# An Optimal Time-Space Algorithm for Dense Stereo Matching

Gutemberg Guerra Filho Yiannis Aloimonos

## Abstract

*An original survey addressing time-space complexity covers several stereo matching algorithms and running time experiments are reported. Taking the point of view that good reconstruction needs to be solved in feedback loops, we then present a new dense stereo matching based on a path computation in disparity space. A procedure which improves disparity maps is also introduced as a post-processing step for any technique solving a dense stereo matching problem. Compared to other algorithms, our algorithm has optimal time-space complexity. The algorithm is faster than "real-time" techniques while producing comparable results. The correctness of our algorithm is demonstrated by experiments in real and synthetic benchmark data.*

## 1. Introduction

The dense stereo matching problem consists in finding a dense disparity map from a pair of images in a stereo configuration. Applications are view synthesis, augmented reality, image-based rendering, 3D reconstruction, object detection for mobile robots, and autonomous vehicles. Many of these applications require real-time performance and, consequently, an optimal algorithm concerning time-space complexity is an important matter specially with high resolution images at s high frequency (frame rate).

Stereo matching is an ill-posed problem and difficulties arise from the aperture problem in textureless regions, depth discontinuities, and occluded pixels. There are several constraints which aid to overcome these issues. The ordering constraint states that if an object is to the left of another in one stereo image, it is also to the left in the other image. Assuming this ordering constraint, the stereo matching problem is reduced to a path finding problem.

We present a new approach to solve the dense stereo matching problem. Our approach reduces the matching of a scanline pair to a path computation in disparity space. A path is found by a number of local steps which assume continuity and deal with occlusions. Each local step is taken based only on local information, but the current state of a path represents global computation.

This new approach is not based in any previous technique. It implements a local search which computes a heuristic path while area-based algorithms search the whole disparity range. Our local search is greedy and differs from any optimization method used in stereo matching as dynamic programming and graph cut based algorithms. Therefore, our algorithm falls into a new class of methods which performs a constant number of computation per pixels and does not iterate over a disparity range.

We also introduce a procedure which improves disparity maps in discontinuity regions. This routine can be applied as a post-processing step to the result of any technique solving the dense stereo matching problem.

The increasing demand for higher resolution images and for capturing faster motions at higher frame rates will always pose a challenge to computational power when real-time performance is required. Therefore, the use of asymptotic analysis to evaluate stereo algorithms is justified. In this paper, several stereo matching algorithms are assessed according to a time and space performance viewpoint. While a similar survey focuses on correctness issues [21], we address resource complexity for the first time. Running time experiments for several algorithms are reported consistently with the complexity analysis. This analysis shows that our algorithm is the only optimal algorithm in the literature concerning time complexity.

Our algorithm is proved to have optimal complexity, since our path framework requires the least amount of resources necessary to solve the stereo matching problem. The algorithm is faster than the so called "real-time" techniques and produces comparable results. The performance improvement over real-time solutions is achieved by avoiding a brute force search over the disparity range at every pixel. Our technique searches the disparity range only at possible occlusions.

The effectiveness of our algorithm is demonstrated by experiments on well-known benchmark stereo pairs (real and synthetic). The algorithm achieved good results with overall gross errors ranging from 0.71% to 3.85%.

We analyze the time-space complexity of several stereo matching approaches in Section 2. Our new algorithm and improving stereo routine are presented in Section 3 and the optimal performance is shown in Section 4. Some validation experiments are discussed in Section 5. In Section 6, we have our conclusions and suggestions for some generalizations.

## 2. Time-Space Complexity Review

We review several dense stereo matching algorithms from a performance perspective. The time and space complexity is analyzed for several approaches including area-based, dynamic programming, Bayesian, cooperative, graph cut, and layered methods. Due to the lack of space, we omit a detailed description of each technique. For a full description of each algorithm, the reader is referred to the original work.

The input for a dense stereo matching problem is a pair of rectified stereo images and the output is a disparity map. Both input and output are represented by matrices with $n = hw$ elements, where $h$ is the height of an image and $w$ is the width. Assuming a square image ($h = w$), the range of disparity is $[0, \Delta]$, where $\Delta$ is at most the image width $w = \sqrt{n}$.

### 2.1. Area-based Methods

A variable window algorithm [28] uses the integral image technique to compute an arbitrary rectangular window correlation in constant time. Pre-processing an integral image for each disparity takes $O(\Delta n)$ time and requires $O(\Delta n)$ space. The algorithm searches for square windows between some range of sizes by using dynamic programming to find the best window for every pixel in $O(n)$ time per disparity. Hence, the search for a minimum window at all disparities takes $O(\Delta n)$ time and space. Finally, the search for the best disparity takes also $O(\Delta n)$ time and space. Therefore, the overall complexity for a variable window approach is $O(\Delta n) = O(n^{1.5})$ time and space.

Instead of a limited number of windows, the matching cost for a class of "compact" windows may be computed [27]. Although the size of this class is exponential in the maximum window size $s$, a minimum ratio cycle algorithm for graphs achieves a $O(s^{1.5})$ time bound. This way, the overall time complexity becomes $O(s^{1.5}\Delta n)$. Assuming $s$ is independent of the input size, this method takes $O(\Delta n) = O(n^{1.5})$ time and $O(n)$ space.

Another multiple window approach uses a window configuration that has a small window in the center surrounded by $m$ partly overlapping windows [13]. The correlation value is the sum of the center correlation with the best surrounding correlation windows. Although each window correlation is computed in constant time with some preprocessing, the selection of the best windows requires sorting in $O(m \log m)$ time per pixel and disparity. This way, the algorithm takes $O(m \log m \Delta n)$ time, which becomes $O(\Delta n)$ since $m$ is independent of the input size. Therefore, this method takes $O(n^{1.5})$ time and requires $O(n)$ space.

### 2.2. Dynamic Programming Techniques

A dynamic programming process finds the best path through a disparity space in $O(\Delta w)$ time and space for each scanline [7]. This method also incorporates ground control points and intensity edges into the matching process. The winner-takes-all technique finds a candidate set of points and thresholding further reduces this set in $O(\Delta w)$ time and space per scanline. Edge detection is performed in $O(n)$ time and space for both images. Hence, this pre-processing takes $O(\Delta n)$ time and $O(n)$ space overall. Therefore, the method takes $O(\Delta n) = O(n^{1.5})$ time and $O(n)$ space.

A different path cost function uses occlusion penalty, match reward, and a dissimilarity measure insensitive to sampling [5]. For each scanline, a dual dynamic programming algorithm iterates over all $\Delta w$ cells in disparity space computing the best paths to the $2\Delta$ following cells. This way, the algorithm runs in $O(\Delta^2 n) = O(n^2)$ time and requires $O(n)$ space.

### 2.3. Bayesian Approaches

In a Bayesian approach [24], the Markov network requires $O(\Delta^2 n) = O(n^2)$ space to store hidden nodes. An approximate solution for the posterior probability is found by a Bayesian belief propagation algorithm. Belief propagation is an iterative inference algorithm that propagates messages in the network. A max-product belief propagation algorithm takes time $O(k\Delta^2 n) = O(kn^2)$, where $k$ is the number of iterations.

The stochastic diffusion optimization method [16] has a conditional probability with a likelihood model, a disparity field model, and a line field model. The likelihood model is computed initially in $O(\Delta n)$ time and space. The last two models use the MRF model with their neighborhood configurations $N$. This way, they are computed in $O(Nn)$ time and require $O(\Delta n)$ space, where $N$ reflects the interactivity of neighboring fields and is sufficient to calculate the probabilistic expectation. Since $N$ is $O(\Delta)$, each state is computed in $O(\Delta n)$ time. The potential space is a 3D disparity space which is iteratively updated by the probabilistic expectation of the neighboring fields and the computational models. The potential space is diffused to a stable local state. Hence, the stochastic diffusion takes $O(k\Delta n)$ time and requires $O(\Delta n)$ space, where $k$ is the number of iterations. When the potential space converges, the optimal fields are deterministically estimated by the localized minimal potential condition in $O(\Delta n)$ time. Therefore, the overall complexity of the stochastic diffusion approach is $O(k\Delta n) = O(kn^{1.5})$ for time and $O(\Delta n) = O(n^{1.5})$ for space.

### 2.4. Cooperative Algorithms

A cooperative algorithm [18, 29] iterates support and inhibition. Matching values are stored in a 3D disparity space, where each element corresponds to a pixel in the reference image and a disparity relative to another image. Hence,

a disparity array requires $O(\Delta n) = O(n^{1.5})$ space. For each element in the disparity space, an update function of match values diffuses support among neighboring match values in a 3D region while inhibition weights down all matches along similar lines of sight. Since a ray of view corresponds to $2\Delta$ elements in the disparity space, the cooperative approach takes $O(k(rcd+\Delta)\Delta n)$, where $k$ is the number of iterations and $rcd$ is the size of the 3D support region. Assuming $rcd$ is constant, the time bound becomes $O(k\Delta^2 n) = O(kn^2)$.

A cooperative method may have only a support region to aggregate disparity with a non-uniform diffusion process [20]. The method uses a membrane diffusion model which only diverges to a certain amount from its initial value. The diffusion process is iterated and a measure of certainty decides whether to diffuse each pixel. A Bayesian model explicitly associates all possible disparities at each pixel with a scalar value between 0 and 1. Initially, the probability distribution from each pixel is based on the intensity errors between matching pixels. An update rule assumes independent distribution of adjacent disparity and corresponds to a smoothed energy. The diffusion computes support values for all pixels and disparities in $O(\Delta n)$ time. The certainty measure is computed in $O(\Delta)$ time for each pixel, and the update step is computed in constant time for each pixel and disparity. This way, each iteration in the non-uniform Bayesian diffusion method takes $O(\Delta n)$ time and space. Therefore, the approach takes $O(k\Delta n) = O(kn^{1.5})$ time and $O(\Delta n) = O(n^{1.5})$ space, where $k$ is the number of iterations.

## 2.5. Graph Cut Methods

A graph cut based algorithm takes $O(k\Delta f(V,E))$ time, where $k$ is the number of iterations, $f$ is the time complexity of graph construction and a maximum flow algorithm, and $(V,E)$ is the size of the graph $G$ modeling the energy function. Assuming the energy function is defined by constants independent of the image size, the number $k$ of iterations is $O(n)$ [26].

In general, each graph cut technique differs by the energy function used and, consequently, by the corresponding graph $G$ which models the function. In a multi-view method [19], the graph represents a 3D mesh corresponding to the disparity space volume. Hence, the number of vertices in $G$ is $O(\Delta n)$. Each vertex is internally six-connected and, consequently, the number of edges is also $O(\Delta n)$.

In a discontinuity preserving method [9], the set of vertices corresponds to pixels in the image and to auxiliary nodes in a fixed neighborhood of the current partition boundaries. Hence, the number of vertices in $G$ is $O(n)$ and, since each vertex has a constant number of edges, the set of edges is also $O(n)$ in size.

In the occlusion handling method [14, 15], the vertices in $G$ correspond to possible pixel assignments. Since each pixel in one image may be assigned to $\Delta$ pixels in the other, the vertex set $V$ requires $O(\Delta n)$ space. The edge set in $G$ represents some fixed-size neighborhood criteria and also requires $O(\Delta n)$ space.

The weights of the edges are each computed in constant time for all graph models. This way, the graph construction takes $O(n)$ time and space for the discontinuity preserving method. For the multi-view method and occlusion handling method, the graph construction takes $O(\Delta n)$ time and space. The maximum flow problem is solved in $O(VE\log\frac{V^2}{E})$ time and requires $O(E)$ space [12]. Therefore, the discontinuity preserving algorithm takes $O(k\Delta n^2\log n) = O(n^{3.5}\log n)$ time and requires $O(n)$ space. The multi-view method and the occlusion handling graph cut based algorithm takes $O(k\Delta^3 n^2\log(\Delta n)) = O(n^{4.5}\log n)$ time and requires $O(\Delta n) = O(n^{1.5})$ space.

## 2.6. Layered Approaches

A layered approach [17] iteratively segments the images into surfaces and estimates the disparity map for each surface. Given an energy function satisfying some conditions, the segmentation is computed by a graph cut approach [9]. The disparity map for a particular surface is found by a surface fitting step. The surface fitting minimizes an energy function using standard gradient-based numerical methods. The layered algorithm takes $O(k_0(f_s(n)+f_d(n)))$ time and requires $O(g_s(n) + g_d(n))$ space, where $k_0$ is the number of iterations, $f_s$ ($g_s$) is the time (space) required to compute the surface segmentation using graph cuts, and $f_d$ ($g_d$) is the time (space) required to find the disparity maps using numerical optimization. A graph cut based algorithm implies that $f_s(n)$ is $O(k_1 n^{3.5}\log n)$ and $g_s(n)$ is $O(n^{1.5})$. For a quasi-Newton method, $f_d(n)$ is $O(k_2 n^2)$ and $g_d(n)$ is $O(k_2 n)$, where $k_2$ is the number of iterations until convergence [11]. Therefore, a layered approach takes $O(k_0(k_1 n^{3.5}\log n + k_2 n^2))$ time and requires $O(n^{1.5} + k_2 n)$ space.

An energy function that allows affine warpings displacements handles slanted surfaces [6]. The energy function is minimized by iteratively alternating between segmenting and fitting. The approach segments the image into non-overlapping regions corresponding to different surfaces by using a graph and finding a local minimum multiway cut of this graph. The graph $G(V,E)$ contains a vertex for every pixel in the image and for every possible label: $|V| = n + S$, where $S$ is the number of surfaces (labels) in the scene. Each pixel is linked to its neighbors and to each label: $|E| = (4 + S)n$. An approximate solution for a multiway cut is found by a graph cut approach in $O(k_1 S^2 f(V,E))$ time and $O(Sn)$ space [8], where $k_1$ is the number of iterations to find an approximate minimum multiway cut and $f(V,E)$ is the time complexity of the

maximum flow problem. Since $S$ is initially assumed to be $\Delta$, the complexity becomes $O(k_1 n^{2.5} \log n)$ and $O(n^{1.5})$ for time and space, respectively. The affine parameters of the displacement function for each region are found using a greedy algorithm [22]. A linear system with 3 unknowns is minimized by a Newton-Raphson technique in $O(n)$ time and space per iteration, since the system evaluation computes values for every pixel in all surfaces. Hence, the fitting step takes $O(k_2 n)$ time and $O(n)$ space, where $k_2$ is the number of iterations in the minimization. Therefore, this approach takes $O(k_0((k_1 n^{2.5} \log n) + (k_2 n)))$ time and requires $O(n^{1.5})$ space, where $k_0$ is the number of iterations of the segment-fit step.

### 2.7. Complexity Summary and Running Time

The next table summarizes the time-space complexity of stereo matching methods. The area-based methods have the best performance and usually are referred to as real-time algorithms. Hence, area-based and dynamic programming techniques are considered fast. Bayesian and cooperative approaches have a good performance, while graph cut and layered methods require much more resources.

The error column displays the gross error evaluated for the tsukuba real image [21]. The area-based methods and dynamic programming have reasonable results ($[2.35\%, 5.12\%]$). Bayesian and Cooperative techniques have some of the best results ($[1.15\%, 6.49\%]$). Graph cut and layered approaches perform very well concerning errors ($[1.19\%, 8.08\%]$).

We report the results of experiments to compare the running (execution) time performance of stereo matching algorithms. The running time experiments are consistent with the time complexity analysis (see Fig. 1). The experiments are performed on a Dell Computer with 2.80GHz processor and 2Gb main memory running a Linux environment for Windows XP.

The performance is evaluated by varying the size of the input images. The input size ranges from 40000 pixels ($200 \times 200$) to 400000 pixels ($632 \times 632$). The execution time for each algorithm was determined as the median of several runs.

The graph cut implementation used [1] is described by Kolmogorov and Zabih [14]. The cooperative algorithm evaluated [2] is presented by Scharstein and Szeliski [20]. We used a regular diffusion (80 iterations). The belief propagation method [3] is proposed by Felzenszwalb and Huttenlocher [10]. An area-based algorithm [4, 25] with shiftable square windows (window size 15) is used [2] in our experiments. Dynamic programming [2] is represented by the method presented by Bobick and Intille [7]. The running time of our optimal algorithm is also reported. For all algorithms, the disparity range was adjusted accordingly when necessary.

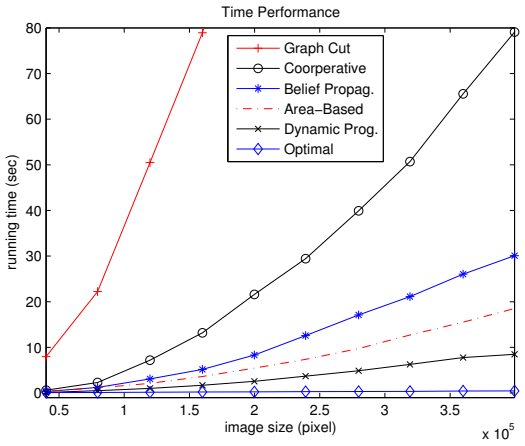



Figure 1: Running time performance of stereo matching algorithms.

## 3. An Optimal Time-Space Algorithm

The epipolar constraint reduces the stereo matching problem to a 1D search. In a stereo rectified configuration, the epipolar lines are the horizontal scanlines with same $y$ coordinate. Each pixel in a scanline of the left image $I_L$ is matched to a pixel in a corresponding scaline of the right image $I_R$.

The disparity space for a pair of scanlines consists of a matrix where each element represents a grid node $(n_x, n_y)$ with the disparity cost between the pixels $p_L = (n_x, y) \in I_L$ and $p_R = (n_y, y) \in I_R$. The disparity cost is the sum of absolute differences (SAD): $\sum_{-s \leq r,c \leq s} |I_L(n_x + c, y + r) - I_R(n_y + c, y + r)|$. This way, the disparity space becomes an environment for a path computation, where each node $(n_x, n_y)$ in the path corresponds to a matching between pixels $p_L$ and $p_R$.

The path in the disparity space is found by a greedy heuristic algorithm. Therefore, global minimization such as dynamic programming is not used [23]. Since dense stereo matching is an ill-posed problem, the concept of optimal solution is non-existent or depends on an energy function definition which is particular to each method. Hence, a heuristic approach gives as good solution as any other technique but with a better performance. Our algorithm improves real-time (area-based) approaches by avoiding a brute force search over the disparity range at every pixel. The heuristic algorithm performs a brute force search only at occlusions.

For each pair of scanlines, the algorithm computes a path which consists of a sequence of nodes in disparity space. Each node $(n_x, n_y)$ in this disparity space is connected in a 3-neighborhood way: $N_3(n_x, n_y) = \{c, o^+, o^-\} = \{(n_x + 1, n_y + 1), (n_x + 1, n_y), (n_x, n_y + 1)\}$. The adjacent nodes are named continuous ($c$), positive occlusion ($o^+$), and negative occlusion ($o^-$). The source node is ob-

| Algorithm | Time | Space | Error |
|---|---|---|---|
| Area-based [27] | $O(n^{1.5})$ | $O(n)$ | 3.36% |
| Dynamic prog. [7] | $O(n^{1.5})$ | $O(n)$ | 4.12% |
| Area-based [13] | $O(n^{1.5})$ | $O(n)$ | 4.25% |
| Area-based [28] | $O(n^{1.5})$ | $O(n^{1.5})$ | 2.35% |
| Dynamic prog. [5] | $O(n^2)$ | $O(n)$ | 5.12% |
| Bayesian [16] | $O(kn^{1.5})$ | $O(n^{1.5})$ | 3.95% |
| Cooperative [20] | $O(kn^{1.5})$ | $O(n^{1.5})$ | 6.49% |
| Cooperative [18] | $O(kn^2)$ | $O(n^{1.5})$ | 1.67% |
| Cooperative [29] | $O(kn^2)$ | $O(n^{1.5})$ | 3.49% |
| Bayesian [24] | $O(kn^2)$ | $O(n^2)$ | 1.15% |
| Graph cut [9] | $O(n^{3.5}\log n)$ | $O(n)$ | 1.86% |
| Graph cut [14] | $O(n^{4.5}\log n)$ | $O(n^{1.5})$ | 1.19% |
| Graph cut [15] | $O(n^{4.5}\log n)$ | $O(n^{1.5})$ | 1.85% |
| Graph cut [19] | $O(n^{4.5}\log n)$ | $O(n^{1.5})$ | 2.98% |
| Layered [6] | $O(k_0((k_1 n^{2.5}\log n) + (k_2 n)))$ | $O(n^{1.5})$ | 8.08% |
| Layered [17] | $O(k_0(k_1 n^{3.5}\log n + k_2 n^2))$ | $O(n^{1.5} + k_2 n)$ | 1.58% |

Table 1: Time-space complexity and gross error of stereo matching algorithms for all pixels in the Tsukuba image [21].

tained by searching the best match for the first pixel in both scanlines: $(n_x, 1)$ or $(1, n_y)$. This way, the path starts from either the left or top side of the disparity space (see Fig. 2). A local step is performed until either the bottom or right side of the disparity space is reached. Hence, the destination node is any node representing a match for the last pixel in both scanlines: $(n_x, w)$ or $(h, n_y)$.

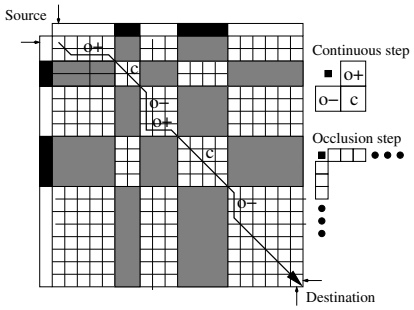



Figure 2: A local path through a binary disparity space.

The local step is either a continuous step or an occlusion step. A continuous step finds the minimum disparity cost in the 3-neighborhood of a node. This way, the next node in the path may represent a continuous match ($c$) or a potential occlusion ($o^+$ and $o^-$). Assuming the scene is smooth almost everywhere (continuity constraint), a continuous match is preferred over a potential occlusion. The local continuous step iterates over a range of correlation window sizes computing the SAD disparity cost among the 3-neighbors. If a potential occlusion ($o^+$ or $o^-$) is the neighbor with the minimum cost for the current window size, the size of the window is increased and the search for a continuous match continues. A continuous match happens when the neighbor node with the minimum cost is the continuous node ($c$). If the whole range of window sizes is explored and a continuous match is not found, then the best potential occlusion is selected as the local step.

An occlusion step is performed when a certain number of positive (negative) potential occlusion steps are performed successively. This number of potential occlusions is related to how much slant is considered in the scene. For a disparity node $(n_x, n_y)$, a positive occlusion step just selects the next node with the minimum disparity cost among all nodes $(n'_x, n_y)$, where $n'_x > n_x$. Similarly, a negative occlusion step finds the node with the minimum disparity cost among all nodes $(n_x, n'_y)$, where $n'_y > n_y$.

At each local step, the node $(n_x, n_y)$ corresponds to a match between a pixel $(n_x, y)$ in the left scanline and a pixel $(n_y, y)$ in the right scanline, where $y$ specifies the scanlines in the pair of images. Using the left image as a reference, the disparity of a pixel $(n_x, y)$ is $n_x - n_y$.

The ordering constraint imposes an orientation in the local path computed by the stereo algorithm. Considering a scanline, stereo matching may be performed in two possible directions: from left to right and from right to left. The disparity maps obtained by the algorithm using each direction may be different (see Fig. 3). This behavior of our stereo algorithm is due to the fact that the path computation is based only on local information.

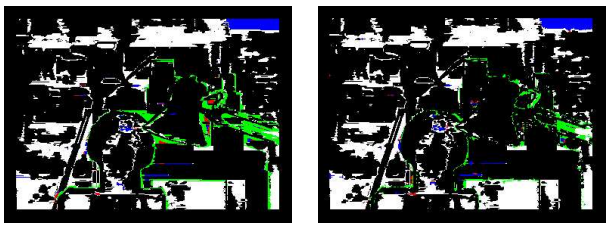
In order to deal with this orientation issue, the main algorithm computes the stereo matching in both directions and a consensus routine produces a single map from the two re-

(a) Left-to-right map. (b) Consensus map. (c) Right-to-left map.

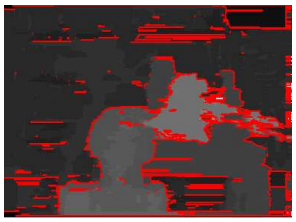
Figure 3: Disparity maps for both directions and the corresponding consensus map.

sults. Our consensus rule just assigns the least disparity for each pixel.

Since most of the gross errors in the stereo matching are close to depth discontinuities, we have designed an iterative algorithm to improve disparity maps in these regions. Initially, the algorithm identifies in the current disparity map $D$ every pixel $p$ with a 4-neighbor pixel $p' \in N_4(p)$ such that $|D(p) - D(p')| > 1$. These pixels would represent the boundary $B$ of objects in the scene (see Fig. 4).



(a) Gross errors before improvement. (b) Gross errors after improvement.



(c) Detected boundary of objects.

Figure 4: Disparity map improvement by boundary refinement.

The algorithm iteratively changes the disparity of boundary pixels until there is no local improvement in the disparity cost for all pixels in the boundary. For each pixel in the boundary, a new disparity cost candidate is the minimum cost achieved by replacing the current disparity with the disparity of all 4-neighbors. In this case, the disparity cost is a special SAD measure using a correlation window only with same disparity pixels. If the minimum disparity cost found among 4-neighbors for a pixel compares to the current cost by a ratio less than a certain threshold, the disparity of this pixel is changed. At the end of each iteration, the boundary is updated by checking the depth discontinuities in the changed disparity map.

# 4. Optimal Performance Analysis

Our stereo matching algorithm computes a path for each left-right pair of scanlines. Hence, the algorithm finds $O(h)$ paths. Each path computation involves a number of local steps. Since the source node is at the top/left side of the disparity space, the destination node is at the bottom/right side, and each continuous step moves either right or down, then the number of continuous steps is $O(w)$. Objects in the scene do not depend on the size of the images. Consequently, the number of occlusion regions in each scanline is constant. This way, a path computation performs $O(w)$ continuous steps in constant time and $O(1)$ occlusion steps in $O(w)$ time. Hence, each path is found in $O(w)$ time and the stereo matching algorithm takes $O(hw) = O(n)$ time.

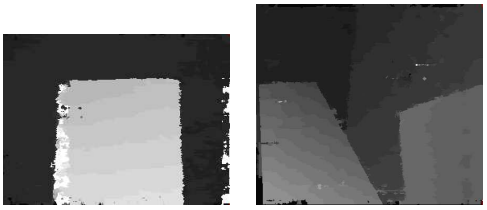
An iteration of the improving stereo algorithm updates each boundary pixel disparity in constant time. Since the number of objects in the scene is constant, the number of pixels in the boundary region is $O(h+w)$ and, consequently, a disparity update iteration takes $O(n^{0.5})$ time. A disparity update represents a move of the boundary region towards the correct location. Since a displacement of a boundary pixel is $O(h+w)$, the number of iterations towards the correct boundary is $O(n^{0.5})$. Therefore, the improving stereo algorithm takes $O(n)$ time.

The stereo matching algorithm requires $O(n)$ space for the disparity map. The improving stereo algorithm keeps the boundary region in a list which requires $O(h+w)$ space. Other data structures have the same size as a disparity map. Hence, the improving stereo procedure also requires linear

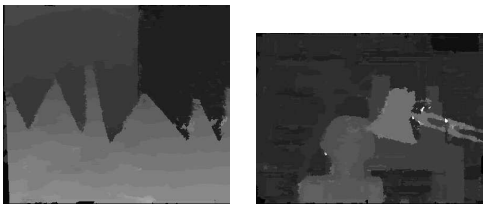
space. Therefore, our method is time-space linear and, consequently, optimal since the output (disparity map) lower bound for performance is linear.

# 5. Experiments

The algorithm designed and implemented is the result of experimentation over some parameters. These parameters include the maximum size of a correlation window considered in order to take a potential positive and negative occlusion step, the number of successive potential occlusions which defines a candidate occlusion, the size of the window for the special SAD used in the improving algorithm, and the associated ratio threshold in the improving stereo routine.



(a) map. (b) venus.



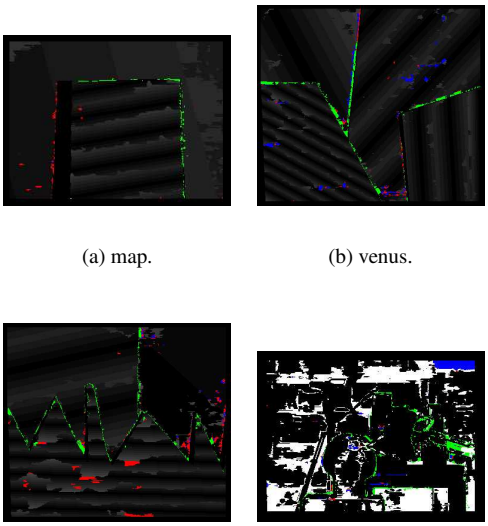
(c) sawtooth. (d) tsukuba.

Figure 5: Disparity maps computed for stereo benchmarks.

Correctness was tested with four stereo pairs for which there exists ground truth disparity. The original image benchmarks are found at [2], where many algorithms are surveyed and evaluated [21]. The disparity maps computed show that our framework achieves good results (see Fig. 5).

The algorithm was evaluated according to gross errors. A gross error occurs when the disparity computed differs from the ground truth from more than one unit. The gross errors of the disparity maps computed by our algorithm are reasonable (see Fig. 6)

The errors are classified according to three regions of interest: all pixels, untextured and discontinuity regions. The next table shows the percentage of gross errors in these regions for each stereo pair. Our algorithm performs well since overall gross errors range from 0.71% (map) to 3.85% in a real image (tsukuba). Therefore, according to Table 2.7,



(a) map. (b) venus.

(c) sawtooth. (d) tsukuba.

Figure 6: Gross errors of the disparity maps.

our approach outperforms dynamic programming, while it compares to area-based, Bayesian and cooperative techniques. However, graph cut based methods are consistently superior concerning correctness.

| Image | All | Untextured | Discontinuity |
|---|---|---|---|
| tsukuba | 3.85% | 4.76% | 12.53% |
| sawtooth | 1.94% | 2.08% | 9.26% |
| venus | 2.12% | 3.31% | 18.33% |
| map | 0.71% | | 5.39% |

Table 2: Numerical gross errors by interest region.

# 6. Conclusions

The contributions of this paper include a new approach to solve the dense stereo matching problem based on a path computation in disparity space. A procedure which improves disparity maps is presented as a post-processing step for any technique solving a dense stereo matching problem.

An original survey addressing time-space complexity covers several stereo matching algorithms. In this context, our algorithm proved to have optimal $O(n)$ time-space complexity. The algorithm is faster than "real-time" techniques while producing comparable results. The effectiveness/correctness of our algorithm is demonstrated by experiments in real and synthetic benchmark data where errors range from 0.71% to 3.85%.

As future work, a generalization of our technique to multi-view stereo matching is trivial. The algorithm will have only to consider multiple potential occlusions between different pairs of images and a local step will involve $2^i$ possibilities, where $i$ is the number of images considered for matching. The use of our algorithm in the motion field computation is also feasible when the ordering constraint is generalized and linearization is applied in the 2D disparity space.

# References


[1] http://www.cs.cornell.edu/People/vnk/software.html.

[2] http://www.middlebury.edu/stereo.

[3] http://people.cs.uchicago.edu/~pff/bp/.

[4] R. Arnold. *Automated stereo perception*. Ph.D. thesis, 1983. Memo AIM-351, Thesis.

[5] S. Birchfield and C. Tomasi. Depth discontinuities by pixel-to-pixel stereo. In *Proceedings of the International Conference on Computer Vision*, pages 1073–1080, 1998.

[6] S. Birchfield and C. Tomasi. Multiway cut for stereo and motion with slanted surfaces. In *Proceedings of the International Conference on Computer Vision*, pages 489–495, 1999.

[7] A. Bobick and S. Intille. Large occlusion stereo. *International Journal of Computer Vision*, 33(3):181–200, 1999.

[8] Y. Boykov, O. Veksler, and R. Zabih. Markov random fields with efficient approximations. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 648–655, 1998.

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

[10] P. Felzenszwalb and D. Huttenlocher. Efficient belief propagation for early vision. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 261–268, 2004.

[11] P. Gill and M. Leonard. Reduced-Hessian quasi-Newton methods for unconstrained optimization. *SIAM Journal on Optimization*, 12(1):209–237, 2001.

[12] A. Goldberg and R. Tarjan. A new approach to the maximum-flow problem. *Journal of the ACM*, 35(4):921–940, 1988.

[13] H. Hirschmüller. Improvements in real-time correlation-based stereo vision. In *Proceedings of the IEEE Workshop on Stereo and Multi-Baseline Vision*, pages 141–148, 2001.

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

[15] V. Kolmogorov and R. Zabih. Multi-camera scene reconstruction via graph cuts. In *Proceedings of the European Conference on Computer Vision*, pages 82–96, 2002.

[16] S. Lee, Y. Kanatsugu, and J.-I. Park. Hierarchical stochastic diffusion for disparity estimation. In *Proceedings of the IEEE Workshop on Stereo and Multi-Baseline Vision*, pages 111–120, 2001.

[17] M. Lin and C. Tomasi. Surfaces with occlusions from layered stereo. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 710–717, 2003.

[18] H. Mayer. Analysis of means to improve cooperative disparity estimation. In *Proceedings of the ISPRS Conference on Photogrammetric Image Analysis*, 2003.

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

[20] D. Scharstein and R. Szeliski. Stereo matching with non-linear diffusion. *International Journal of Computer Vision*, 28(2):155–174, 1998.

[21] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1-3):7–42, 2002.

[22] J. Shi and C. Tomasi. Good features to track. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[23] C. Sun. Multi-resolution stereo matching using maximum-surface techniques. In *Proceedings of the Digital Image Computing: Techniques and Applications*, pages 195–200, 1999.

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

[25] H. Tao, H. Sawhney, and R. Kumar. A global matching framework for stereo computation. In *Proceedings of the International Conference on Computer Vision*, pages 532–539, 2001.

[26] O. Veksler. *Efficient graph-based energy minimization methods in computer vision*. Ph.D. thesis, Department of Computer Science, Cornell University, 1999.

[27] O. Veksler. Stereo correspondence with compact windows via minimum ratio cycle. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(12):1654–1660, 2002.

[28] O. Veksler. Fast variable window for stereo correspondence using integral images. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 556–561, 2003.

[29] C. Zitnick and T. Kanade. A cooperative algorithm for stereo matching and occlusion detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(7):675–684, 2000.