Our taxonomy is based on the observation that stereo algorithms generally perform (subsets of) the following four steps [97, 96]:

1. matching cost computation;
2. cost (support) aggregation;
3. disparity computation / optimization; and
4. disparity refinement.

The actual sequence of steps taken depends on the specific algorithm.

For example, *local* (window-based) algorithms, where the disparity computation at a given point depends only on intensity values within a finite window, usually make implicit smoothness assumptions by aggregating support. Some of these algorithms can cleanly be broken down into steps 1, 2, 3. For example, the traditional sum-of-squared-differences (SSD) algorithm can be described as:

1. the matching cost is the squared difference of intensity values at a given disparity;
2. aggregation is done by summing matching cost over square windows with constant disparity;
3. disparities are computed by selecting the minimal (winning) aggregated value at each pixel.

Some local algorithms, however, combine steps 1 and 2 and use a matching cost that is based on a support region, e.g. normalized cross-correlation [51, 19] and the rank transform [129]. (This can also be viewed as a preprocessing step; see Section 3.1.)

On the other hand, *global* algorithms make explicit smoothness assumptions and then solve an optimization problem. Such algorithms typically do not perform an aggregation step, but rather seek a disparity assignment (step 3) that minimizes a global cost function that combines data (step 1) and smoothness terms. The main distinction between these algorithms is the minimization procedure used, e.g., simulated annealing [75, 6], probabilistic (mean-field) diffusion [97], or graph cuts [23].

In between these two broad classes are certain iterative algorithms that do not explicitly state a global function that is to be minimized, but whose behavior mimics closely that of iterative optimization algorithms [73, 97, 132]. Hierarchical (coarse-to-fine) algorithms resemble such iterative algorithms, but typically operate on an image pyramid, where results from coarser levels are used to constrain a more local search at finer levels [126, 90, 11].

3.1. Matching cost computation

The most common pixel-based matching costs include *squared intensity differences* (SD) [51, 1, 77, 107] and *absolute intensity differences* (AD) [58]. In the video processing community, these matching criteria are referred to as the *mean-squared error* (MSE) and *mean absolute difference* (MAD) measures; the term *displaced frame difference* is also often used [118].

More recently, robust measures, including truncated quadratics and contaminated Gaussians have been proposed [15, 16, 97]. These measures are useful because they limit the influence of mismatches during aggregation.

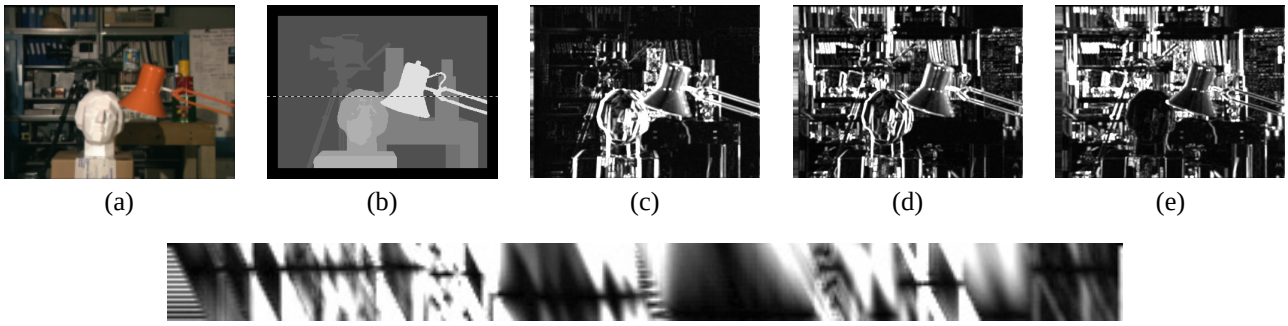


Figure 1: *Slices through a typical disparity space image (DSI): (a) original color image; (b) ground-truth disparities; (c–e) three (x, y) slices for $d = 10, 16, 21$; (e) an (x, d) slice for $y = 151$ (the dashed line in Figure (b)). Different dark (matching) regions are visible in Figures (c–e), e.g., the bookshelves, table and cans, and head statue, while three different disparity levels can be seen as horizontal lines in the (x, d) slice (Figure (f)). Note the dark bands in the various DSIs, which indicate regions that match at this disparity. (Smaller dark regions are often the result of textureless regions.)*

Other traditional matching costs include normalized cross-correlation [51, 93, 19], which behaves similar to sum-of-squared-differences (SSD), and binary matching costs (i.e., match / no match) [73], based on binary features such as edges [4, 50, 27] or the sign of the Laplacian [82]. Binary matching costs are not commonly used in dense stereo methods, however.

Some costs are insensitive to differences in camera gain or bias, for example gradient-based measures [100, 95] and non-parametric measures such as rank and census transforms [129]. Of course, it is also possible to correct for different camera characteristics by performing a preprocessing step for bias-gain or histogram equalization [48, 32]. Other matching criteria include phase and filter-bank responses [74, 63, 56, 57]. Finally, Birchfield and Tomasi have proposed a matching cost that is insensitive to image sampling [12]. Rather than just comparing pixel values shifted by integral amounts (which may miss a valid match), they compare each pixel in the reference image against a linearly interpolated function of the other image.

The matching cost values over all pixels and all disparities form the initial disparity space image $C_0(x, y, d)$. While our study is currently restricted to two-frame methods, the initial DSI can easily incorporate information from more than two images by simply summing up the cost values for each matching image m, since the DSI is associated with a fixed reference image r (Equation (1)). This is the idea behind multiple-baseline SSSD and SSAD methods [85, 62, 81]. As mentioned in Section 2.2, this idea can be generalized to arbitrary camera configurations using a plane sweep algorithm [30, 113].

3.2. Aggregation of cost

Local and window-based methods aggregate the matching cost by summing or averaging over a *support region* in the DSI $C(x, y, d)$. A support region can be either two-dimensional at a fixed disparity (favoring fronto-parallel surfaces), or three-dimensional in x-y-d space (supporting slanted surfaces). Two-dimensional evidence aggregation has been implemented using square windows or Gaussian convolution (traditional), multiple windows anchored at different points, i.e., shiftable windows [2, 18], windows with adaptive sizes [84, 60, 124, 61], and windows based on connected components of constant disparity [22]. Three-dimensional support functions that have been proposed include limited disparity difference [50], limited disparity gradient [88], and Prazdny's coherence principle [89].

Aggregation with a fixed support region can be performed using 2D or 3D convolution,

$$C(x, y, d) = w(x, y, d) * C_0(x, y, d), \tag{2}$$

or, in the case of rectangular windows, using efficient (moving average) box-filters. Shiftable windows can also be implemented efficiently using a separable sliding min-filter (Section 4.2). A different method of aggregation is *iterative diffusion*, i.e., an aggregation (or averaging) operation that is implemented by repeatedly adding to each pixel's cost the weighted values of its neighboring pixels' costs [114, 103, 97].

3.3. Disparity computation and optimization

Local methods. In local methods, the emphasis is on the matching cost computation and on the cost aggregation steps. Computing the final disparities is trivial: simply choose at each pixel the disparity associated with the minimum cost value. Thus, these methods perform a local "winner-take-all" (WTA) optimization at each pixel. A limitation of this approach (and many other correspondence algorithms) is that uniqueness of matches is only enforced for one image (the *reference image*), while points in the other image might

get matched to multiple points.

Global optimization. In contrast, global methods perform almost all of their work during the disparity computation phase and often skip the aggregation step. Many global methods are formulated in an energy-minimization framework [119]. The objective is to find a disparity function d that minimizes a global energy,

$$E(d) = E_{data}(d) + \lambda E_{smooth}(d). \quad (3)$$

The data term, $E_{data}(d)$, measures how well the disparity function d agrees with the input image pair. Using the disparity space formulation,

$$E_{data}(d) = \sum_{(x,y)} C(x, y, d(x, y)), \quad (4)$$

where C is the (initial or aggregated) matching cost DSI.

The smoothness term $E_{smooth}(d)$ encodes the smoothness assumptions made by the algorithm. To make the optimization computationally tractable, the smoothness term is often restricted to only measuring the differences between neighboring pixels' disparities,

$$E_{smooth}(d) = \sum_{(x,y)} \rho(d(x, y) - d(x+1, y)) + \rho(d(x, y) - d(x, y+1)), \quad (5)$$

where ρ is some monotonically increasing function of disparity difference. (An alternative to smoothness functionals is to use a lower-dimensional representation such as splines [112].)

In regularization-based vision [87], ρ is a quadratic function, which makes d smooth everywhere and may lead to poor results at object boundaries. Energy functions that do not have this problem are called *discontinuity-preserving* and are based on robust ρ functions [119, 16, 97]. Geman and Geman's seminal paper [47] gave a Bayesian interpretation of these kinds of energy functions [110] and proposed a discontinuity-preserving energy function based on Markov Random Fields (MRFs) and additional *line processes*. Black and Rangarajan [16] show how line processes can be often be subsumed by a robust regularization framework.

The terms in E_{smooth} can also be made to depend on the intensity differences, e.g.,

$$\rho_d(d(x, y) - d(x+1, y)) \cdot \rho_I(\|I(x, y) - I(x+1, y)\|), \quad (6)$$

where ρ_I is some monotonically *decreasing* function of intensity differences that lowers smoothness costs at high intensity gradients. This idea [44, 42, 18, 23] encourages disparity discontinuities to coincide with intensity/color edges and appears to account for some of the good performance of global optimization approaches.

Once the global energy has been defined, a variety of algorithms can be used to find a (local) minimum. Traditional approaches associated with regularization and Markov Random Fields include continuation [17], simulated annealing [47, 75, 6], highest confidence first [28], and mean-field annealing [45].

More recently, *max-flow* and *graph-cut* methods have been proposed to solve a special class of global optimization problems [92, 55, 23, 123, 65]. Such methods are more efficient than simulated annealing and have produced good results.

Dynamic programming. A different class of global optimization algorithms are those based on *dynamic programming*. While the 2D-optimization of Equation (3) can be shown to be NP-hard for common classes of smoothness functions [123], dynamic programming can find the global minimum for independent scanlines in polynomial time. Dynamic programming was first used for stereo vision in sparse, edge-based methods [3, 83]. More recent approaches have focused on the dense (intensity-based) scanline optimization problem [10, 9, 46, 31, 18, 13]. These approaches work by computing the minimum-cost path through the matrix of all pairwise matching costs between two corresponding scanlines. Partial occlusion is handled explicitly by assigning a group of pixels in one image to a single pixel in the other image. Figure 2 shows one such example.

Problems with dynamic programming stereo include the selection of the right cost for occluded pixels and the difficulty of enforcing inter-scanline consistency, although several methods propose ways of addressing the latter [83, 9, 31, 18, 13]. Another problem is that the dynamic programming approach requires enforcing the *monotonicity* or *ordering constraint* [128]. This constraint requires that the relative ordering of pixels on a scanline remain the same between the two views, which may not be the case in scenes containing narrow foreground objects.

Cooperative algorithms. Finally, *cooperative* algorithms, inspired by computational models of human stereo vision, were among the earliest methods proposed for disparity computation [36, 73, 76, 114]. Such algorithms iteratively perform local computations, but use nonlinear operations that result in an overall behavior similar to global optimization algorithms. In fact, for some of these algorithms, it is possible to explicitly state a global function that is being minimized [97]. Recently, a promising variant of Marr and Poggio's original cooperative algorithm has been developed [132].

3.4. Refinement of disparities

Most stereo correspondence algorithms compute a set of disparity estimates in some discretized space, e.g., for inte-

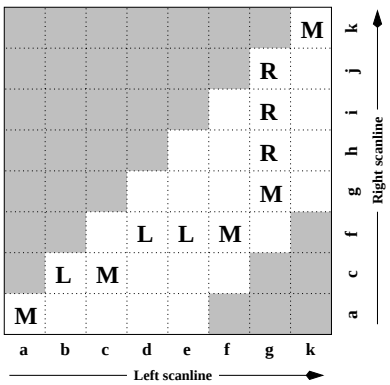


Figure 2: *Stereo matching using dynamic programming. For each pair of corresponding scanlines, a minimizing path through the matrix of all pairwise matching costs is selected. Lowercase letters (**a**–**k**) symbolize the intensities along each scanline. Uppercase letters represent the selected path through the matrix. Matches are indicated by **M**, while partially occluded points (which have a fixed cost) are indicated by **L** and **R**, corresponding to points only visible in the left and right image, respectively. Usually, only a limited disparity range is considered, which is 0–4 in the figure (indicated by the non-shaded squares). Note that this diagram shows an "unskewed" x-d slice through the DSI.*

ger disparities (exceptions include continuous optimization techniques such as optic flow [11] or splines [112]). For applications such as robot navigation or people tracking, these may be perfectly adequate. However for image-based rendering, such quantized maps lead to very unappealing view synthesis results (the scene appears to be made up of many thin shearing layers). To remedy this situation, many algorithms apply a sub-pixel refinement stage after the initial discrete correspondence stage. (An alternative is to simply start with more discrete disparity levels.)

Sub-pixel disparity estimates can be computed in a variety of ways, including iterative gradient descent and fitting a curve to the matching costs at discrete disparity levels [93, 71, 122, 77, 60]. This provides an easy way to increase the resolution of a stereo algorithm with little additional computation. However, to work well, the intensities being matched must vary smoothly, and the regions over which these estimates are computed must be on the same (correct) surface.

Recently, some questions have been raised about the advisability of fitting correlation curves to integer-sampled matching costs [105]. This situation may even be worse when sampling-insensitive dissimilarity measures are used [12]. We investigate this issue in Section 6.4 below.

Besides sub-pixel computations, there are of course other ways of post-processing the computed disparities. Occluded areas can be detected using cross-checking (comparing left-to-right and right-to-left disparity maps) [29, 42]. A median filter can be applied to "clean up" spurious mismatches, and holes due to occlusion can be filled by surface fitting or by distributing neighboring disparity estimates [13, 96]. In our implementation we are not performing such clean-up steps since we want to measure the performance of the raw algorithm components.

3.5. Other methods

Not all dense two-frame stereo correspondence algorithms can be described in terms of our basic taxonomy and representations. Here we briefly mention some additional algorithms and representations that are not covered by our framework.

The algorithms described in this paper first enumerate all possible matches at all possible disparities, then select the best set of matches in some way. This is a useful approach when a large amount of ambiguity may exist in the computed disparities. An alternative approach is to use methods inspired by classic (infinitesimal) optic flow computation. Here, images are successively warped and motion estimates incrementally updated until a satisfactory registration is achieved. These techniques are most often implemented within a coarse-to-fine hierarchical refinement framework [90, 11, 8, 112].

A univalued representation of the disparity map is also not essential. Multi-valued representations, which can represent several depth values along each line of sight, have been extensively studied recently, especially for large multi-view data set. Many of these techniques use a *voxel-based* representation to encode the reconstructed colors and spatial occupancies or opacities [113, 101, 67, 34, 33, 24]. Another way to represent a scene with more complexity is to use multiple layers, each of which can be represented by a plane plus residual parallax [5, 14, 117]. Finally, deformable surfaces of various kinds have also been used to perform 3D shape reconstruction from multiple images [120, 121, 43, 38].

3.6. Summary of methods

Table 1 gives a summary of some representative stereo matching algorithms and their corresponding taxonomy, i.e., the matching cost, aggregation, and optimization techniques used by each. The methods are grouped to contrast different matching costs (top), aggregation methods (middle), and optimization techniques (third section), while the last section lists some papers outside the framework. As can be seen from this table, quite a large subset of the possible algorithm design space has been explored over the years, albeit not very systematically.

4. Implementation

We have developed a stand-alone, portable C++ implementation of several stereo algorithms. The implementation is closely tied to the taxonomy presented in Section 3 and currently includes window-based algorithms, diffusion algo-

Method	Matching cost	Aggregation	Optimization
SSD (traditional)	squared difference	square window	WTA
Hannah [51]	cross-correlation	(square window)	WTA
Nishihara [82]	binarized filters	square window	WTA
Kass [63]	filter banks	-none-	WTA
Fleet *et al.* [40]	phase	-none-	phase-matching
Jones and Malik [57]	filter banks	-none-	WTA
Kanade [58]	absolute difference	square window	WTA
Scharstein [95]	gradient-based	Gaussian	WTA
Zabih and Woodfill [129]	rank transform	(square window)	WTA
Cox *et al.* [32]	histogram eq.	-none-	DP
Frohlinghaus and Buhmann [41]	wavelet phase	-none-	phase-matching
Birchfield and Tomasi [12]	shifted abs. diff	-none-	DP
Marr and Poggio [73]	binary images	iterative aggregation	WTA
Prazdny [89]	binary images	3D aggregation	WTA
Szeliski and Hinton [114]	binary images	iterative 3D aggregation	WTA
Okutomi and Kanade [84]	squared difference	adaptive window	WTA
Yang *et al.* [127]	cross-correlation	non-linear filtering	hier. WTA
Shah [103]	squared difference	non-linear diffusion	regularization
Boykov *et al.* [22]	thresh. abs. diff.	connected-component	WTA
Scharstein and Szeliski [97]	robust sq. diff.	iterative 3D aggregation	mean-field
Zitnick and Kanade [132]	squared difference	iterative aggregation	WTA
Veksler [124]	abs. diff - avg.	adaptive window	WTA
Quam [90]	cross-correlation	-none-	hier. warp
Barnard [6]	squared difference	-none-	SA
Geiger *et al.* [46]	squared difference	shiftable window	DP
Belhumeur [9]	squared difference	-none-	DP
Cox *et al.* [31]	squared difference	-none-	DP
Ishikawa and Geiger [55]	squared difference	-none-	graph cut
Roy and Cox [92]	squared difference	-none-	graph cut
Bobick and Intille [18]	absolute difference	shiftable window	DP
Boykov *et al.* [23]	squared difference	-none-	graph cut
Kolmogorov and Zabih [65]	squared difference	-none-	graph cut
Birchfield and Tomasi [14]	shifted abs. diff.	-none-	GC + planes
Tao *et al.* [117]	squared difference	(color segmentation)	WTA + regions

Table 1: *Summary taxonomy of several dense two-frame stereo correspondence methods. The methods are grouped to contrast different matching costs (top), aggregation methods (middle), and optimization techniques (third section). The last section lists some papers outside our framework. Key to abbreviations: hier. – hierarchical (coarse-to-fine), WTA – winner-take-all, DP – dynamic programming, SA – simulated annealing, GC – graph cut.*

rithms, as well as global optimization methods using dynamic programming, simulated annealing, and graph cuts. While many published methods include special features and post-processing steps to improve the results, we have chosen to implement the basic versions of such algorithms, in order to assess their respective merits most directly.

The implementation is modular and can easily be extended to include other algorithms or their components. We plan to add several other algorithms in the near future, and we hope that other authors will contribute their methods to our framework as well. Once a new algorithm has been integrated, it can easily be compared with other algorithms using our evaluation module, which can measure disparity error and reprojection error (Section 5.1). The implementation contains a sophisticated mechanism for specifying parameter values that supports recursive script files for exhaustive performance comparisons on multiple data sets.

We provide a high-level description of our code using the same division into four parts as in our taxonomy. Within our code, these four sections are (optionally) executed in sequence, and the performance/quality evaluator is then invoked. A list of the most important algorithm parameters is given in Table 2.

4.1. Matching cost computation

The simplest possible matching cost is the squared or absolute difference in color / intensity between corresponding pixels (*match_fn*). To approximate the effect of a robust matching score [16, 97], we truncate the matching score to a maximal value *match_max*. When color images are being compared, we sum the squared or absolute intensity difference in each channel before applying the clipping. If fractional disparity evaluation is being performed (*disp_step* < 1), each scanline is first interpolated up using either a linear or cubic interpolation filter (*match_interp*) [77]. We also optionally apply Birchfield and Tomasi's sampling insensitive interval-based matching criterion (*match_interval*) [12], i.e., we take the minimum of the pixel matching score and the score at $\pm\frac{1}{2}$-step displacements, or 0 if there is a sign change in either interval. We apply this criterion separately to each color channel, which is not physically plausible (the sub-pixel shift must be consistent across channels), but is easier to implement.

4.2. Aggregation

The aggregation section of our test bed implements some commonly used aggregation methods (*aggr_fn*):

- Box filter: use a separable moving average filter (add one right/bottom value, subtract one left/top). This implementation trick makes such window-based aggregation insensitive to window size in terms of computation time and accounts for the fast performance seen in real-time matchers [59, 64].

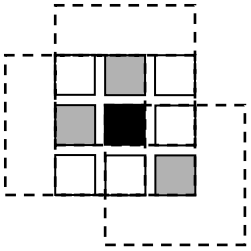


Figure 3: *Shiftable window. The effect of trying all* 3×3 *shifted windows around the black pixel is the same as taking the minimum matching score across all* centered *(non-shifted) windows in the same neighborhood. (Only 3 of the neighboring shifted windows are shown here for clarity.)*

- Binomial filter: use a separable FIR (finite impulse response) filter. We use the coefficients $1/16\{1, 4, 6, 4, 1\}$, the same ones used in Burt and Adelson's [26] Laplacian pyramid.

Other convolution kernels could also be added later, as could recursive (bi-directional) IIR filtering, which is a very efficient way to obtain large window sizes [35]. The width of the box or convolution kernel is controlled by *aggr_window_size*.

To simulate the effect of shiftable windows [2, 18, 117], we can follow this aggregation step with a separable square min-filter. The width of this filter is controlled by the parameter *aggr_minfilter*. The cascaded effect of a box-filter and an equal-sized min-filter is the same as evaluating a complete set of shifted windows, since the value of a shifted window is the same as that of a centered window at some neighboring pixel (Figure 3). This step adds very little additional computation, since a moving 1-D min-filter can be computed efficiently by only recomputing the min when a minimum value leaves the window. The value of *aggr_minfilter* can be less than that of *aggr_window_size*, which simulates the effect of a partially shifted window. (The converse doesn't make much sense, since the window then no longer includes the reference pixel.)

We have also implemented all of the diffusion methods developed in [97] except for local stopping, i.e., regular diffusion, the membrane model, and Bayesian (mean-field) diffusion. While this last algorithm can also be considered an optimization method, we include it in the aggregation module since it resembles other iterative aggregation algorithms closely. The maximum number of aggregation iterations is controlled by *aggr_iter*. Other parameters controlling the diffusion algorithms are listed in Table 2.

4.3. Optimization

Once we have computed the (optionally aggregated) costs, we need to determine which discrete set of disparities best represents the scene surface. The algorithm used to deter-

Name	Typical values	Description
disp_min	0	smallest disparity
disp_max	15	largest disparity
disp_step	0.5	disparity step size
match_fn	SD, AD	matching function
match_interp	Linear, Cubic	interpolation function
match_max	20	maximum difference for truncated SAD/SSD
match_interval	false	1/2 disparity match [12]
aggr_fn	Box, Binomial	aggregation function
aggr_window_size	9	size of window
aggr_minfilter	9	spatial min-filter (shiftable window)
aggr_iter	1	number of aggregation iterations
diff_lambda	0.15	parameter λ for regular and membrane diffusion
diff_beta	0.5	parameter β for membrane diffusion
diff_scale_cost	0.01	scale of cost values (needed for Bayesian diffusion)
diff_mu	0.5	parameter μ for Bayesian diffusion
diff_sigmaP	0.4	parameter σ_P for robust prior of Bayesian diffusion
diff_epsP	0.01	parameter ϵ_P for robust prior of Bayesian diffusion
opt_fn	WTA, DP, SA, GC	optimization function
opt_smoothness	1.0	weight of smoothness term (λ)
opt_grad_thresh	8.0	threshold for magnitude of intensity gradient
opt_grad_penalty	2.0	smoothness penalty factor if gradient is too small
opt_occlusion_cost	20	cost for occluded pixels in DP algorithm
opt_sa_var	Gibbs, Metropolis	simulated annealing update rule
opt_sa_start_T	10.0	starting temperature
opt_sa_end_T	0.01	ending temperature
opt_sa_schedule	Linear	annealing schedule
refine_subpix	true	fit sub-pixel value to local correlation
eval_bad_thresh	1.0	acceptable disparity error
eval_textureless_width	3	box filter width applied to $\|\nabla_x I\|^2$
eval_textureless_thresh	4.0	threshold applied to filtered $\|\nabla_x I\|^2$
eval_disp_gap	2.0	disparity jump threshold
eval_discont_width	9	width of discontinuity region
eval_ignore_border	10	number of border pixels to ignore
eval_partial_shuffle	0.2	analysis interval for prediction error

Table 2: *The most important stereo algorithm parameters of our implementation.*

mine this is controlled by *opt_fn*, and can be one of:

- winner-take-all (WTA);
- dynamic programming (DP);
- scanline optimization (SO);
- simulated annealing (SA);
- graph cut (GC).

The winner-take-all method simply picks the lowest (aggregated) matching cost as the selected disparity at each pixel. The other methods require (in addition to the matching cost) the definition of a smoothness cost. Prior to invoking one of the optimization algorithms, we set up tables containing the values of ρ_d in Equation (6) and precompute the spatially varying weights $\rho_I(x, y)$. These tables are controlled by the parameters *opt_smoothness*, which controls the overall scale of the smoothness term (i.e., λ in Equation (3)), and the parameters *opt_grad_thresh* and *opt_grad_penalty*, which control the gradient-dependent smoothness costs. We currently use the smoothness terms defined by Veksler [123]:

$$\rho_I(\Delta I) = \begin{cases} p & \text{if} \quad \Delta I < \textit{opt\_grad\_thresh} \\ 1 & \text{if} \quad \Delta I \geq \textit{opt\_grad\_thresh}, \end{cases} \tag{7}$$

where $p =$ *opt_grad_penalty*. Thus, the smoothness cost is multiplied by p for low intensity gradient to encourage disparity jumps to coincide with intensity edges. All of the optimization algorithms minimize the same objective function, enabling a more meaningful comparison of their performance.

Our first global optimization technique, DP, is a dynamic programming method similar to the one proposed by Bobick and Intille [18]. The algorithm works by computing the minimum-cost path through each x-d slice in the DSI (see Figure 2). Every point in this slice can be in one of three states: M (match), L (left-visible only), or R (right-visible only). Assuming the ordering constraint is being enforced, a valid path can take at most three directions at a point, each associated with a deterministic state change. Using dynamic programming, the minimum cost of all paths to a point can be accumulated efficiently. Points in state M are simply charged the matching cost at this point in the DSI. Points in states L and R are charged a fixed *occlusion cost* (*opt_occlusion_cost*). Before evaluating the final disparity map, we fill all occluded pixels with the nearest background disparity value on the same scanline.

The DP stereo algorithm is fairly sensitive to this parameter (see Section 6). Bobick and Intille address this problem by precomputing *ground control points* (GCPs) that are then used to constrain the paths through the DSI slice. GCPs are high-confidence matches that are computed using SAD and shiftable windows. At this point we are not using GCPs in our implementation since we are interested in comparing the basic version of different algorithms. However, GCPs are potentially useful in other algorithms as well, and we plan to add them to our implementation in the future.

Our second global optimization technique, *scanline optimization* (SO), is a simple (and, to our knowledge, novel) approach designed to assess different smoothness terms. Like the previous method, it operates on individual x-d DSI slices and optimizes one scanline at a time. However, the method is asymmetric and does not utilize visibility or ordering constraints. Instead, a d value is assigned at each point x such that the overall cost along the scanline is minimized. (Note that without a smoothness term, this would be equivalent to a winner-take-all optimization.) The global minimum can again be computed using dynamic programming; however, unlike in traditional (symmetric) DP algorithms, the ordering constraint does not need to be enforced, and no occlusion cost parameter is necessary. Thus, the SO algorithm solves the same optimization problem as the graph-cut algorithm described below, except that vertical smoothness terms are ignored.

Both DP and SO algorithms suffer from the well-known difficulty of enforcing inter-scanline consistency, resulting in horizontal "streaks" in the computed disparity map. Bobick and Intille's approach to this problem is to detect edges in the DSI slice and to lower the occlusion cost for paths along those edges. This has the effect of aligning depth discontinuities with intensity edges. In our implementation, we achieve the same goal by using an intensity-dependent smoothness cost (Equation (6)), which, in our DP algorithm, is charged at all L-M and R-M state transitions.

Our implementation of simulated annealing supports both the Metropolis variant (where downhill steps are always taken, and uphill steps are sometimes taken), and the Gibbs Sampler, which chooses among several possible states according to the full marginal distribution [47]. In the latter case, we can either select one new state (disparity) to flip to at random, or evaluate all possible disparities at a given pixel. Our current annealing schedule is linear, although we plan to add a logarithmic annealing schedule in the future.

Our final global optimization method, GC, implements the α-β swap move algorithm described in [23, 123]. (We plan to implement the α-expansion in the future.) We randomize the α-β pairings at each (inner) iteration and stop the algorithm when no further (local) energy improvements are possible.

4.4. Refinement

The sub-pixel refinement of disparities is controlled by the boolean variable *refine_subpix*. When this is enabled, the three aggregated matching cost values around the winning disparity are examined to compute the sub-pixel disparity estimate. (Note that if the initial DSI was formed with fractional disparity steps, these are really sub-sub-pixel values. A more appropriate name might be *floating point disparity*

values.) A parabola is fit to these three values (the three ending values are used if the winning disparity is either *disp_min* or *disp_max*). If the curvature is positive and the minimum of the parabola is within a half-step of the winning disparity (and within the search limits), this value is used as the final disparity estimate.

In future work, we would like to investigate whether initial or aggregated matching scores should be used, or whether some other approach, such as Lucas-Kanade, might yield higher-quality estimates [122].

5. Evaluation methodology

In this section, we describe the quality metrics we use for evaluating the performance of stereo correspondence algorithms and the techniques we used for acquiring our image data sets and ground truth estimates.

5.1. Quality metrics

To evaluate the performance of a stereo algorithm or the effects of varying some of its parameters, we need a quantitative way to estimate the quality of the computed correspondences. Two general approaches to this are to compute error statistics with respect to some ground truth data [8] and to evaluate the synthetic images obtained by warping the reference or unseen images by the computed disparity map [111].

In the current version of our software, we compute the following two quality measures based on known ground truth data:

1. RMS (root-mean-squared) error (measured in disparity units) between the computed disparity map $d_C(x, y)$ and the ground truth map $d_T(x, y)$, i.e.,

$$R = \left(\frac{1}{N} \sum_{(x,y)} |d_C(x, y) - d_T(x, y)|^2 \right)^{\frac{1}{2}}, \qquad (8)$$

   where N is the total number of pixels.

2. Percentage of bad matching pixels,

$$B = \frac{1}{N} \sum_{(x,y)} (|d_C(x, y) - d_T(x, y)| > \delta_d), \qquad (9)$$

   where δ_d (*eval_bad_thresh*) is a disparity error tolerance. For the experiments in this paper we use $\delta_d = 1.0$, since this coincides with some previously published studies [116, 132, 65].

In addition to computing these statistics over the whole image, we also focus on three different kinds of regions. These regions are computed by pre-processing the reference image and ground truth disparity map to yield the following three binary segmentations (Figure 4):

Name	Symb.	Description
rms_error_all	R	RMS disparity error
rms_error_nonocc	$R_{\overline{\mathcal{O}}}$	" (no occlusions)
rms_error_occ	$R_{\mathcal{O}}$	" (at occlusions)
rms_error_textured	$R_{\mathcal{T}}$	" (textured)
rms_error_textureless	$R_{\overline{\mathcal{T}}}$	" (textureless)
rms_error_discont	$R_{\mathcal{D}}$	" (near discontinuities)
bad_pixels_all	B	bad pixel percentage
bad_pixels_nonocc	$B_{\overline{\mathcal{O}}}$	" (no occlusions)
bad_pixels_occ	$B_{\mathcal{O}}$	" (at occlusions)
bad_pixels_textured	$B_{\mathcal{T}}$	" (textured)
bad_pixels_textureless	$B_{\overline{\mathcal{T}}}$	" (textureless)
bad_pixels_discont	$B_{\mathcal{D}}$	" (near discontinuities)
predict_err_near	P_{-}	view extr. error (near)
predict_err_middle	$P_{1/2}$	view extr. error (mid)
predict_err_match	P_1	view extr. error (match)
predict_err_far	P_{+}	view extr. error (far)

Table 3: *Error (quality) statistics computed by our evaluator. See the notes in the text regarding the treatment of occluded regions.*

- textureless regions $\mathcal{T}$: regions where the squared horizontal intensity gradient averaged over a square window of a given size (*eval_textureless_width*) is below a given threshold (*eval_textureless_thresh*);
- occluded regions $\mathcal{O}$: regions that are occluded in the matching image, i.e., where the forward-mapped disparity lands at a location with a larger (nearer) disparity; and
- depth discontinuity regions $\mathcal{D}$: pixels whose neighboring disparities differ by more than *eval_disp_gap*, dilated by a window of width *eval_discont_width*.

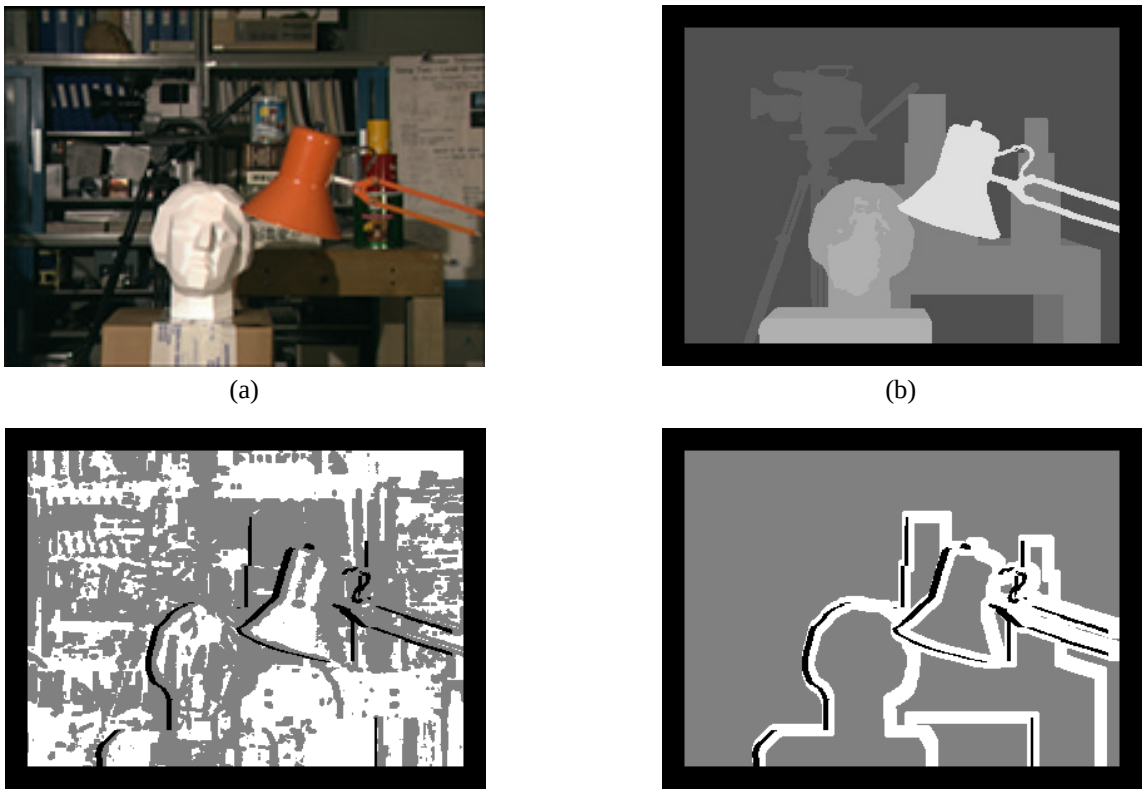
These regions were selected to support the analysis of matching results in typical problem areas. For the experiments in this paper we use the values listed in Table 2.

The statistics described above are computed for each of the three regions and their complements, e.g.,

$$B_{\mathcal{T}} = \frac{1}{N_{\mathcal{T}}} \sum_{(x,y) \in \mathcal{T}} (|d_c(x, y) - d_t(x, y)| < \delta_d),$$

and so on for $R_{\mathcal{T}}$, $B_{\overline{\mathcal{T}}}$, ..., $R_{\overline{\mathcal{D}}}$.

Table 3 gives a complete list of the statistics we collect. Note that for the textureless, textured, and depth discontinuity statistics, we exclude pixels that are in occluded regions, on the assumption that algorithms generally do not produce meaningful results in such occluded regions. Also, we exclude a border of *eval_ignore_border* pixels when computing all statistics, since many algorithms do not compute meaningful disparities near the image boundaries.



(a) (b)

(c) (d)

Figure 4: *Segmented region maps: (a) original image, (b) true disparities, (c) textureless regions (white) and occluded regions (black), (d) depth discontinuity regions (white) and occluded regions (black).*

Figure 5: *Series of forward-warped reference images. The reference image is the middle one, the matching image is the second from the right. Pixels that are invisible (gaps) are shown in light magenta.*

Figure 6: *Series of inverse-warped original images. The reference image is the middle one, the matching image is the second from the right. Pixels that are invisible are shown in light magenta. Viewing this sequence (available on our web site) as an animation loop is a good way to check for correct rectification, other misalignments, and quantization effects.*

The second major approach to gauging the quality of reconstruction algorithms is to use the color images and disparity maps to predict the appearance of other views [111]. Here again there are two major flavors possible:

1. Forward warp the reference image by the computed disparity map to a different (potentially unseen) view (Figure 5), and compare it against this new image to obtain a *forward prediction error*.

2. Inverse warp a new view by the computed disparity map to generate a *stabilized* image (Figure 6), and compare it against the reference image to obtain an *inverse prediction error*.

There are pros and cons to either approach.

The forward warping algorithm has to deal with tearing problems: if a single-pixel splat is used, gaps can arise even between adjacent pixels with similar disparities. One possible solution would be to use a two-pass renderer [102]. Instead, we render each pair of neighboring pixel as an interpolated color line in the destination image (i.e., we use *Gouraud shading*). If neighboring pixels differ by more that a disparity of *eval_disp_gap*, the segment is replaced by single pixel spats at both ends, which results in a visible tear (light magenta regions in Figure 5).

For inverse warping, the problem of gaps does not occur. Instead, we get "ghosted" regions when pixels in the reference image are not actually visible in the source. We eliminate such pixels by checking for visibility (occlusions) first, and then drawing these pixels in a special color (light magenta in Figure 6). We have found that looking at the inverse warped sequence, based on the ground-truth disparities, is a very good way to determine if the original sequence is properly calibrated and rectified.

In computing the prediction error, we need to decide how to treat gaps. Currently, we ignore pixels flagged as gaps in computing the statistics and report the percentage of such missing pixels. We can also optionally compensate for small misregistrations [111]. To do this, we convert each pixel in the original and predicted image to an interval, by blending the pixel's value with some fraction *eval_partial_shuffle* of its neighboring pixels min and max values. This idea is a generalization of the sampling-insensitive dissimilarity measure [12] and the shuffle transformation of [66]. The reported difference is then the (signed) distance between the two computed intervals. We plan to investigate these and other sampling-insensitive matching costs in the future [115].

5.2. Test data

To quantitatively evaluate our correspondence algorithms, we require data sets that either have a ground truth disparity map, or a set of additional views that can be used for prediction error test (or preferably both).

We have begun to collect such a database of images, building upon the methodology introduced in [116]. Each image sequence consists of 9 images, taken at regular intervals with a camera mounted on a horizontal translation stage, with the camera pointing perpendicularly to the direction of motion. We use a digital high-resolution camera (Canon G1) set in manual exposure and focus mode and rectify the images using tracked feature points. We then downsample the original 2048×1536 images to 512×384 using a high-quality 8-tap filter and finally crop the images to normalize the motion of background objects to a few pixels per frame.

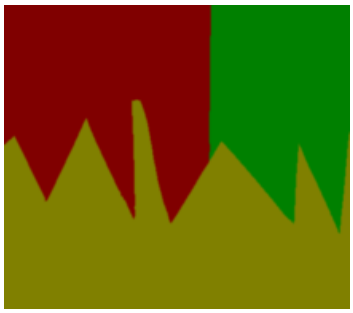
All of the sequences we have captured are made up of piecewise planar objects (typically posters or paintings, some with cut-out edges). Before downsampling the images, we hand-label each image into its piecewise planar components (Figure 7). We then use a direct alignment technique on each planar region [5] to estimate the affine motion of each patch. The horizontal component of these motions is then used to compute the ground truth disparity. In future work we plan to extend our acquisition methodology to handle scenes with quadric surfaces (e.g., cylinders, cones, and spheres).

Of the six image sequences we acquired, all of which are available on our web page, we have selected two ("Sawtooth" and "Venus") for the experimental study in this paper. We also use the University of Tsukuba "head and lamp" data set [81], a 5×5 array of images together with hand-labeled integer ground-truth disparities for the center image. Finally, we use the monochromatic "Map" data set first introduced by Szeliski and Zabih [116], which was taken with a Point Grey Research trinocular stereo camera, and whose ground-truth disparity map was computed using the piecewise planar technique described above. Figure 7 shows the reference image and the ground-truth disparities for each of these four sequences. We exclude a border of 18 pixels in the Tsukuba images, since no ground-truth disparity values are provided there. For all other images, we use *eval_ignore_border* $= 10$ for the experiments reported in this paper.

In the future, we hope to add further data sets to our collection of "standard" test images, in particular other sequences from the University of Tsukuba, and the GRASP Laboratory's "Buffalo Bill" data set with registered laser range finder ground truth [80]. There may also be suitable images among the CMU Computer Vision Home Page data sets. Unfortunately, we cannot use data sets for which only a sparse set of feature matches has been computed [19, 54].

It should be noted that high-quality ground-truth data is critical for a meaningful performance evaluation. Accurate sub-pixel disparities are hard to come by, however. The ground-truth data for the Tsukuba images, for example, is strongly quantized since it only provides integer disparity estimates for a very small disparity range ($d = 5 \ldots 14$). This is clearly visible when the images are stabilized using

Sawtooth ref. image



planar regions

ground-truth disparities

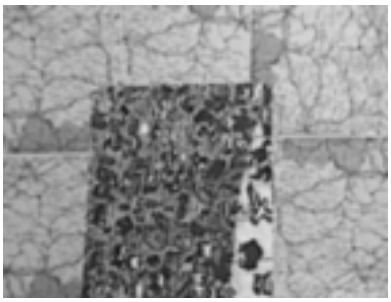
Venus ref. image

planar regions

ground-truth disparities

Tsukuba ref. image

ground-truth disparities



Map ref. image

ground-truth disparities

Figure 7: *Stereo images with ground truth used in this study. The Sawtooth and Venus images are two of our new 9-frame stereo sequences of planar objects. The figure shows the reference image, the planar region labeling, and the ground-truth disparities. We also use the familiar Tsukuba "head and lamp" data set, and the monochromatic Map image pair.*

the ground-truth data and viewed in a video loop. In contrast, the ground-truth disparities for our piecewise planar scenes have high (subpixel) precision, but at the cost of limited scene complexity. To provide an adequate challenge for the best-performing stereo methods, new stereo test images with complex scenes *and* sub-pixel ground truth will soon be needed.

Synthetic images have been used extensively for qualitative evaluations of stereo methods, but they are often restricted to simple geometries and textures (e.g., random-dot stereograms). Furthermore, issues arising with real cameras are seldom modeled, e.g., aliasing, slight misalignment, noise, lens aberrations, and fluctuations in gain and bias. Consequently, results on synthetic images usually do not extrapolate to images taken with real cameras. We have experimented with the University of Bonn's synthetic "Corridor" data set [41], but have found that the clean, noise-free images are unrealistically easy to solve, while the noise-contaminated versions are too difficult due to the complete lack of texture in much of the scene. There is a clear need for synthetic, photo-realistic test imagery that properly models real-world imperfections, while providing accurate ground truth.

6. Experiments and results

In this section, we describe the experiments used to evaluate the individual building blocks of stereo algorithms. Using our implementation framework, we examine the four main algorithm components identified in Section 3 (matching cost, aggregation, optimization, and sub-pixel fitting) In Section 7, we perform an overall comparison of a large set of stereo algorithms, including other authors' implementations. We use the Tsukuba, Sawtooth, Venus, and Map data sets in all experiments and report results on subsets of these images. The complete set of results (all experiments run on all data sets) is available on our web site at `www.middlebury.edu/stereo`.

Using the evaluation measures presented in Section 5.1, we focus on common problem areas for stereo algorithms. Of the 12 ground-truth statistics we collect (Table 3), we have chosen three as the most important subset. First, as a measure of overall performance, we use $B_{\overline{\mathcal{O}}}$, the percentage of bad pixels in non-occluded areas. We exclude the occluded regions for now since few of the algorithms in this study explicitly model occlusions, and most perform quite poorly in these regions. As algorithms get better at matching occluded regions [65], however, we will likely focus more on the total matching error B.

The other two important measures are $B_{\overline{\mathcal{T}}}$ and $B_{\mathcal{D}}$, the percentage of bad pixels in textureless areas and in areas near depth discontinuities. These measures provide important information about the performance of algorithms in two critical problem areas. The parameter names for these three measures are *bad_pixels_nonocc*, *bad_pixels_textureless*, and *bad_pixels_discont*, and they appear in most of the plots below. We prefer the percentage of bad pixels over RMS disparity errors since this gives a better indication of the overall performance of an algorithm. For example, an algorithm is performing reasonably well if $B_{\overline{\mathcal{O}}} < 10\%$. The RMS error figure, on the other hand, is contaminated by the (potentially large) disparity errors in those poorly matched 10% of the image. RMS errors become important once the percentage of bad pixels drops to a few percent and the quality of a sub-pixel fit needs to be evaluated (see Section 6.4).

Note that the algorithms always take exactly two images as input, even when more are available. For example, with our 9-frame sequences, we use the third and seventh frame as input pair. (The other frames are used to measure the prediction error.)

6.1. Matching cost

We start by comparing different matching costs, including absolute differences (AD), squared differences (SD), truncated versions of both, and Birchfield and Tomasi's [12] sampling-insensitive dissimilarity measure (BT).

An interesting issue when trying to assess a single algorithm component is how to fix the parameters that control the other components. We usually choose good values based on experiments that assess the other algorithm components. (The inherent boot-strapping problem disappears after a few rounds of experiments.) Since the best settings for many parameters vary depending on the input image pair, we often have to compromise and select a value that works reasonably well for several images.

Experiment 1: In this experiment we compare the matching costs AD, SD, AD+BT, and SD+BT using a local algorithm. We aggregate with a 9×9 window, followed by winner-take-all optimization (i.e., we use the standard SAD and SSD algorithms). We do not compute sub-pixel estimates. Truncation values used are 1, 2, 5, 10, 20, 50, and ∞ (no truncation); these values are squared when truncating SD.

Results: Figure 8 shows plots of the three evaluation measures $B_{\overline{\mathcal{O}}}$, $B_{\overline{\mathcal{T}}}$, and $B_{\mathcal{D}}$ for each of the four matching costs as a function of truncation values, for the Tsukuba, Sawtooth, and Venus images. Overall, there is little difference between AD and SD. Truncation matters mostly for points near discontinuities. The reason is that for windows containing mixed populations (both foreground and background points), truncating the matching cost limits the influence of wrong matches. Good truncation values range from 5 to 50, typically around 20. Once the truncation values drop below the noise level (e.g., 2 and 1), the errors become very large. Using Birchfield-Tomasi (BT) helps for these small truncation values, but yields little improvement for good truncation

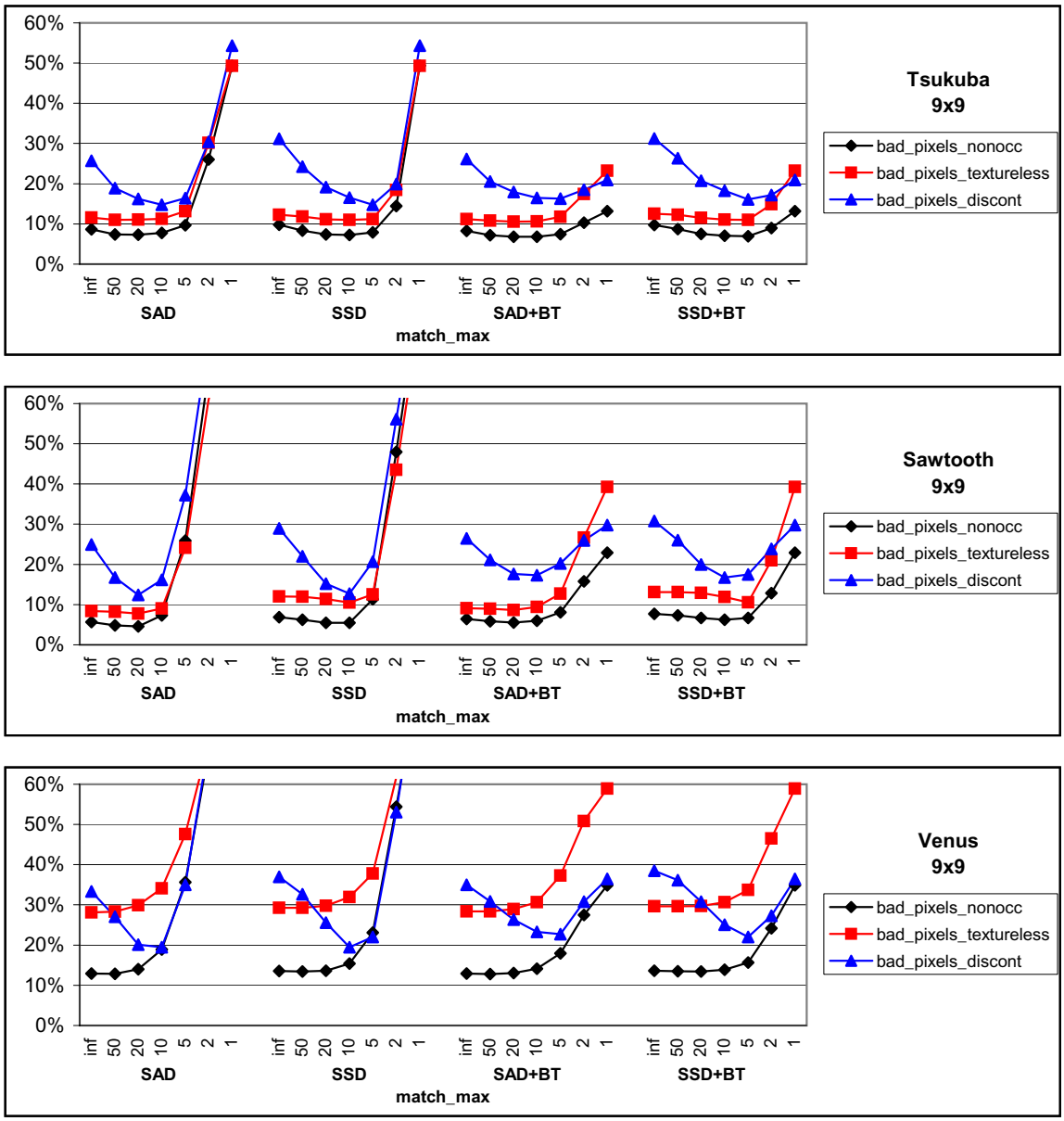


Figure 8: *Experiment 1. Performance of different matching costs aggregated with a* 9×9 *window as a function of truncation values* match_max *for three different image pairs. Intermediate truncation values (5–20) yield the best results. Birchfield-Tomasi (BT) helps when truncation values are low.*

values. The results are consistent across all data sets; however, the best truncation value varies. We have also tried a window size of 21, with similar results.
Conclusion: Truncation can help for AD and SD, but the best truncation value depends on the images' signal-to-noise-ratio (SNR), since truncation should happen right above the noise level present (see also the discussion in [97]).

Experiment 2: This experiment is identical to the previous one, except that we also use a 9×9 min-filter (in effect, we aggregate with shiftable windows).
Results: Figure 9 shows the plots for this experiment, again for Tsukuba, Sawtooth, and Venus images. As before, there are negligible differences between AD and SD. Now, however, the non-truncated versions perform consistently the best. In particular, for points near discontinuities we get the lowest errors overall, but also the total errors are comparable to the best settings of truncation in Experiment 1. BT helps bring down larger errors, but as before, does not significantly decrease the best (non-truncated) errors. We again also tried a window size of 21 with similar results.
Conclusion: The problem of selecting the best truncation value can be avoided by instead using a shiftable window (min-filter). This is an interesting result, as both robust matching costs (trunctated functions) and shiftable windows have been proposed to deal with outliers in windows that straddle object boundaries. The above experiments suggest that *avoiding* outliers by shifting the window is preferable to limiting their influence using truncated cost functions.

Experiment 3: We now assess how matching costs affect global algorithms, using dynamic programming (DP), scanline optimization (SO), and graph cuts (GC) as optimization techniques. A problem with global techniques that minimize a weighted sum of data and smoothness terms (Equation (3)) is that the range of matching cost values affects the optimal value for λ, i.e., the relative weight of the smoothness term. For example, squared differences require much higher values for λ than absolute differences. Similarly, truncated difference functions result in lower matching costs and require lower values for λ. Thus, in trying to isolate the effect of the matching costs, we are faced with the problem of how to choose λ. The cleanest solution to this dilemma would perhaps be to find a (different) optimal λ independently for each matching cost under consideration, and then to report which matching cost gives the overall best results. The optimal λ, however, would not only differ across matching costs, but also across different images. Since in a practical matcher we need to choose a constant λ, we have done the same in this experiment. We use $\lambda = 20$ (guided by the results discussed in Section 6.3 below) and restrict the matching costs to absolute differences (AD), truncated by varying amounts. For the DP algorithm we use a fixed occlusion cost of 20.

Results: Figure 10 shows plots of the bad pixel percentages $B_{\overline{\mathcal{O}}}$, $B_{\overline{\mathcal{T}}}$, and $B_{\mathcal{D}}$ as a function of truncation values for Tsukuba, Sawtooth, and Venus images. Each plot has six curves, corresponding to DP, DP+BT, SO, SO+BT, GC, GC+BT. It can be seen that the truncation value affects the performance. As with the local algorithms, if the truncation value is too small (in the noise range), the errors get very large. Intermediate truncation values of 50–5, depending on algorithm and image pair, however, can sometimes improve the performance. The effect of Birchfield-Tomasi is mixed; as with the local algorithms in Experiments 1 and 2, it limits the errors if the truncation values are too small. It can be seen that BT is most beneficial for the SO algorithm, however, this is due to the fact that SO really requires a higher value of λ to work well (see Experiment 5), in which case the positive effect of BT is less pronounced.
Conclusion: Using robust (truncated) matching costs can slightly improve the performance of global algorithms. The best truncation value, however, varies with each image pair. Setting this parameter automatically based on an estimate of the image SNR may be possible and is a topic for further research. Birchfield and Tomasi's matching measure can improve results slightly. Intuitively, truncation should not be necessary for global algorithms that operate on unaggregated matching costs, since the problem of outliers in a window does not exist. An important problem for global algorithms, however, is to find the correct balance between data and smoothness terms (see Experiment 5 below). Truncation can be useful in this context since it limits the range of possible cost values.

6.2. Aggregation

We now turn to comparing different aggregation methods used by local methods. While global methods typically operate on raw (unaggregated) costs, aggregation can be useful for those methods as well, for example to provide starting values for iterative algorithms, or a set of high-confidence matches or *ground control points* (GCPs) [18] used to restrict the search of dynamic-programming methods.

In this section we examine aggregation with square windows, shiftable windows (min-filter), binomial filters, regular diffusion, and membrane diffusion [97]. Results for Bayesian diffusion, which combines aggregation and optimization, can be found in Section 7.

Experiment 4: In this experiment we use (non-truncated) absolute differences as matching cost and perform a winner-take-all optimization after the aggregation step (no sub-pixel estimation). We compare the following aggregation methods:

1. square windows with window sizes 3, 5, 7, . . . , 29;
2. shiftable square windows (min-filter) with window sizes 3, 5, 7, . . . 29;

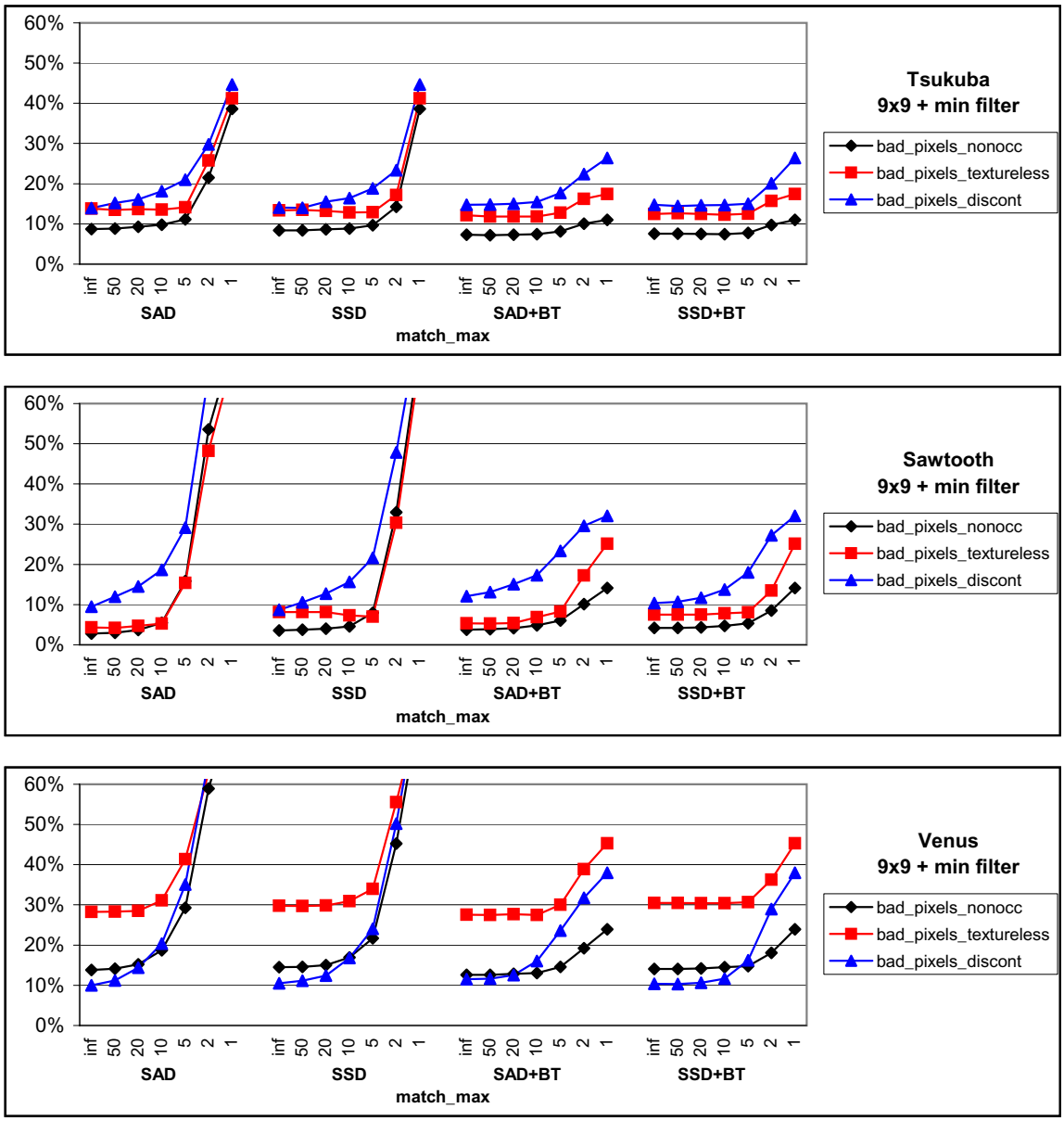


Figure 9: *Experiment 2. Performance of different matching costs aggregated with a* 9×9 *shiftable window (min-filter) as a function of truncation values* match_max *for three different image pairs. Large truncation values (no truncation) work best when using shiftable windows.*

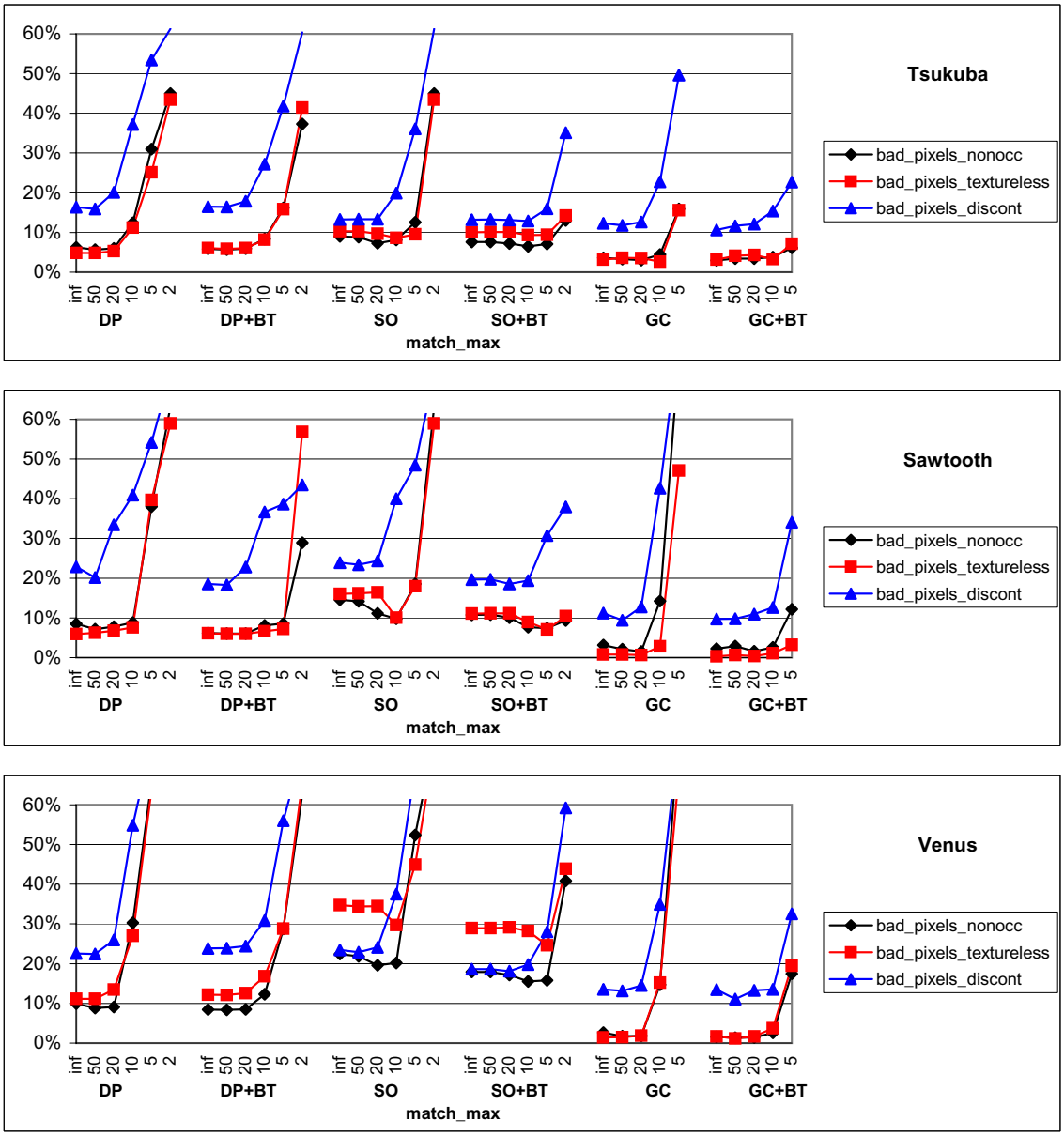


Figure 10: *Experiment 3. Performance of different matching costs for global algorithms as a function of truncation values* match_max *for three different image pairs. Intermediate truncation values (*~ 20*) can sometimes improve the performance.*

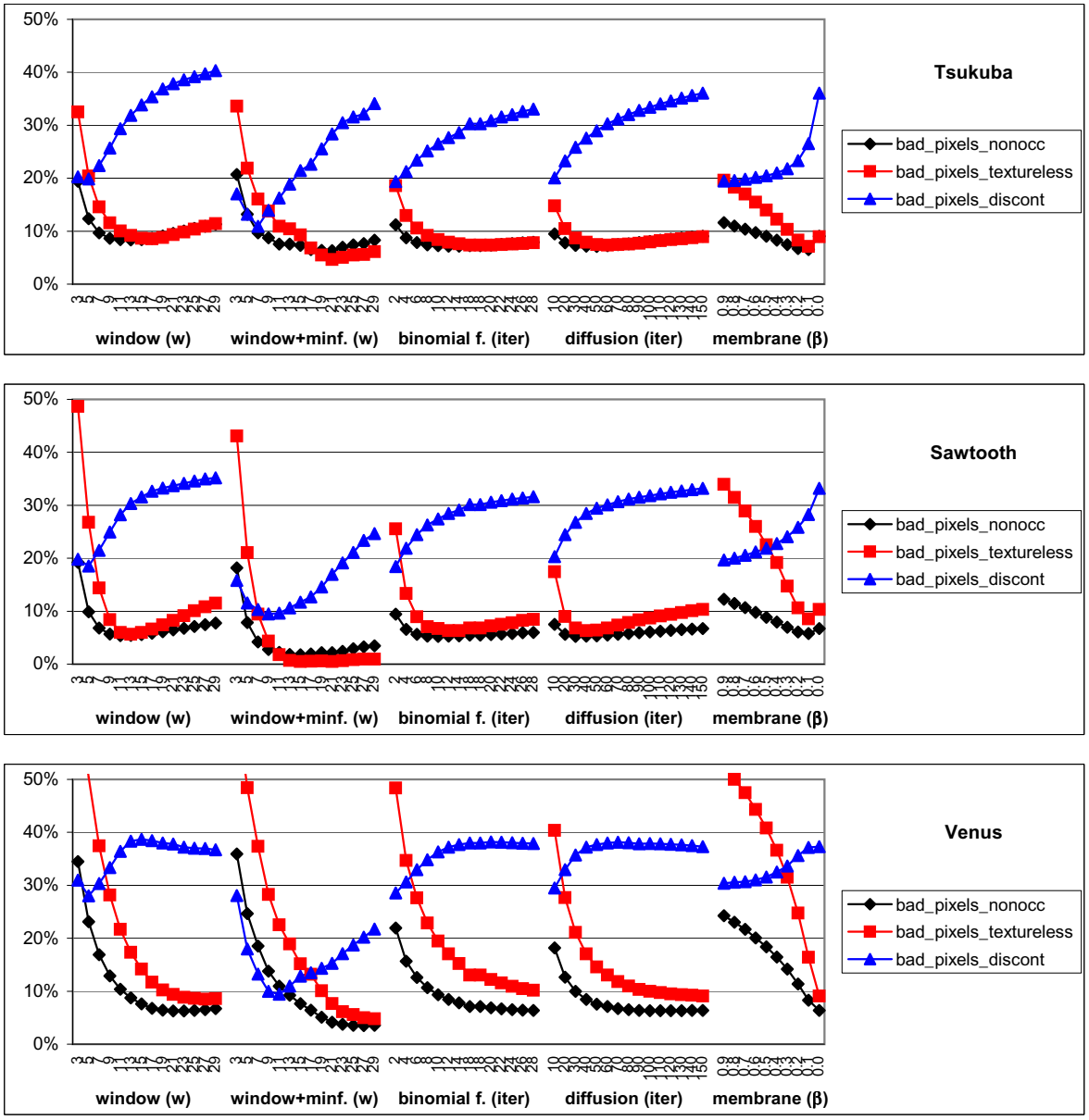


Figure 11: *Experiment 4. Performance of different aggregation methods as a function of spatial extent (window size, number of iterations, and diffusion β). Larger window extents do worse near discontinuities, but better in textureless areas, which tend to dominate the overall statistics. Near discontinuities, shiftable windows have the best performance.*

3. iterated binomial (1-4-6-4-1) filter, for 2, 4, 6, ..., 28 iterations;

4. regular diffusion [97] for 10, 20, 30, . . . , 150 iterations;

5. membrane diffusion [97] for 150 iterations and β = 0.9, 0.8. 0.7, . . . , 0.0.

Note that for each method we are varying the parameter that controls the spatial extent of the aggregation (i.e., the equivalent of window size). In particular, for the binomial filter and regular diffusion, this amounts to changing the number of iterations. The membrane model, however, converges after sufficiently many iterations, and the spatial extent of the aggregation is controlled by the parameter β, the weight of the original cost values in the diffusion equation [97].

Results: Figure 11 shows plots of $B_{\overline{\mathcal{O}}}$, $B_{\overline{\mathcal{T}}}$, and $B_{\mathcal{D}}$ as a function of spatial extent of aggregation for Tsukuba, Sawtooth, and Venus images. Each plot has five curves, corresponding to the five aggregation methods listed above. The most striking feature of these curves is the opposite trends of errors in textureless areas ($B_{\overline{\mathcal{T}}}$) and at points near discontinuities ($B_{\mathcal{D}}$). Not surprisingly, more aggregation (larger window sizes or higher number of iterations) clearly helps to recover textureless areas (note especially the Venus images, which contain large untextured regions). At the same time, too much aggregation causes errors near object boundaries (depth discontinuities). The overall error in non-occluded regions, $B_{\overline{\mathcal{O}}}$, exhibits a mixture of both trends. Depending on the image, the best performance is usually achieved at an intermediate amount of aggregation. Among the five aggregation methods, shiftable windows clearly perform best, most notably in discontinuity regions, but also overall. The other four methods (square windows, binomial filter, regular diffusion, and membrane model) perform very similarly, except for differences in the shape of the curves, which are due to our (somewhat arbitrary) definition of spatial extent for each method. Note however that even for shiftable windows, the optimal window size for recovering discontinuities is small, while much larger windows are necessary in untextured regions.

Discussion: This experiment exposes some of the fundamental limitations of local methods. While large windows are needed to avoid wrong matches in regions with little texture, window-based stereo methods perform poorly near object boundaries (i.e., depth discontinuities). The reason is that such methods implicitly assume that all points within a window have similar disparities. If a window straddles a depth boundary, some points in the window match at the foreground disparity, while others match at the background disparity. The (aggregated) cost function at a point near a depth discontinuity is thus bimodal in the d direction, and stronger of the two modes will be selected as the winning disparity. Which one of the two modes will win? This depends on the amount of (horizontal) texture present in the two regions.

Consider first a purely horizontal depth discontinuity (top edge of the foreground square in Figure 12). Whichever of the two regions has more horizontal texture will create a stronger mode, and the computed disparities will thus "bleed" into the less-textured region. For non-horizontal depth boundaries, however, the most prominent horizontal texture is usually the object boundary itself, since different objects typically have different colors and intensities. Since the object boundary is at the foreground disparity, a strong preference for the foreground disparity at points near the boundary is created, even if the background is textured. This is the explanation for the well-known "foreground fattening" effect exhibited by window-based algorithms. This can be seen at the right edge of the foreground in Figure 12; the left edge is an occluded area, which can't be recovered in any case.

Adaptive window methods have been developed to combat this problem. The simplest variant, shiftable windows (min-filters) can be effective, as is shown in the above experiment. Shiftable windows can recover object boundaries quite accurately if both foreground and background regions are textured, and as long as the window fits as a whole within the foreground object. The size of the min-filter should be chosen to match the window size. As is the case with all local methods, however, shiftable windows fail in textureless areas.

Conclusion: Local algorithms that aggregate support can perform well, especially in textured (even slanted) regions. Shiftable windows perform best, in particular near depth discontinuities. Large amounts of aggregation are necessary in textureless regions.

6.3. Optimization

In this section we compare the four global optimization techniques we implemented: dynamic programming (DP), scanline optimization (SO), graph cuts (GC), and simulated annealing (SA).

Experiment 5: In this experiment we investigate the role of *opt_smoothness*, the smoothness weight λ in Equation (3). We compare the performance of DP, SO, GC, and SA for $\lambda = 5, 10, 20, 50, 100, 200, 500$, and 1000. We use unaggregated absolute differences as the matching cost (squared differences would require much higher values for λ), and no sub-pixel estimation. The number of iterations for simulated annealing (SA) is 500.

Results: Figure 13 shows plots of $B_{\overline{\mathcal{O}}}$, $B_{\overline{\mathcal{T}}}$, and $B_{\mathcal{D}}$ as a function of λ for Tsukuba, Venus, and Map images. (To show more varied results, we use the Map images instead of Sawtooth in this experiment.) Since DP has an extra parameter, the occlusion cost, we include three runs, for

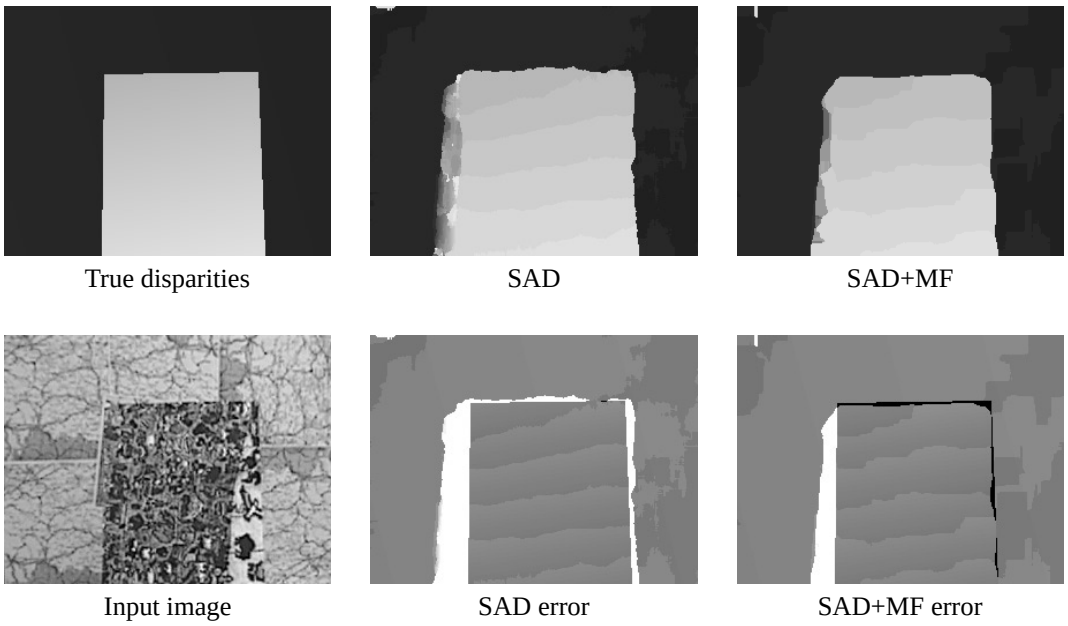


Figure 12: *Illustration of the "foreground fattening" effect, using the Map image pair and a* 21×21 *SAD algorithm, with and without a min-filter. The error maps encode the signed disparity error, using gray for 0, light for positive errors, and dark for negative errors. Note that without the min-filter (middle column) the foreground region grows across the vertical depth discontinuity towards the right. With the min-filter (right column), the object boundaries are recovered fairly well.*

opt_occlusion_cost = 20, 50, and 80. Using as before $B_{\overline{\mathcal{O}}}$ (*bad_pixels_nonocc*) as our measure of overall performance, it can be seen that the graph-cut method (GC) consistently performs best, while the other three (DP, SO, and SA) perform slightly worse, with no clear ranking among them. GC also performs best in textureless areas and near discontinuities. The best performance for each algorithm, however, requires different values for λ depending on the image pair. For example, the Map images, which are well textured and only contain two planar regions, require high values (around 500), while the Tsukuba images, which contain many objects at different depths, require smaller values (20–200, also depending on the algorithm). The occlusion cost parameter for the DP algorithm, while not changing the performance dramatically, also affects the optimal value for λ. Although GC is the clear winner here, it is also the slowest algorithm: DP and SO, which operate on each scanline independently, typically run in less than 2 seconds, while GC and SA require 10–30 minutes.

Conclusion: The graph-cut method consistently outperforms the other optimization methods, although at the cost of much higher running times. GC is clearly superior to simulated annealing, which is consistent with other published results [23, 116]. When comparing GC and scanline methods (DP and SO), however, it should be noted that the latter solve a different (easier) optimization problem, since vertical smoothness terms are ignored. While this enables the use of highly efficient dynamic programming techniques, it negatively affects the performance, as exhibited in the characteristic "streaking" in the disparity maps (see Figures 17 and 18 below). Several authors have proposed methods for increasing inter-scanline consistency in dynamic-programming approaches, e.g., [9, 31, 13]. We plan to investigate this area in future work.

Experiment 6: We now focus on the graph-cut optimization method to see whether the results can be improved. We try both Birchfield-Tomasi matching costs and a smoothness cost that depends on the intensity gradients

Results: Figure 14 shows the usual set of performance measures $B_{\overline{\mathcal{O}}}$, $B_{\overline{\mathcal{T}}}$, and $B_{\mathcal{D}}$ for four different experiments for Tsukuba, Sawtooth, Venus, and Map images. We use a smoothness weight of $\lambda = 20$, except for the Map images, where $\lambda = 50$. The matching cost are (non-truncated) absolute differences. The parameters for the gradient-dependent smoothness costs are *opt_grad_thresh* = 8 (same in all experiments), and *opt_grad_penalty* = 1, 2, or 4 (denoted p1, p2, and p4 in the plots). Recall that the smoothness cost is multiplied by *opt_grad_penalty* if the intensity gradient is below *opt_grad_thresh* to encourage disparity jumps to coincide with intensity edges. Each plot in Figure 14 shows 4 runs: p1, p1+BT, p2+BT, and p4+BT. In the first run, the penalty is 1, i.e., the gradient dependency is turned off. This gives the same results as in Experiment 5. In the second run, we

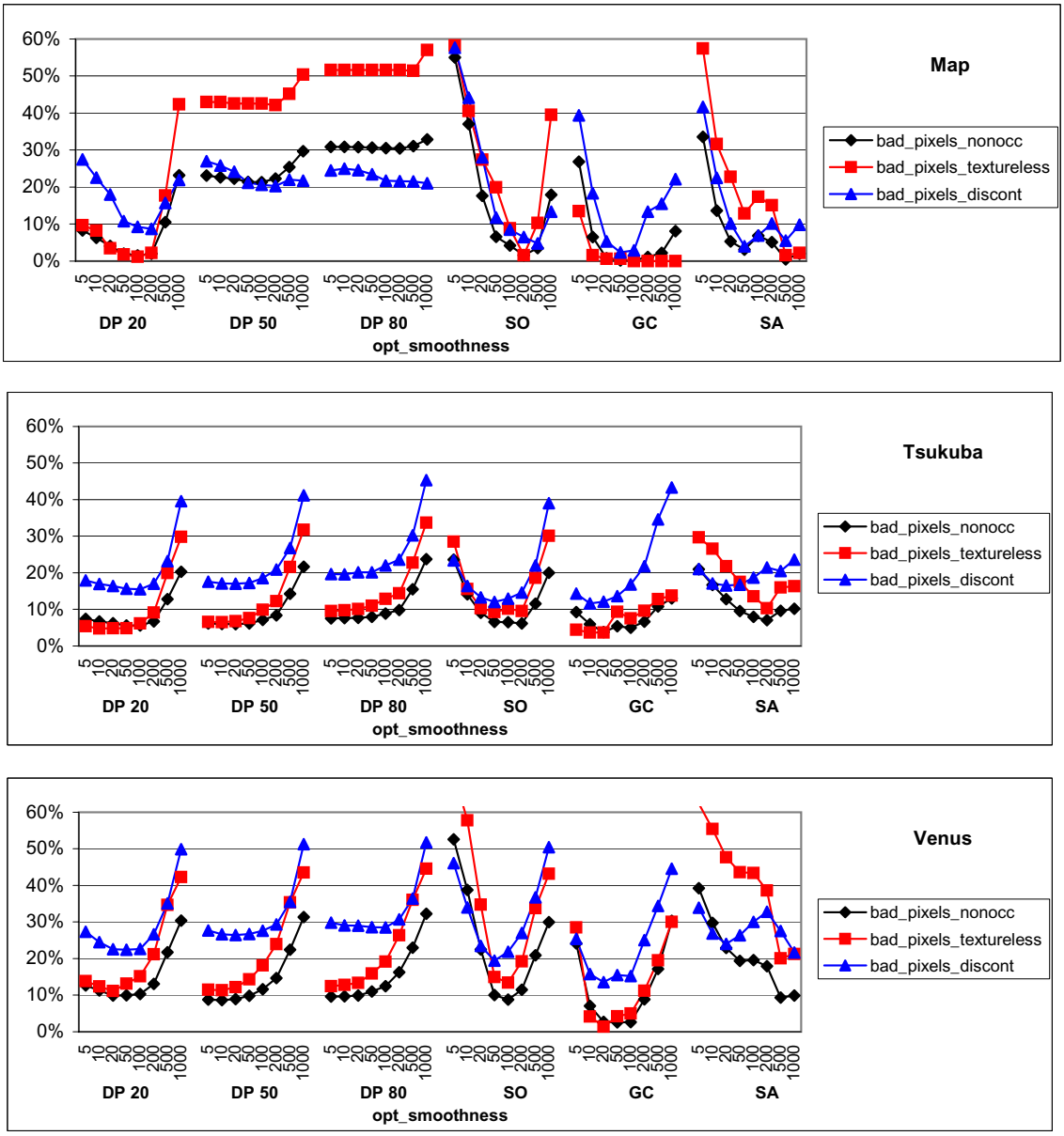


Figure 13: *Experiment 5. Performance of global optimization techniques as a function of the smoothness weight* λ (opt_smoothness) *for Map, Tsukuba, and Venus images. Note that each image pair requires a different value of* λ *for optimal performance.*

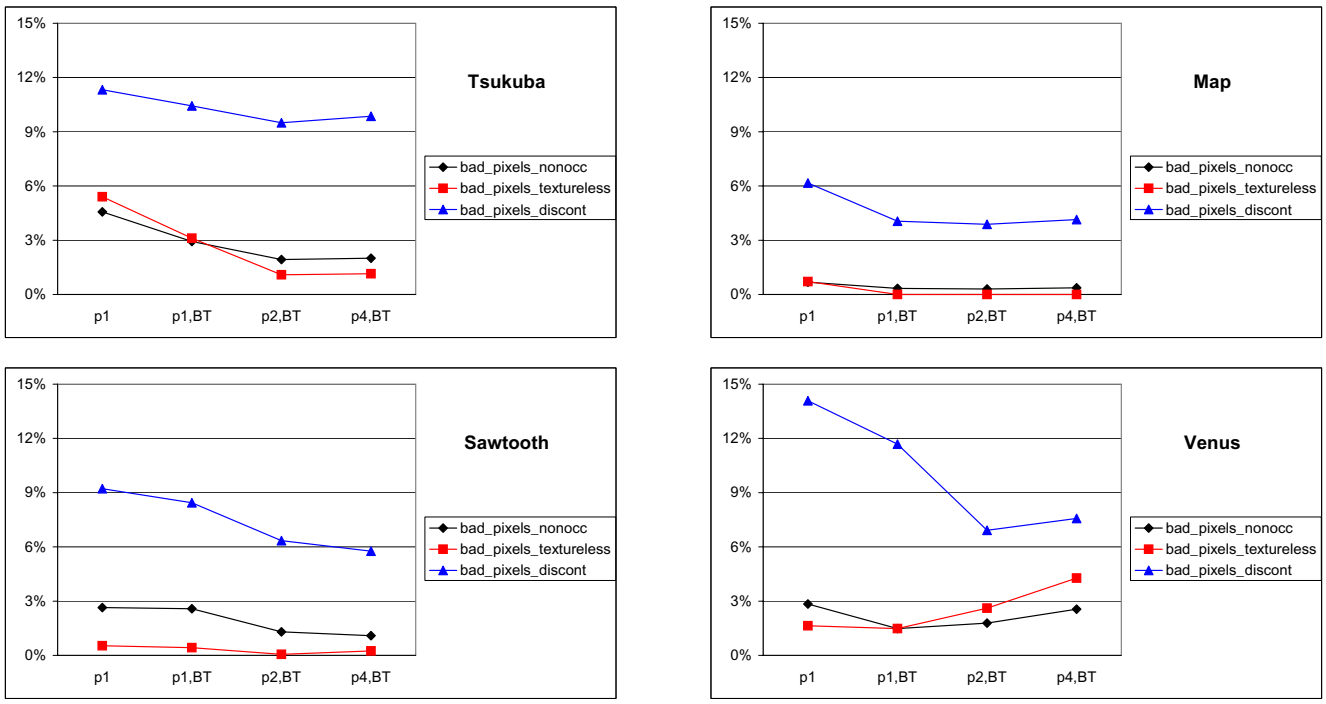


Figure 14: *Experiment 6. Performance of the graph-cut optimization technique with different gradient-dependent smoothness penalties (p1, p2, p4) and with and without Birchfield-Tomasi (BT).*

add Birchfield-Tomasi, still without a penalty. We then add a penalty of 2 and 4 in the last two runs. It can be seen that the low-gradient penalty clearly helps recovering the discontinuities, and also in the other regions. Which of the two penalties works better depends on the image pair. Birchfield-Tomasi also yields a slight improvement. We have also tried other values for the threshold, with mixed results. In future work we plan to replace the simple gradient threshold with an edge detector, which should improve edge localization. The issue of selecting the right penalty factor is closely related to selecting the right value for λ, since it affects the overall relationship between the data term and the smoothness term. This also deserves more investigation.

Conclusion: Both Birchfield-Tomasi's matching cost and the gradient-based smoothness cost improve the performance of the graph-cut algorithm. Choosing the right parameters (threshold and penalty) remains difficult and image-specific.

We have performed these experiments for scanline-based optimization methods (DP and SO) as well, with similar results. Gradient-based penalties usually increase performance, in particular for the SO method. Birchfield-Tomasi always seems to increase overall performance, but it sometimes decreases performance in textureless areas. As before, the algorithms are highly sensitive to the weight of the smoothness term λ and the penalty factor.

6.4. Sub-pixel estimation

Experiment 7: To evaluate the performance of the sub-pixel refinement stage, and also to evaluate the influence of the matching criteria and disparity sampling, we cropped a small planar region from one of our image sequences (Figure 15a, second column of images). The image itself is a page of newsprint mounted on cardboard, with high-frequency text and a few low-frequency white and dark regions. (These textureless regions were excluded from the statistics we gathered.) The disparities in this region are in the order of 0.8–3.8 pixels and are slanted both vertically and horizontally.

Results: We first run a simple 9×9 SSD window (Figure 15b). One can clearly see the discrete disparity levels computed. The disparity error map (second column of images) shows the staircase error, and the histogram of disparities (third column) also shows the discretization. If we apply the sub-pixel parabolic fit to refine the disparities, the disparity map becomes smoother (note the drop in RMS error in Figure 15c), but still shows some soft staircasing, which is visible in the disparity error map and histogram as well. These results agree with those reported by Shimizu and Okutomi [105].

In Figure 15d, we investigate whether using the Birchfield-Tomasi sampling-invariant measure [12] improves or degrades this behavior. For integral sampling, their idea does help slightly, as can be seen by the reduced

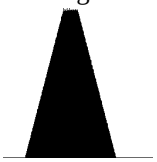
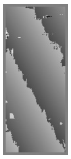
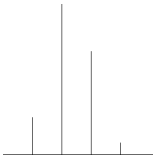
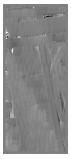
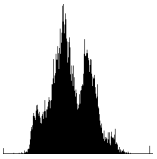
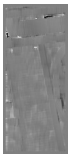
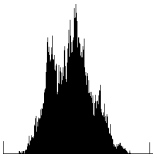
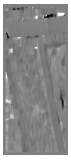
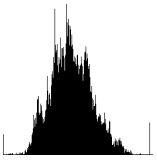
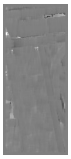
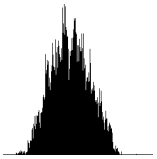
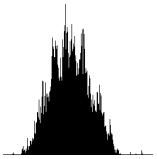
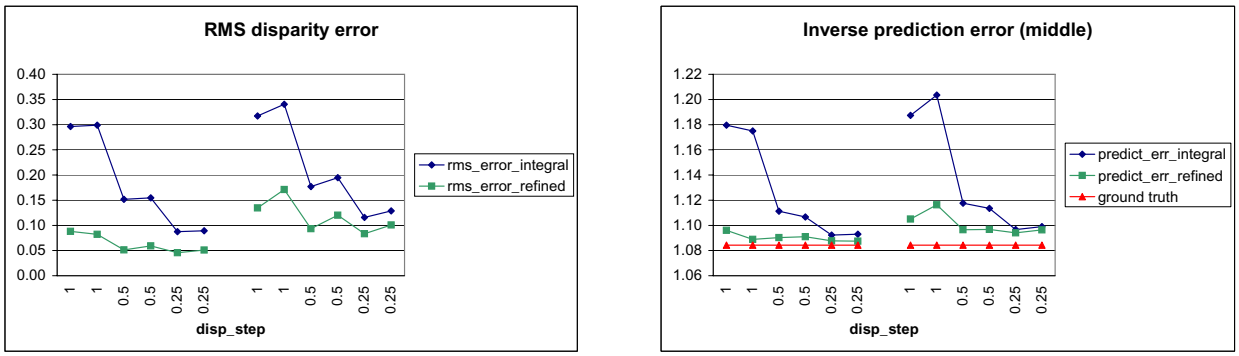
	disp. step	refine subpix	Birchf.-Tomasi	preproc. blur	RMS disp. error	disp. map	disp. error	disp. histogram
(a)	ground truth				0			
(b)	1	no	no	no	0.296			
(c)	1	yes	no	no	0.088			
(d)	1	yes	yes	no	0.082			
(e)	1	yes	no	yes	0.135			
(f)	$\frac{1}{2}$	yes	no	no	0.051			
(g)	$\frac{1}{4}$	no	no	no	0.087			
(h)	$\frac{1}{4}$	yes	no	no	0.046			

Figure 15: *RMS disparity errors for cropped image sequence (planar region of newspaper). The reference image is shown in row (a) in the "disp. error" column. The columns indicate the disparity step, the sub-pixel refinement option, Birchfield-Tomasi's sampling-insensitive matching option, the optional initial blur, and the RMS disparity error from ground truth. The first image column shows the computed disparity map, the second shows the signed disparity error, and the last column shows a histogram of computed disparities.*



(a) (b)

Figure 16: *Plots of RMS disparity error and inverse prediction errors as a function of* disp_step *and* match_interval. *The even data points are with sampling-insensitive matching* match_interval *turned on. The second set of plots in each figure is with* preproc_blur *enabled (1 blur iteration).*

RMS value and the smoother histogram in Figure 15d. In all other instances, it leads to poorer performance (see Figure 16a, where the sampling-invariant results are the even data points).

In Figure 15e, we investigate whether lightly blurring the input images with a $(1/4, 1/2, 1/4)$ kernel helps sub-pixel refinement, because the first order Taylor series expansion of the imaging function becomes more valid. Blurring does indeed slightly reduce the staircasing effect (compare Figure 15e to Figure 15c), but the overall (RMS) performance degrades, probably because of loss of high-frequency detail.

We also tried $1/2$ and $1/4$ pixel disparity sampling at the initial matching stages, with and without later sub-pixel refinement. Sub-pixel refinement always helps to reduce the RMS disparity error, although it has negligible effect on the inverse prediction error (Figure 16b). From these prediction error plots, and also from visual inspection of the inverse warped (stabilized) image sequence, it appears that using sub-pixel refinement after *any* original matching scheme is sufficient to reduce the prediction error (and the appearance of "jitter" or "shearing") to negligible levels. This is despite the fact that the theoretical justification for sub-pixel refinement is based on a quadratic fit to an adequately sampled quadratic energy function. At the moment, for global methods, we rely on the per-pixel costs that go into the optimization to do the sub-pixel disparity estimation. Alternative approaches, such as using local plane fits [5, 14, 117] could also be used to get sub-pixel precision.

Conclusions: To eliminate "staircasing" in the computed disparity map and to also eliminate the appearance of "shearing" in reprojected sequences, it is necessary to initially evaluate the matches at a fractional disparity ($1/2$ pixel steps appear to be adequate). This should be followed by finding the minima of local quadratic fits applied to the computed matching costs.

7. Overall comparison

We close our experimental investigation with an overall comparison of 20 different stereo methods, including 5 algorithms implemented by us and 15 algorithms implemented by their authors, who have kindly sent us their results. We evaluate all algorithms using our familiar set of Tsukuba, Sawtooth, Venus, and Map images. All algorithms are run with constant parameters over all four images. We do not perform sub-pixel estimation in this comparison.

Among the algorithms in our implementation framework, we have selected the following five:

(1) SSD – 21×21 shiftable window SSD,

(2) DP – dynamic programming,

(3) SO – scanline optimization,

(4) GC – graph-cut optimization, and

(5) Bay – Bayesian diffusion.

We chose shiftable-window SSD as best-performing representative of all local (aggregation-based) algorithms. We are not including simulated annealing here, since GC solves the same optimization problem better and more efficiently. For each of the five algorithms, we have chosen fixed parameters that yield reasonably good performance over a variety of input images (see Table 4).

We compare the results of our implementation with results provided by the authors of the following algorithms:

(6) Max-flow / min-cut algorithm, Roy and Cox [92, 91] (one of the first methods to formulate matching as a graph flow problem);

(7) Pixel-to-pixel stereo, Birchfield and Tomasi [13] (scanline algorithm using gradient-modulated costs, followed by vertical disparity propagation into unreliable areas);

	SSD	DP	SO	GC	Bay
Matching cost					
match_fn	SD	AD	AD	AD	AD
Truncation	no	no	no	no	no
Birchfield / Tomasi	no	yes	yes	yes	no
Aggregation					
aggr_window_size	21	—	—	—	—
aggr_minfilter	21	—	—	—	—
aggr_iter	1	—	—	—	1000
diff_mu	—	—	—	—	0.5
diff_sigmaP	—	—	—	—	0.4
diff_epsP	—	—	—	—	0.01
diff_scale_cost	—	—	—	—	0.01
Optimization					
opt_fn	WTA	DP	SO	GC	Bayesian
opt_smoothness (λ)	—	20	50	20	—
opt_occlusion_cost	—	20	—	—	—
opt_grad_thresh	—	8	8	8	—
opt_grad_penalty	—	4	2	2	—

Table 4: *Parameters for the five algorithms implemented by us.*

(8) Multiway cut, Birchfield and Tomasi [14] (alternate segmentation and finding affine parameters for each segment using graph cuts);

(9) Cooperative algorithm, Zitnick and Kanade [132] (a new variant of Marr and Poggio's algorithm [73]);

(10) Graph cuts, Boykov *et al.* [23] (same as our GC method, but a much faster implementation);

(11) Graph cuts with occlusions, Kolmogorov and Zabih [65] (an extension of the graph-cut framework that explicitly models occlusions);

(12) Compact windows, Veksler [124] (an adaptive window technique allowing non-rectangular windows);

(13) Genetic algorithm, Gong and Yang [49] (a global optimization technique operating on quadtrees);

(14) Realtime method, Hirschmüller [53] (9×9 SAD with shiftable windows, followed by consistency checking and interpolation of uncertain areas);

(15) Stochastic diffusion, Lee *et al.* [68] (a variant of Bayesian diffusion [97]);

(16) Fast correlation algorithm, Mühlmann *et al.* [79] (an efficient implementation of correlation-based matching with consistency and uniqueness validation);

(17) Discontinuity-preserving regularization, Shao [104] (a multi-view technique for virtual view generation);

(18) Maximum-surface technique, Sun [108] (a fast stereo algorithm using rectangular subregions);

(19) Belief propagation, Sun *et al.* [109] (a MRF formulation using Bayesian belief propagation);

(20) Layered stereo, Lin and Tomasi [69] (a preliminary version of an extension of the multiway-cut method [14]).

Some of these algorithms do not compute disparity estimates everywhere, in particular those that explicitly model occlusions (3 and 11), but also (16) which leaves low-confidence regions unmatched. In these cases we fill unmatched areas as described for the DP method in Section 4.3.

Table 5 summarizes the results for all algorithms. As in the previous section, we report $B_{\overline{\mathcal{O}}}$ (*bad_pixels_nonocc*) as a measure of overall performance, as well as $B_{\overline{\mathcal{T}}}$ (*bad_pixels_textureless*), and $B_{\mathcal{D}}$ (*bad_pixels_discont*). We do not report $B_{\overline{\mathcal{T}}}$ for the Map images since the images are textured almost everywhere. The algorithms are listed roughly in order of overall performance.

The disparity maps for Tsukuba and Venus images are shown in Figures 17 and 18. The full set of disparity maps, as well as signed and binary error maps are available on our web site at `www.middlebury.edu/stereo`.

Looking at these results, we can draw several conclusions about the overall performance of the algorithms. First, global optimization methods based on 2-D MRFs generally perform the best in all regions of the image (overall, textureless, and discontinuities). Most of these techniques are based on graph-cut optimization (4, 8, 10, 11, 20), but belief propagation (19) also does well. Approaches that explicitly model planar surfaces (8, 20) are especially good with piecewise planar scenes such as Sawtooth and Venus.

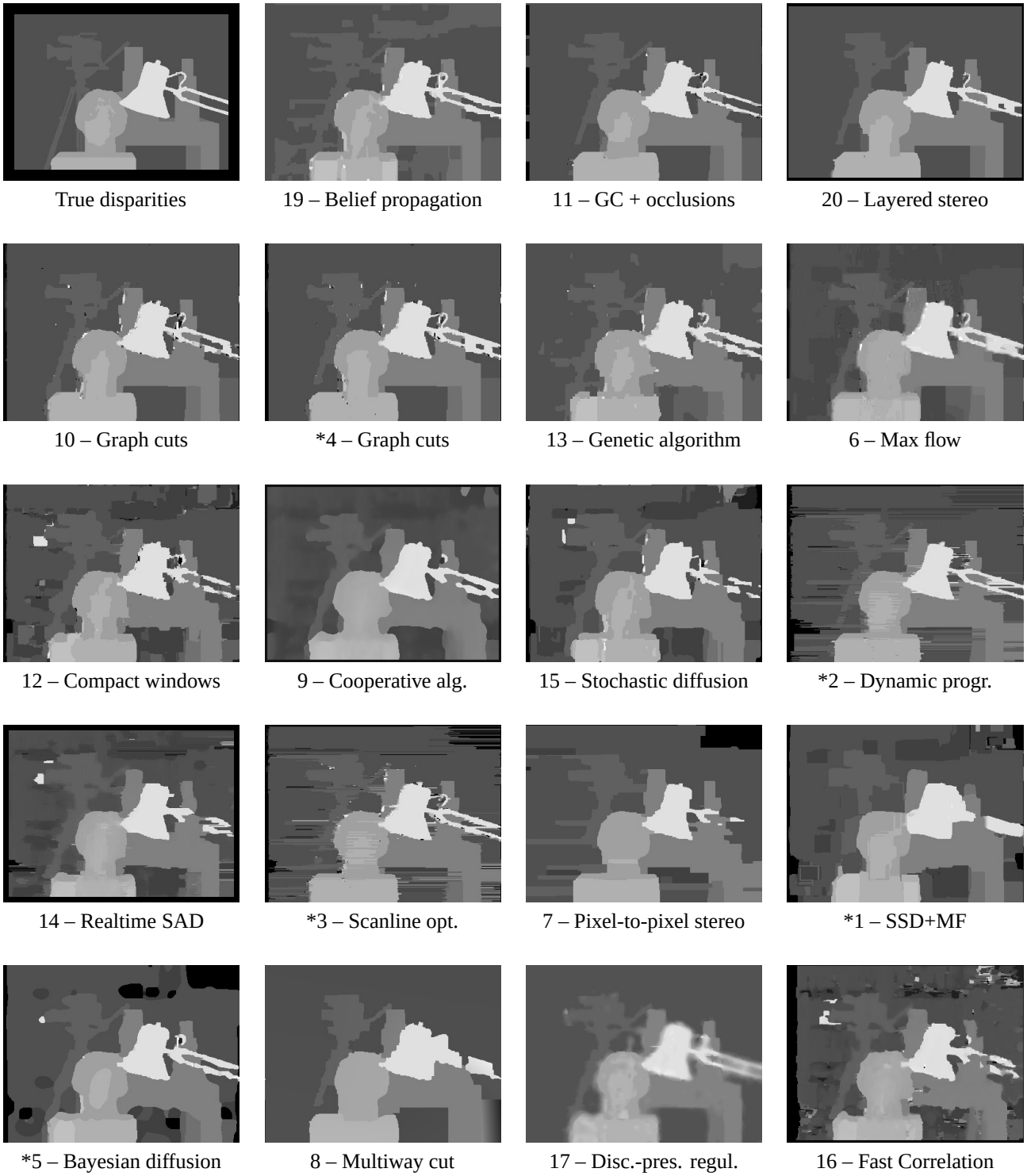


Figure 17: *Comparative results on the Tsukuba images. The results are shown in decreasing order of overall performance ($B_{\overline{O}}$). Algorithms implemented by us are marked with a star.*

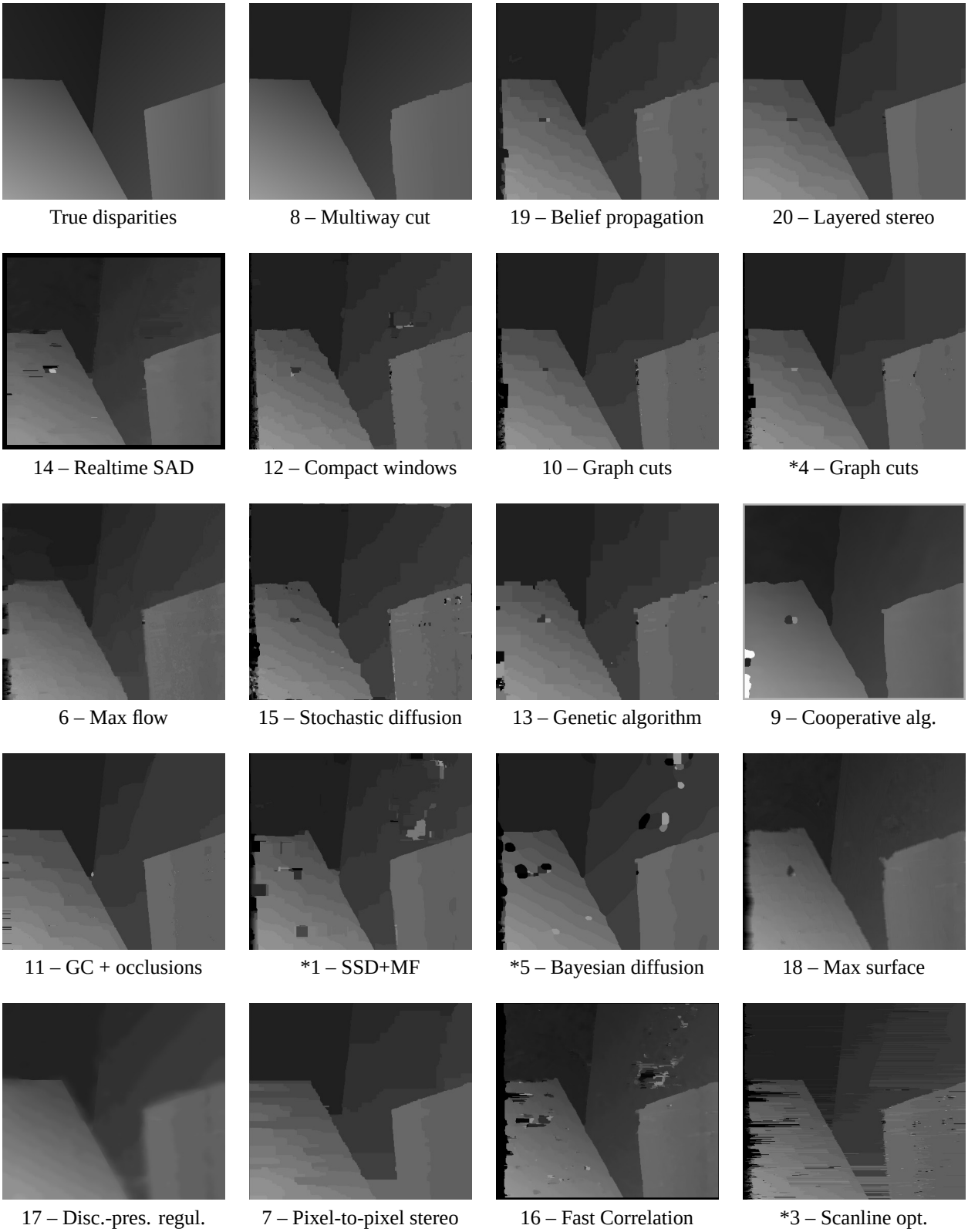


Figure 18: *Comparative results on the Venus images. The results are shown in decreasing order of overall performance ($B_{\overline{O}}$). Algorithms implemented by us are marked with a star.*

	Tsukuba			Sawtooth			Venus			Map	
	$B_{\overline{\mathcal{O}}}$	$B_{\overline{\mathcal{T}}}$	$B_{\mathcal{D}}$	$B_{\overline{\mathcal{O}}}$	$B_{\overline{\mathcal{T}}}$	$B_{\mathcal{D}}$	$B_{\overline{\mathcal{O}}}$	$B_{\overline{\mathcal{T}}}$	$B_{\mathcal{D}}$	$B_{\overline{\mathcal{O}}}$	$B_{\mathcal{D}}$
20 Layered	1.58 *3*	1.06 *4*	8.82 *3*	**0.34** *1*	**0.00** *1*	**3.35** *1*	1.52 *3*	2.96 *10*	2.62 *2*	0.37 *6*	5.24 *6*
*4 Graph cuts	1.94 *5*	1.09 *5*	9.49 *5*	1.30 *6*	0.06 *3*	6.34 *6*	1.79 *7*	2.61 *8*	6.91 *4*	0.31 *4*	3.88 *4*
19 Belief prop.	**1.15** *1*	**0.42** *1*	**6.31** *1*	0.98 *5*	0.30 *5*	4.83 *5*	1.00 *2*	0.76 *2*	9.13 *6*	0.84 *10*	5.27 *7*
11 GC+occl.	1.27 *2*	0.43 *2*	6.90 *2*	0.36 *2*	**0.00** *1*	3.65 *2*	2.79 *12*	5.39 *13*	**2.54** *1*	1.79 *13*	10.08 *12*
10 Graph cuts	1.86 *4*	1.00 *3*	9.35 *4*	0.42 *3*	0.14 *4*	3.76 *3*	1.69 *6*	2.30 *6*	5.40 *3*	2.39 *16*	9.35 *10*
8 Multiw. cut	8.08 *17*	6.53 *14*	25.33 *18*	0.61 *4*	0.46 *8*	4.60 *4*	**0.53** *1*	**0.31** *1*	8.06 *5*	0.26 *3*	3.27 *3*
12 Compact win.	3.36 *8*	3.54 *8*	12.91 *9*	1.61 *9*	0.45 *7*	7.87 *7*	1.67 *5*	2.18 *4*	13.24 *9*	0.33 *5*	3.94 *5*
14 Realtime	4.25 *12*	4.47 *12*	15.05 *13*	1.32 *7*	0.35 *6*	9.21 *8*	1.53 *4*	1.80 *3*	12.33 *7*	0.81 *9*	11.35 *15*
*5 Bay. diff.	6.49 *16*	11.62 *19*	12.29 *7*	1.45 *8*	0.72 *9*	9.29 *9*	4.00 *14*	7.21 *16*	18.39 *13*	**0.20** *1*	2.49 *2*
9 Cooperative	3.49 *9*	3.65 *9*	14.77 *11*	2.03 *10*	2.29 *14*	13.41 *13*	2.57 *11*	3.52 *11*	26.38 *17*	0.22 *2*	**2.37** *1*
*1 SSD+MF	5.23 *15*	3.80 *10*	24.66 *17*	2.21 *11*	0.72 *10*	13.97 *15*	3.74 *13*	6.82 *15*	12.94 *8*	0.66 *8*	9.35 *10*
15 Stoch. diff.	3.95 *10*	4.08 *11*	15.49 *15*	2.45 *14*	0.90 *11*	10.58 *10*	2.45 *9*	2.41 *7*	21.84 *15*	1.31 *12*	7.79 *9*
13 Genetic	2.96 *6*	2.66 *7*	14.97 *12*	2.21 *12*	2.76 *16*	13.96 *14*	2.49 *10*	2.89 *9*	23.04 *16*	1.04 *11*	10.91 *14*
7 Pix-to-pix	5.12 *14*	7.06 *17*	14.62 *10*	2.31 *13*	1.79 *12*	14.93 *17*	6.30 *17*	11.37 *18*	14.57 *10*	0.50 *7*	6.83 *8*
6 Max flow	2.98 *7*	2.00 *6*	15.10 *14*	3.47 *15*	3.00 *17*	14.19 *16*	2.16 *8*	2.24 *5*	21.73 *14*	3.13 *17*	15.98 *18*
*3 Scanl. opt.	5.08 *13*	6.78 *15*	11.94 *6*	4.06 *16*	2.64 *15*	11.90 *11*	9.44 *19*	14.59 *19*	18.20 *12*	1.84 *14*	10.22 *13*
*2 Dyn. prog.	4.12 *11*	4.63 *13*	12.34 *8*	4.84 *19*	3.71 *19*	13.26 *12*	10.10 *20*	15.01 *20*	17.12 *11*	3.33 *18*	14.04 *17*
17 Shao	9.67 *18*	7.04 *16*	35.63 *19*	4.25 *17*	3.19 *18*	30.14 *20*	6.01 *16*	6.70 *14*	43.91 *20*	2.36 *15*	33.01 *20*
16 Fast Correl.	9.76 *19*	13.85 *20*	24.39 *16*	4.76 *18*	1.87 *13*	22.49 *18*	6.48 *18*	10.36 *17*	31.29 *18*	8.42 *20*	12.68 *16*
18 Max surf.	11.10 *20*	10.70 *18*	41.99 *20*	5.51 *20*	5.56 *20*	27.39 *19*	4.36 *15*	4.78 *12*	41.13 *19*	4.17 *19*	27.88 *19*

Table 5: *Comparative performance of stereo algorithms, using the three performance measures* $B_{\overline{\mathcal{O}}}$ *(*bad_pixels_nonocc*),* $B_{\overline{\mathcal{T}}}$ *(*bad_pixels_textureless*), and* $B_{\mathcal{D}}$ *(*bad_pixels_discont*). All algorithms are run with constant parameter settings across all images. The small numbers indicate the rank of each algorithm in each column. The algorithms are listed roughly in decreasing order of overall performance, and the minimum (best) value in each column is shown in bold. Algorithms implemented by us are marked with a star.*

Next, cooperative and diffusion-based methods (5, 9, 15) do reasonably well, but often get the boundaries wrong, especially on the more complex Tsukuba images. On the highly textured and relatively simple Map sequence, however, they can outperform some of the full optimization approaches. The Map sequence is also noisier than the others, which works against algorithms that are sensitive to internal parameter settings. (In these experiments, we asked everyone to use a single set of parameters for all four datasets.)

Lastly, local (1, 12, 14, 16) and scanline methods (2, 3, 7) perform less well, although (14) which performs additional consistency checks and clean-up steps does reasonably well, as does the compact window approach (12), which uses a sophisticated adaptive window. Simpler local approaches such as SSD+MF (1) generally do poorly in textureless areas (if the window size is small) or near discontinuities (if the window is large). The disparity maps created by the scanline-based algorithms (DP and SO) are promising and show a lot of detail, but the larger quantitative errors are clearly a result of the "streaking" due to the lack of inter-scanline consistency.

To demonstrate the importance of parameter settings, Table 6 compares the overall results ($B_{\overline{\mathcal{O}}}$) of algorithms 1–5 for the fixed parameters listed in Table 4 with the "best" results when parameters are allowed to vary for each image. Note that we did not perform a true optimization over all parameters values, but rather simply chose the overall best results among the entire set of experiments we performed. It can be seen that for some of the algorithms the performance can be improved substantially with different parameters. The global optimization algorithms are particularly sensitive to the parameter λ, and DP is also sensitive to the occlusion cost parameter. This is consistent with our observations in Section 6.3. Note that the Map image pair can virtually be "solved" using GC, Bay, or SSD, since the images depict a simple geometry and are well textured. More challenging data sets with many occlusions and textureless regions may be useful in future extensions of this study.

Finally, we take a brief look at the efficiency of different methods. Table 7 lists the image sizes and number of disparity levels for each image pair, and the running times for 9 selected algorithms. Not surprisingly,

	Tsukuba		Sawtooth		Venus		Map	
	fixed	best	**fixed**	best	**fixed**	best	**fixed**	best
1 SSD	**5.23**	5.23	**2.21**	1.55	**3.74**	2.92	**0.66**	0.22
2 DP	**4.12**	3.82	**4.84**	3.70	**10.10**	9.13	**3.33**	1.21
3 SO	**5.08**	4.66	**4.06**	3.47	**9.44**	8.31	**1.84**	1.04
4 GC	**1.94**	1.94	**1.30**	0.98	**1.79**	1.48	**0.31**	0.09
5 Bay	**6.49**	6.49	**1.45**	1.45	**4.00**	4.00	**0.20**	0.20

Table 6: *Overall performance $B_{\overline{O}}$ (*bad_pixels_nonocc*) for algorithms 1–5, both using fixed parameters across all images, and best parameters for each image. Note that for some algorithms significant performance gains are possible if parameters are allowed to vary for each image.*

the speed-optimized methods (14 and 16) are fastest, followed by local and scanline-based methods (1 – SSD, 2 – DP, 3 – SO). Our implementations of Graph cuts (4) and Bayesian diffusion (5) are several orders of magnitude slower. The authors' implementations of the graph cut methods (10 and 11), however, are much faster than our implementation. This is due to the new max-flow code by Boykov and Kolmorogov [21], which is available at `www.cs.cornell.edu/People/vnk/software.html`.

In summary, if efficiency is an issue, a simple shiftable-window method is a good choice. In particular, method 14 by Hirschmüller [53] is among the fastest and produces very good results. New implementations of graph-cut methods give excellent results and have acceptable running times. Further research is needed to fully exploit the potential of scanline methods without sacrificing their efficiency.

8. Conclusion

In this paper, we have proposed a taxonomy for dense two-frame stereo correspondence algorithms. We use this taxonomy to highlight the most important features of existing stereo algorithms and to study important algorithmic components in isolation. We have implemented a suite of stereo matching algorithm components and constructed a test harness that can be used to combine these, to vary the algorithm parameters in a controlled way, and to test the performance of these algorithm on interesting data sets. We have also produced some new calibrated multi-view stereo data sets with hand-labeled ground truth. We have performed an extensive experimental investigation in order to assess the impact of the different algorithmic components. The experiments reported here have demonstrated the limitations of local methods, and have assessed the value of different global techniques and their sensitivity to key parameters.

We hope that publishing this study along with our sample code and data sets on the Web will encourage other stereo researchers to run their algorithms on our data and to report their comparative results. Since publishing the initial version of this paper as a technical report [98], we have already received experimental results (disparity maps) from 15 different research groups, and we hope to obtain more in the future. We are planning to maintain the on-line version of Table 5 that lists the overall results of the currently best-performing algorithms on our web site. We also hope that some researchers will take the time to add their algorithms to our framework for others to use and to build upon. In the long term, we hope that our efforts will lead to some set of data and testing methodology becoming an accepted standard in the stereo correspondence community, so that new algorithms will have to pass a "litmus test" to demonstrate that they improve on the state of the art.

There are many other open research questions we would like to address. How important is it to devise the right cost function (e.g., with better gradient-dependent smoothness terms) in global optimization algorithms vs. how important is it to find a global minimum? What are the best (sampling-invariant) matching metrics? What kind of adaptive/shiftable windows work best? Is it possible to automatically adapt parameters to different images? Also, is prediction error a useful metric for gauging the quality of stereo algorithms? We would also like to try other existing data sets and to produce some labeled data sets that are not all piecewise planar.

Once this study has been completed, we plan to move on to study multi-frame stereo matching with arbitrary camera geometry. There are many technical solutions possible to this problem, including voxel representations, layered representations, and multi-view representations. This more general version of the correspondence problem should also prove to be more useful for image-based rendering applications.

Developing the taxonomy and implementing the algorithmic framework described in this paper has given us a much deeper understanding of what does and does not work well in stereo matching. We hope that other researchers will also take the time to carefully analyze the behavior of their own algorithms using the framework and methodology developed in this paper, and that this will lead to a deeper understanding of the complex behavior of stereo correspondence algorithms.

	Tsukuba	Sawtooth	Venus	Map
width	384	434	434	284
height	288	380	383	216
disparity levels	16	20	20	30
Time (seconds):				
14 – Realtime	0.1	0.2	0.2	0.1
16 – Efficient	0.2	0.3	0.3	0.2
*1 – SSD+MF	1.1	1.5	1.7	0.8
*2 – DP	1.0	1.8	1.9	0.8
*3 – SO	1.1	2.2	2.3	1.3
10 – GC	23.6	48.3	51.3	22.3
11 – GC+occlusions	69.8	154.4	239.9	64.0
*4 – GC	662.0	735.0	829.0	480.0
*5 – Bay	1055.0	2049.0	2047.0	1236.0

Table 7: *Running times of selected algorithms on the four image pairs.*

Acknowledgements

Many thanks to Ramin Zabih for his help in laying the foundations for this study and for his valuable input and suggestions throughout this project. We are grateful to Y. Ohta and Y. Nakamura for supplying the ground-truth imagery from the University of Tsukuba, and to Lothar Hermes for supplying the synthetic images from the University of Bonn. Thanks to Padma Ugbabe for helping to label the image regions, and to Fred Lower for providing his paintings for our image data sets. Finally, we would like to thank Stan Birchfield, Yuri Boykov, Minglung Gong, Heiko Hirschmüller, Vladimir Kolmogorov, Sang Hwa Lee, Mike Lin, Karsten Mühlmann, Sébastien Roy, Juliang Shao, Harry Shum, Changming Sun, Jian Sun, Carlo Tomasi, Olga Veksler, and Larry Zitnick, for running their algorithms on our data sets and sending us the results, and Gary Bradski for helping to facilitate this comparison of stereo algorithms.

This research was supported in part by NSF CAREER grant 9984485.

References

[1] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *IJCV*, 2(3):283–310, 1989.

[2] R. D. Arnold. Automated stereo perception. Technical Report AIM-351, Artificial Intelligence Laboratory, Stanford University, 1983.

[3] H. Baker and T. Binford. Depth from edge and intensity based stereo. In *IJCAI81*, pages 631–636, 1981.

[4] H. H. Baker. Edge based stereo correlation. In L. S. Baumann, editor, *Image Understanding Workshop*, pages 168–175. Science Applications International Corporation, 1980.

[5] S. Baker, R. Szeliski, and P. Anandan. A layered approach to stereo reconstruction. In *CVPR*, pages 434–441, 1998.

[6] S. T. Barnard. Stochastic stereo matching over scale. *IJCV*, 3(1):17–32, 1989.

[7] S. T. Barnard and M. A. Fischler. Computational stereo. *ACM Comp. Surveys*, 14(4):553–572, 1982.

[8] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. *IJCV*, 12(1):43–77, 1994.

[9] P. N. Belhumeur. A Bayesian approach to binocular stereopsis. *IJCV*, 19(3):237–260, 1996.

[10] P. N. Belhumeur and D. Mumford. A Bayesian treatment of the stereo correspondence problem using half-occluded regions. In *CVPR*, pages 506–512, 1992.

[11] J. R. Bergen, P. Anandan, K. J. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *ECCV*, pages 237–252, 1992.

[12] S. Birchfield and C. Tomasi. Depth discontinuities by pixel-to-pixel stereo. In *ICCV*, pages 1073–1080, 1998.

[13] S. Birchfield and C. Tomasi. A pixel dissimilarity measure that is insensitive to image sampling. *IEEE TPAMI*, 20(4):401–406, 1998.

[14] S. Birchfield and C. Tomasi. Multiway cut for stereo and motion with slanted surfaces. In *ICCV*, pages 489–495, 1999.

[15] M. J. Black and P. Anandan. A framework for the robust estimation of optical flow. In *ICCV*, pages 231–236, 1993.

[16] M. J. Black and A. Rangarajan. On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. *IJCV*, 19(1):57–91, 1996.

[17] A. Blake and A. Zisserman. *Visual Reconstruction*. MIT Press, Cambridge, MA, 1987.

[18] A. F. Bobick and S. S. Intille. Large occlusion stereo. *IJCV*, 33(3):181–200, 1999.

[19] R. C. Bolles, H. H. Baker, and M. J. Hannah. The JISCT stereo evaluation. In *DARPA Image Understanding Workshop*, pages 263–274, 1993.

[20] R. C. Bolles, H. H. Baker, and D. H. Marimont. Epipolar-plane image analysis: An approach to determining structure from motion. *IJCV*, 1:7–55, 1987.

[21] Y. Boykov and V. Kolmogorov. A new algorithm for energy

minimization with discontinuities. In *Intl. Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 205–220, 2001.

[22] Y. Boykov, O. Veksler, and R. Zabih. A variable window approach to early vision. *IEEE TPAMI*, 20(12):1283–1294, 1998.

[23] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE TPAMI*, 23(11):1222–1239, 2001.

[24] A. Broadhurst, T. Drummond, and R. Cipolla. A probabilistic framework for space carving. In *ICCV*, volume I, pages 388–393, 2001.

[25] L. G. Brown. A survey of image registration techniques. *Computing Surveys*, 24(4):325–376, 1992.

[26] P. J. Burt and E. H. Adelson. The Laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, COM-31(4):532–540, 1983.

[27] J. F. Canny. A computational approach to edge detection. *IEEE TPAMI*, 8(6):34–43, 1986.

[28] P. B. Chou and C. M. Brown. The theory and practice of Bayesian image labeling. *IJCV*, 4(3):185–210, 1990.

[29] S. D. Cochran and G. Medioni. 3-D surface description from binocular stereo. *IEEE TPAMI*, 14(10):981–994, 1992.

[30] R. T. Collins. A space-sweep approach to true multi-image matching. In *CVPR*, pages 358–363, 1996.

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

[32] I. J. Cox, S. Roy, and S. L. Hingorani. Dynamic histogram warping of image pairs for constant image brightness. In *IEEE International Conference on Image Processing (ICIP'95)*, volume 2, pages 366–369. IEEE Computer Society, 1995.

[33] B. Culbertson, T. Malzbender, and G. Slabaugh. Generalized voxel coloring. In *International Workshop on Vision Algorithms*, pages 100–114, Kerkyra, Greece, 1999. Springer.

[34] J. S. De Bonet and P. Viola. Poxels: Probabilistic voxelized volume reconstruction. In *ICCV*, pages 418–425, 1999.

[35] R. Deriche. Fast algorithms for low-level vision. *IEEE TPAMI*, 12(1):78–87, 1990.

[36] P. Dev. Segmentation processes in visual perception: A cooperative neural model. COINS Technical Report 74C-5, University of Massachusetts at Amherst, 1974.

[37] U. R. Dhond and J. K. Aggarwal. Structure from stereo—a review. *IEEE Trans. on Systems, Man, and Cybern.*, 19(6):1489–1510, 1989.

[38] O. Faugeras and R. Keriven. Variational principles, surface evolution, PDE's, level set methods, and the stereo problem. *IEEE Trans. Image Proc.*, 7(3):336–344, 1998.

[39] O. Faugeras and Q.-T. Luong. *The Geometry of Multiple Images*. MIT Press, Cambridge, MA, 2001.

[40] D. J. Fleet, A. D. Jepson, and M. R. M. Jenkin. Phase-based disparity measurement. *CVGIP*, 53(2):198–210, 1991.

[41] T. Frohlinghaus and J. M. Buhmann. Regularizing phase-based stereo. In *ICPR*, volume A, pages 451–455, 1996.

[42] P. Fua. A parallel stereo algorithm that produces dense depth maps and preserves image features. *Machine Vision and Applications*, 6:35–49, 1993.

[43] P. Fua and Y. G. Leclerc. Object-centered surface reconstruction: Combining multi-image stereo and shading. *IJCV*, 16:35–56, 1995.

[44] E. Gamble and T. Poggio. Visual integration and detection of discontinuities: the key role of intensity edges. A. I. Memo 970, AI Lab, MIT, 1987.

[45] D. Geiger and F. Girosi. Parallel and deterministic algorithms for MRF's: Surface reconstruction. *IEEE TPAMI*, 13(5):401–412, 1991.

[46] D. Geiger, B. Ladendorf, and A. Yuille. Occlusions and binocular stereo. In *ECCV*, pages 425–433, 1992.

[47] S. Geman and D. Geman. Stochastic relaxation, Gibbs distribution, and the Bayesian restoration of images. *IEEE TPAMI*, 6(6):721–741, 1984.

[48] M. A. Gennert. Brightness-based stereo matching. In *ICCV*, pages 139–143, 1988.

[49] M. Gong and Y.-H. Yang. Multi-baseline stereo matching using genetic algorithm. In *IEEE Workshop on Stereo and Multi-Baseline Vision*, 2001. IJCV (this issue).

[50] W. E. L. Grimson. Computational experiments with a feature based stereo algorithm. *IEEE TPAMI*, 7(1):17–34, 1985.

[51] M. J. Hannah. *Computer Matching of Areas in Stereo Images*. PhD thesis, Stanford University, 1974.

[52] R. I. Hartley and A. Zisserman. *Multiple View Geometry*. Cambridge University Press, Cambridge, UK, 2000.

[53] H. Hirschmüller. Improvements in real-time correlation-based stereo vision. In *IEEE Workshop on Stereo and Multi-Baseline Vision*, 2001. IJCV (this issue).

[54] Y. C. Hsieh, D. McKeown, and F. P. Perlant. Performance evaluation of scene registration and stereo matching for cartographic feature extraction. *IEEE TPAMI*, 14(2):214–238, 1992.

[55] H. Ishikawa and D. Geiger. Occlusions, discontinuities, and epipolar lines in stereo. In *ECCV*, pages 232–248, 1998.

[56] M. R. M. Jenkin, A. D. Jepson, and J. K. Tsotsos. Techniques for disparity measurement. *CVGIP: Image Understanding*, 53(1):14–30, 1991.

[57] D. G. Jones and J. Malik. A computational framework for determining stereo correspondence from a set of linear spatial filters. In *ECCV*, pages 395–410, 1992.

[58] T. Kanade. Development of a video-rate stereo machine. In *Image Understanding Workshop*, pages 549–557, Monterey, CA, 1994. Morgan Kaufmann Publishers.

[59] T. Kanade et al. A stereo machine for video-rate dense depth mapping and its new applications. In *CVPR*, pages 196–202, 1996.

[60] T. Kanade and M. Okutomi. A stereo matching algorithm with an adaptive window: Theory and experiment. *IEEE TPAMI*, 16(9):920–932, 1994.

[61] S. B. Kang, R. Szeliski, and J. Chai. Handling occlusions in dense multi-view stereo. In *CVPR*, 2001.

[62] S. B. Kang, J. Webb, L. Zitnick, and T. Kanade. A multi-baseline stereo system with active illumination and real-time image acquisition. In *ICCV*, pages 88–93, 1995.

[63] M. Kass. Linear image features in stereopsis. *IJCV*, 1(4):357–368, 1988.

[64] R. Kimura et al. A convolver-based real-time stereo machine (SAZAN). In *CVPR*, volume 1, pages 457–463, 1999.

[65] V. Kolmogorov and R. Zabih. Computing visual correspondence with occlusions using graph cuts. In *ICCV*, volume II, pages 508–515, 2001.

[66] K. N. Kutulakos. Approximate N-view stereo. In *ECCV*, volume I, pages 67–83, 2000.

[67] K. N. Kutulakos and S. M. Seitz. A theory of shape by space carving. *IJCV*, 38(3):199–218, 2000.

[68] S. H. Lee, Y. Kanatsugu, and J.-I. Park. Hierarchical stochastic diffusion for disparity estimation. In *IEEE Workshop on Stereo and Multi-Baseline Vision*, 2001. IJCV (this issue).

[69] M. Lin and C. Tomasi. Surfaces with occlusions from layered stereo. Technical report, Stanford University, 2002. In preparation.

[70] C. Loop and Z. Zhang. Computing rectifying homographies for stereo vision. In *CVPR*, volume I, pages 125–131, 1999.

[71] B. D. Lucas and T. Kanade. An iterative image registration technique with an application in stereo vision. In *Seventh International Joint Conference on Artificial Intelligence (IJCAI-81)*, pages 674–679, Vancouver, 1981.

[72] D. Marr. *Vision*. W. H. Freeman and Company, New York, 1982.

[73] D. Marr and T. Poggio. Cooperative computation of stereo disparity. *Science*, 194:283–287, 1976.

[74] D. C. Marr and T. Poggio. A computational theory of human stereo vision. *Proceedings of the Royal Society of London*, B 204:301–328, 1979.

[75] J. Marroquin, S. Mitter, and T. Poggio. Probabilistic solution of ill-posed problems in computational vision. *Journal of the American Statistical Association*, 82(397):76–89, 1987.

[76] J. L. Marroquin. Design of cooperative networks. Working Paper 253, AI Lab, MIT, 1983.

[77] L. Matthies, R. Szeliski, and T. Kanade. Kalman filter-based algorithms for estimating depth from image sequences. *IJCV*, 3:209–236, 1989.

[78] A. Mitiche and P. Bouthemy. Computation and analysis of image motion: A synopsis of current problems and methods. *IJCV*, 19(1):29–55, 1996.

[79] K. Mühlmann, D. Maier, J. Hesser, and R. Männer. Calculating dense disparity maps from color stereo images, an efficient implementation. In *IEEE Workshop on Stereo and Multi-Baseline Vision*, 2001. IJCV (this issue).

[80] J. Mulligan, V. Isler, and K. Daniilidis. Performance evaluation of stereo for tele-presence. In *ICCV*, volume II, pages 558–565, 2001.

[81] Y. Nakamura, T. Matsuura, K. Satoh, and Y. Ohta. Occlusion detectable stereo — occlusion patterns in camera matrix. In *CVPR*, pages 371–378, 1996.

[82] H. K. Nishihara. Practical real-time imaging stereo matcher. *Optical Engineering*, 23(5):536–545, 1984.

[83] Y. Ohta and T. Kanade. Stereo by intra- and inter-scanline search using dynamic programming. *IEEE TPAMI*, 7(2):139–154, 1985.

[84] M. Okutomi and T. Kanade. A locally adaptive window for signal matching. *IJCV*, 7(2):143–162, 1992.

[85] M. Okutomi and T. Kanade. A multiple-baseline stereo. *IEEE TPAMI*, 15(4):353–363, 1993.

[86] M. Otte and H.-H. Nagel. Optical flow estimation: advances and comparisons. In *ECCV*, volume 1, pages 51–60, 1994.

[87] T. Poggio, V. Torre, and C. Koch. Computational vision and regularization theory. *Nature*, 317(6035):314–319, 1985.

[88] S. B. Pollard, J. E. W. Mayhew, and J. P. Frisby. PMF: A stereo correspondence algorithm using a disparity gradient limit. *Perception*, 14:449–470, 1985.

[89] K. Prazdny. Detection of binocular disparities. *Biological Cybernetics*, 52(2):93–99, 1985.

[90] L. H. Quam. Hierarchical warp stereo. In *Image Understanding Workshop*, pages 149–155, New Orleans, Louisiana, 1984. Science Applications International Corporation.

[91] S. Roy. Stereo without epipolar lines: A maximum flow formulation. *IJCV*, 34(2/3):147–161, 1999.

[92] S. Roy and I. J. Cox. A maximum-flow formulation of the N-camera stereo correspondence problem. In *ICCV*, pages 492–499, 1998.

[93] T. W. Ryan, R. T. Gray, and B. R. Hunt. Prediction of correlation errors in stereo-pair images. *Optical Engineering*, 19(3):312–322, 1980.

[94] H. Saito and T. Kanade. Shape reconstruction in projective grid space from large number of images. In *CVPR*, volume 2, pages 49–54, 1999.

[95] D. Scharstein. Matching images by comparing their gradient fields. In *ICPR*, volume 1, pages 572–575, 1994.

[96] D. Scharstein. *View Synthesis Using Stereo Vision*, volume 1583 of *Lecture Notes in Computer Science (LNCS)*. Springer-Verlag, 1999.

[97] D. Scharstein and R. Szeliski. Stereo matching with nonlinear diffusion. *IJCV*, 28(2):155–174, 1998.

[98] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. Technical Report MSR-TR-2001-81, Microsoft Research, 2001.

[99] D. Scharstein, R. Szeliski, and R. Zabih. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. In *IEEE Workshop on Stereo and Multi-Baseline Vision*, 2001.

[100] P. Seitz. Using local orientation information as image primitive for robust object recognition. In *SPIE Visual Communications and Image Processing IV*, volume 1199, pages 1630–1639, 1989.

[101] S. M. Seitz and C. M. Dyer. Photorealistic scene reconstruction by voxel coloring. *IJCV*, 35(2):1–23, 1999.

[102] J. Shade, S. Gortler, L.-W. He, and R. Szeliski. Layered depth images. In *SIGGRAPH*, pages 231–242, 1998.

[103] J. Shah. A nonlinear diffusion model for discontinuous disparity and half-occlusion in stereo. In *CVPR*, pages 34–40, 1993.

[104] J. Shao. Combination of stereo, motion and rendering for 3d footage display. In *IEEE Workshop on Stereo and Multi-Baseline Vision*, 2001. IJCV (this issue).

[105] M. Shimizu and M. Okutomi. Precise sub-pixel estimation on area-based matching. In *ICCV*, volume I, pages 90–97, 2001.

[106] H.-Y. Shum and R. Szeliski. Stereo reconstruction from multiperspective panoramas. In *ICCV*, pages 14–21, 1999.

[107] E. P. Simoncelli, E. H. Adelson, and D. J. Heeger. Probability distributions of optic flow. In *CVPR*, pages 310–315, 1991.

[108] C. Sun. Rectangular subregioning and 3-d maximum-

surface techniques for fast stereo matching. In *IEEE Workshop on Stereo and Multi-Baseline Vision*, 2001. IJCV (this issue).

[109] J. Sun, H. Y. Shum, and N. N. Zheng. Stereo matching using belief propagation. In *ECCV*, 2002. Submitted.

[110] R. Szeliski. *Bayesian Modeling of Uncertainty in Low-Level Vision*. Kluwer Academic Publishers, Boston, MA, 1989.

[111] R. Szeliski. Prediction error as a quality metric for motion and stereo. In *ICCV*, pages 781–788, 1999.

[112] R. Szeliski and J. Coughlan. Spline-based image registration. *IJCV*, 22(3):199–218, 1997.

[113] R. Szeliski and P. Golland. Stereo matching with transparency and matting. *IJCV*, 32(1):45–61, 1999. Special Issue for Marr Prize papers.

[114] R. Szeliski and G. Hinton. Solving random-dot stereograms using the heat equation. In *CVPR*, pages 284–288, 1985.

[115] R. Szeliski and D. Scharstein. Symmetric sub-pixel stereo matching. In *ECCV*, 2002. Submitted.

[116] R. Szeliski and R. Zabih. An experimental comparison of stereo algorithms. In *International Workshop on Vision Algorithms*, pages 1–19, Kerkyra, Greece, 1999. Springer.

[117] H. Tao, H. Sawhney, and R. Kumar. A global matching framework for stereo computation. In *ICCV*, volume I, pages 532–539, 2001.

[118] M. Tekalp. *Digital Video Processing*. Prentice Hall, Upper Saddle River, NJ, 1995.

[119] D. Terzopoulos. Regularization of inverse visual problems involving discontinuities. *IEEE TPAMI*, 8(4):413–424, 1986.

[120] D. Terzopoulos and K. Fleischer. Deformable models. *The Visual Computer*, 4(6):306–331, 1988.

[121] D. Terzopoulos and D. Metaxas. Dynamic 3D models with local and global deformations: Deformable superquadrics. *IEEE TPAMI*, 13(7):703–714, 1991.

[122] Q. Tian and M. N. Huhns. Algorithms for subpixel registration. *CVGIP*, 35:220–233, 1986.

[123] O. Veksler. *Efficient Graph-based Energy Minimization Methods in Computer Vision*. PhD thesis, Cornell University, 1999.

[124] O. Veksler. Stereo matching by compact windows via minimum ratio cycle. In *ICCV*, volume I, pages 540–547, 2001.

[125] J. Y. A. Wang and E. H. Adelson. Layered representation for motion analysis. In *CVPR*, pages 361–366, 1993.

[126] A. Witkin, D. Terzopoulos, and M. Kass. Signal matching through scale space. *IJCV*, 1:133–144, 1987.

[127] Y. Yang, A. Yuille, and J. Lu. Local, global, and multilevel stereo matching. In *CVPR*, pages 274–279, 1993.

[128] A. L. Yuille and T. Poggio. A generalized ordering constraint for stereo correspondence. A.I. Memo 777, AI Lab, MIT, 1984.

[129] R. Zabih and J. Woodfill. Non-parametric local transforms for computing visual correspondence. In *ECCV*, volume II, pages 151–158, 1994.

[130] Z. Zhang. Determining the epipolar geometry and its uncertainty: A review. *IJCV*, 27(2):161–195, 1998.

[131] Z. Zhang. A flexible new technique for camera calibration. *IEEE TPAMI*, 22(11):1330–1334, 2000.

[132] C. L. Zitnick and T. Kanade. A cooperative algorithm for stereo matching and occlusion detection. *IEEE TPAMI*, 22(7):675–684, 2000.