International Journal of Computer Vision 47(1/2/3), 275–285, 2002

A Hierarchical Symmetric Stereo Algorithm Using Dynamic Programming

G. VAN MEERBERGEN, M. VERGAUWEN, M. POLLEFEYS AND L. VAN GOOL
ESAT-PSI, K.U. Leuven, B-3001 Leuven, Belgium

Abstract. In this paper, a new hierarchical stereo algorithm is presented. The algorithm matches individual pixels in corresponding scanlines by minimizing a cost function. Several cost functions are compared. The algorithm achieves a tremendous gain in speed and memory requirements by implementing it hierarchically. The images are downsampled an optimal number of times and the disparity map of a lower level is used as 'offset' disparity map at a higher level. An important contribution consists of the complexity analysis of the algorithm. It is shown that this complexity is independent of the disparityrange. This result is also used to determine the optimal number of downsample levels. This speed gain results in the ability to use more complex (compute intensive) cost functions that deliver high quality disparity maps. Another advantage of this algorithm is that cost functions can be chosen independent of the optimisation algorithm. The algorithm in this paper is symmetric, i.e. exactly the same matches are found if left and right image are swapped. Finally, the algorithm was carefully implemented so that a minimal amount of memory is used. It has proven its efficiency on large images with a high disparity range as well as its quality. Examples are given in this paper.

Keywords: stereo vision, hierarchical stereo algorithm, dynamic programming, disparity computation, symmetric stereo

1. Introduction

The goal of this work is to compute fast accurate disparity maps, even for big images with large disparity ranges. It is assumed that these images are rectified first. The importance of the disparity maps for good 3D reconstruction is well known. The results of research on this topic include a wide variety of algorithms (Dhond and Aggarwal, 1989). Most of these algorithms are minimizing a cost function. The cost has 2 aspects: A first component decides how well a pixel of the left image corresponds to a pixel at the right image, i.e. the dissimilarity between 2 pixels. A second, respectively third component are related to the continuity in the horizontal, respectively vertical dimension of the disparity map. The latter is only dominant if no strong image data are available.

A possible criterion to classify these algorithms is the dimension of the optimization problem. The cost function decides how well a pixel of the left image corresponds to a pixel of the right image (zero dimensional cost function) (Koch, 1996). More robust is to optimize the sum of all costs of all matches in one scanline (one dimensional). The best is to optimize the sum of all costs over the whole image (two dimensional) (Roy and Cox, 1998). This algorithm, like many other algorithms (Cox et al., 1996; Birchfield and Tomasi, 1999), has chosen the second method, because it seems the best tradeoff between robustness and computational complexity. However, we will take into account information of the other scanlines by propagating information between them.

Our stereo algorithm has a number of specific properties. The algorithm is based on the hierarchical solution of the correspondence problem. Using this approach, we will show that the complexity decreases from $O(W \cdot H \cdot \delta_{range}^2)$ to $O(W \cdot H)$, where W and H represent respectively the width and height of the pictures. Note that the complexity becomes independent of the disparity search range δ_{range}. The algorithm offers a significant advantage over other algorithms, especially when this search range is large (Koch, 1996; Roy and Cox, 1998).

Another advantage of this hierarchical approach is that the memory usage can be greatly reduced. The acceleration discussed above does not lead to an explosion of memory usage. As will be explained in the following paragraphs, rather the contrary is true. Considering that today the bottleneck lies in the memory bandwidth and the cache sizes, this is very important in implementing the algorithm. Even for large images (1500×1500), it uses only a 100 kbyte, so that it fits easily into Pentium memory caches.

This algorithm is a candidate for real time implementation, as it offers a good tradeoff between speed and quality. Because of the gain in speed, it was possible to use more complex and computation intensive cost functions. Secondly, because of decoupling between algorithm and cost function, several cost functions could be tested.

The paper is structured as follows. First a formal problem definition is given. In Section 3, a tree search algorithm that has been used as a starting point for this work is given. In the next section, an efficient hierarchical implementation is developed, followed by its complexity analysis in Section 5. In Section 6, the symmetry of the algorithm is discussed and the algorithm is slightly modified to provide this symmetry. In Section 7, a comparison is made with known solutions from litterature. Finally, results and conclusions are presented.

2. Problem Definition

In this section, we first introduce some important definitions that are used troughout this paper. Then, we discuss the cost function and go more in detail about one of its components, the dissimilarity function. We end up with a formal problem definition.

2.1. Match Sequence

The correspondence problem, that we are trying to solve, searches for matches between left and right image. A match is noted $M = (M_{xL}, M_{xR})$, M_{xL} is the index of the pixel in the left image and M_{xR} is the pixel index in the right image. The disparity δ of a match must lie between 2 predefined constants, as postulated by constraint $C1$:

$$C1 : \delta_{min} \leq \delta = xL - xR \leq \delta_{max} \tag{1}$$

We define $\delta_{range} = \delta_{max} - \delta_{min} + 1$.

An ordered set of matches $\{M_i\}_{i=1..N_m}$ that obeys constraints $C2..C4$, is called a match sequence *MS*.

$$\begin{aligned} C2 &: M_{i,xL} < M_{j,xL} \;\&\; M_{i,xR} < M_{j,xR} \\ &\qquad\qquad \text{where } 0 \leq i < j < N_m - 1 \\ C3 &: M_{i+1,xL} = M_{i,xL} + 1 \vee M_{i+1,xR} < M_{i,xR} + 1 \\ &\qquad\qquad \text{where } i = 0..N_m - 2 \\ C4 &: M_{0,xL} = 0 \vee M_{0,xR} = 0 \end{aligned} \tag{2}$$

It is important to have a closer look at these constraints. The first constraint $C2$ expresses that no pixel can ever belong to 2 matches in the same match sequence and an ordening constraint is imposed. Constraint $C3$ imposes that the match sequence may not contain gaps. If there is a match M, then either the next pixel in the left, or the next pixel in the right scanline must be matched to the match following M. We say that the match sequence is complete if also the last pixel of left and/or right scanline appears in the (last) match. In Fig. 1, an example is given to clarify these definitions. The match sequence used in the example is:

$$\begin{aligned} MS = \{&(2, 0), (3, 1), (4, 2), (5, 3), (6, 4), (7, 6), \\ &(8, 7), (9, 8), (12, 9), (13, 10), (14, 11), \\ &(15, 12)\} \end{aligned} \tag{3}$$

Instead of coding a match sequence as given in the example above, one can as well just code the disparities in what we call a δ*-sequence*. The disparity map consists of nothing else than all δ*-sequences*. The δ*-sequence* that corresponds with the example above is:[1]

$$\begin{aligned} DS = \{&Undef, Undef, 2, 2, 2, 2, 2, \\ &1, 1, 1, \underline{2}, \underline{2}, 3, 3, 3, 3\} \end{aligned} \tag{4}$$

2.2. Cost Function

Once we defined a match sequence, a cost can be associated to it. As said in the introduction, the cost function

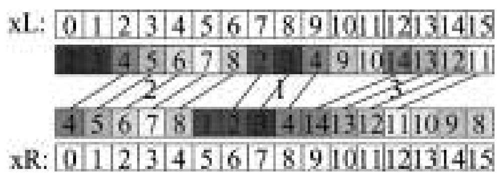

Figure 1. Graphical representation of a match sequence with a left and a right occlusion. The left scanline is on top. The corresponding disparities are also indicated.

consists of 3 components (The intensity of the images is represented by I.):

$$\begin{aligned}\kappa(DS) = &\sum_{xL=0}^{W-1} (D(xL, xR)|_{match}) \\ &+ \sum_{xL=0}^{W-1} (\beta|_{occl}) + \sum_{xR=0}^{W-1} (\beta|_{occl}) \\ &+ \sum_{xL=0}^{W-1} (\alpha |DS(xL) - DS_{-1}(xL)|\, IG^Y(xL))\end{aligned} \tag{5}$$

- For each unoccluded pixel, a cost that equals the dissimilarity between 2 pixels is taken into account. This dissimilarity function is discussed below in detail.
- If the pixel is occluded, a cost equal to β is counted. In contrast to other algorithms in the literature (Birchfield, 1999), larger occlusions are (linearly) more severely penalized. Left occlusions are taken into account in the second summation, while right occlusions are in the third summation.
- The last term takes into account the vertical dimension in the image. If this δ-*sequence* differs a lot from the scanline above, its penalty will be higher. This term is also scaled by an intensity gradient factor IG^Y that decreases this penalty if there is a large vertical intensity gradient near the pixel under consideration. This is an inter-scanline penalty with far lower complexity than described in previous work (Okutomi and Kanade, 1992), but with comparable performance. The intensity gradients $\frac{\partial I(xL)}{\partial y}$ are easily calculated using the Sobel operator. The actual function used is:

$$IG^Y(xL) = \max\left(1, \frac{a^2}{64 + S_Y(xL)}\right) \tag{6}$$

where $S_Y(xL) = Sobel(xL)$

The parameter a is scaled such that a Sobel output of 70 gives $IG^Y = 1$. Outputs above 1 are clipped.

2.3. *Dissimilarity Function*

The dissimilarity function indicates how well (the neighbourhood of) the pixel at position *xL* matches with pixel *xR*. It is the most dominant term in the cost function. Three dissimilarity functions are discussed. Each time, the dimension of the neighbourhood is considered, because it is directly related to the robustness of the dissimilarity function.

1. The easiest dissimilarity function is to take the absolute value of difference in intensities.

$$D(xL, xR) = |I(xL) - I(xR)| \tag{7}$$

The advantages are its simplicity and speed. Its practical use is very limited because it is not robust (0 dimensional). To make it more robust, this value can be calculated over a 2 dimensional window. It is named Sum of Absolute Differences (SAD). The Sum of the Square Differences can also be calculated (SSD).

2. A second alternative is a more complex dissimilarity function that implements a 2 dimensional cross correlation between the neighbourhoods of the pixels under consideration (Falkenhagen, 1994). It is implemented according to following formula:

$$D(xL, xR) = \frac{\sum_{i=-by}^{by}\sum_{j=-bx}^{bx} L_{ij} R_{ij}}{\sqrt{\sum_{i=-by}^{by}\sum_{j=-bx}^{bx} L_{ij}^2 \sum_{i=-by}^{by}\sum_{j=-bx}^{bx} R_{ij}^2}}$$

where

$$\begin{aligned} L_{ij} &= (I_L(y+i, xL+j) - \bar{I}_L) \\ R_{ij} &= (I_L(y+i, xR+j) - \bar{I}_R) \\ \bar{I}_L &= \frac{\sum_{i=-by}^{by}\sum_{j=-bx}^{bx} I_L(y+i, xL+j)}{(2.bx+1)(2.by+1)} \\ \bar{I}_R &= \frac{\sum_{i=-by}^{by}\sum_{j=-bx}^{bx} I_R(y+i, xL+j)}{(2\cdot bx+1)(2\cdot by+1)} \end{aligned} \tag{8}$$

This is the most robust dissimilarity function discussed in this paper. Because of the normalization in the denominator, it can deal with zone differences between left and right images, such as global intensity variations. The biggest disadvantage is its computational cost. Attempts to minimize the number of operations, result in an explosion of memory usage. With nowaday's computers, we should definitely avoid this.

3. The last dissimilarity function presented here is the one described by Birchfield and Tomasi (1998). This dissimilarity function is insensitive to image sampling. The latter phenomenon can significantly change the intensity value of a pixel where the intensity function is changing rapidly. Most stereo algorithms (Intille and Bobick, 1994; Cox et al.,

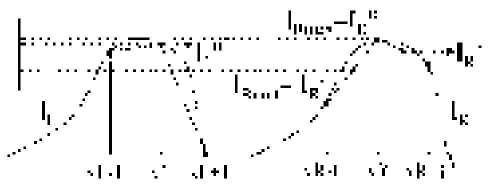

Figure 2. Birchfield's dissimilarity measure: $I_{R\,min} \leq I_L^0 \leq I_{R\,max}$, thus D_1 equals 0.

1996) just work at pixel resolution, so it is important to use a dissimilarity function that eliminates errors due to sampling. It is illustrated in Fig. 2. First we calculate I_R^- and I_R^+, the linearly interpolated intensities halfway between xR and its neighbours. Actually, Birchfield's dissimilarity measure is looking for the minimum distance between $I_L^0 = I_L(xL)$ and the linearly interpolated intensity curve in an interval $[-\frac{1}{2}, \frac{1}{2}]$ around xR. Indeed, considering the maximum and minimum of the intensities I_R^-, I_R^+, $I_R^0 = I_R(xR)$, an intensity band $[I_{R\ min}, I_{R\ max}]$ is formed. The dissimilarity measure $D(xL, xR)$ equals the distance between I_L^0 and the nearest boundary of this band. If I_L^0 is found in the band, as illustrated in Fig. 2, $D(xL, xR)$ equals 0. The same procedure is repeated in which the left image is interpolated to obtain a symmetric dissimilarity measure. This dissimilarity measure performs very well, and will be used indeed in our algorithm, unless extra robustness is needed. It needs only 3 additions per pixel in an efficient implementation. Further details can be found in Birchfield's paper (Birchfield and Tomasi, 1998).

2.4. Problem Definition

Now, all the material is ready for a formal problem definition: Find between all match sequences that obey constraints $C1..C4$, the sequence with the lowest cost, according to the cost function defined above. The cost of a scanline can be calculated incrementally, i.e. one can reuse the result if one adds a match to the match sequence. In the next section, it is shown how this sequence is found, making use of dynamic programming.

3. Tree Search Algorithm

In this section, we describe a first solution, using a tree representation in which the best path is found making use of dynamic programming (Ohta and Kanade, 1985a). The matrix structure that is used to solve this optimization problem is also covered in this section.

3.1. Stereo Tree

To describe the stereo tree, we define the nodes and edges. Each node is just a valid match, obeying constraint $C1$. An edge is defined by the constraints of a match sequence: between every 2 nodes that comply with constraints $C2$ and $C3$ exists an edge. It is not difficult to see that a kind of network is built between all possible matches that include the first pixel (beginnodes) and all matches that have the last pixel of a scanline (endnode). The construction of the tree is best explained with an example (The 2 scanlines used are the same as in the introduction). Figure 3 shows such a tree, with the matches as nodes in the shaded bullets. The beginnodes are found in the upper row and the left column. Also shown are the successors of match (0,0) and match (1,0). Also remark that the nodes (xL, xR) are placed in a special configuration, namely the matrix with horizontal xL and vertical xR axis. This way, a kind of bandmatrix from upper left to bottom right corner is formed. This is the subject of the next paragraph.

3.2. Matrix

The implementation uses 4 matrices. First it is explained using indices xL, xR, as in Fig. 3. Afterwards, it will be explained how a transformation of xL, xR is applied yielding Fig. 4.

- Every node (xL, xR) has its own matchcost, calculated by the dissimilarity function. Because this function can be quite expensive from computational

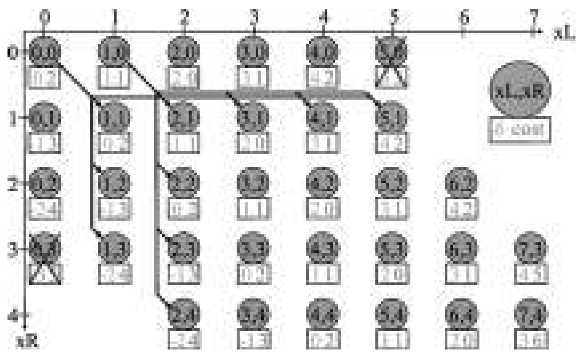

Figure 3. Stereo tree with the successors of match (0,0) and match (1,0).

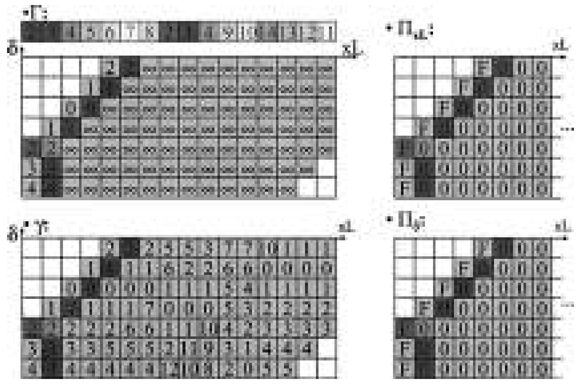

Figure 4. The 4 compressed matrices Π_{xL}, Π_{δ}, γ and Γ. The filling has just started.

point of view, it is calculated before and stored in matrix γ.

- Every node has one and only one predecessor, by the definition of the tree structure. This predecessor can be unambiguously characterized by its (xL_p, xR_p) numbers. That's why we need 2 other matrices Π_{xL} and Π_{xR} that store xL_p, respectively xR_p of each node.
- Because each node has only one predecessor, this node also characterizes the path from one of the beginnodes to that node. The path cost of that path is stored in a 4th matrix Γ, thus $\Gamma(xL, xR) = \kappa(MS)$.

The matrices are filled top-down, the endnode with the lowest cost (Γ) is selected and trivial backtracking is performed to find the best path.

To be more memory efficient, only the diagonal band of the matrices is stored. In fact, this is equivalent to a transformation by which the *xR* axis is transformed into the δ axis. This way, implementation is far more efficient and the *δ-sequence* can be directly read out from the Γ matrix. The 4 compressed matrices are shown in Fig. 4, when the filling has just started. Note the special pattern by which the successors of node (0,0) can be found.

4. Hierarchical Implementation

The complexity of this algorithm is $O(W \cdot H \cdot \delta^2_{range})$, as will be shown in the next section. In this section, we develop a hierarchical implementation of the above algorithm, which will be shown to have a complexity independent of δ_{range}. We first need to introduce the concept of an *offset δ-sequence*.

4.1. Offset Stereo Tree

An offset stereo tree is a normal stereo tree, but the number of allowed nodes is limited by an extra constraint:

$$C5 : \delta_{s\,min} \leq \delta_s \leq \delta_{s\,max} \text{ where } \delta_s = \delta - DS_{\Delta}(xL) \tag{9}$$

This means that the nodes that are allowed must have a disparity in the interval $[\delta_{s\,min} + DS_{\Delta}(xL), \delta_{s\,max} + DS_{\Delta}(xL)]$, around a given offset disparity. This offset disparity sequence $DS_{\Delta}(xL)$ is given. How it is calculated is subject of the next paragraph.

4.2. Hierarchical Calculation of the Disparity Map

As already suggested in the introduction, this offset disparity sequence is the result of the disparity calculation at a higher (downsampled) level. We assume in this paper that downsampling is always done by a factor 2 in 2 dimensions. An example of one scanline is given in Fig. 5, where the scanlines of the same example are used again. Both of them are downsampled by which the disparity search range halves. Then, our algorithm is executed with an initial offset disparity map, which is chosen all-zero.[2] The result (in our example, all ones, but this is coincidence) is upsampled afterwards and multiplied by 2. This *δ-sequence* is then used as an *offset δ-sequence* in our next run at the original level in a $\delta_{s\,range} = [-1, 1]$. From next section, it will be clear that executing the algorithm twice with a small $\delta_{s\,range}$ is cheaper than one 'big' search in the full δ_{range}.

Next, the memory requirements for the 4 matrices can be reduced further by using the relative δ_s

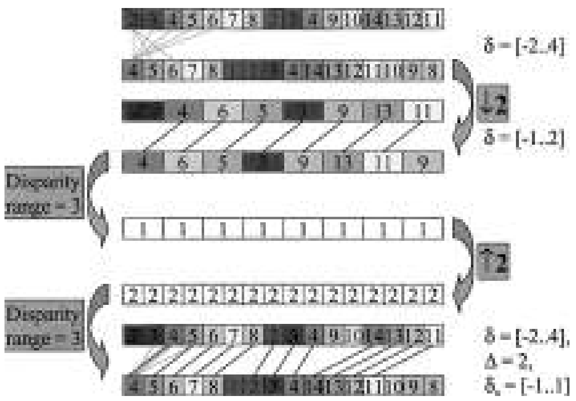

Figure 5. Overview of the hierarchical stereo calculation.

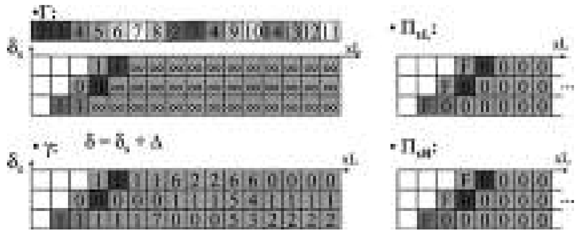

Figure 6. The 4 reduced matrices used in the hierarchical stereo calculation.

axis instead of the absolute disparity δ. The algorithm is programmed in such way that the *offset* DS_Δ sequence is read from the disparity map, used to calculate an updated (refined) map using our 4 matrices and rewritten in the same disparity map, after being interpolated. The 4 matrices are shown in Fig. 6.

One question remains unanswered: What's the largest refinement that's allowed, i.e. what's the choice of $\delta_{s\,min}$, $\delta_{s\,max}$? As already suggested in Falkenhagen (1997), this depends on the disparity estimation error in each level. Experiments have shown that this error is below 1.5 pixel. In the previous upsampled level, this error corresponds to 3 pixels. That's why we choose the constants $\delta_{s\,min} = -3$ and $\delta_{s\,max} = 3$. Disparity errors up to 3 pixels will be corrected in a lower level. As a consequence $\delta_{s\,range} = 7$.

5. Complexity Analysis

5.1. Reduction of Complexity

As already discussed in Birchield's thesis (1999), the complexity Θ of the non-hierarchical algorithm is proportional to the square of the disparity search range. L represents the number of times downsampling is applied. ($L = 0$ represents the non-hierarchical algorithm.)

$$\Theta(L = 0) = O\big(WH\delta^2_{range}\big) \tag{10}$$

Downsampling once ($L = 1$) reduces the overall complexity to:[3]

$$\Theta(L = 1) = O\left(WH\delta^2_{srange} + \frac{W}{2}\frac{H}{2}\delta^{(1)2}_{range}\right) \tag{11}$$

where $\delta^{(1)}_{range} = \frac{\delta_{range} - \delta_{s\,range}}{2} + 1$

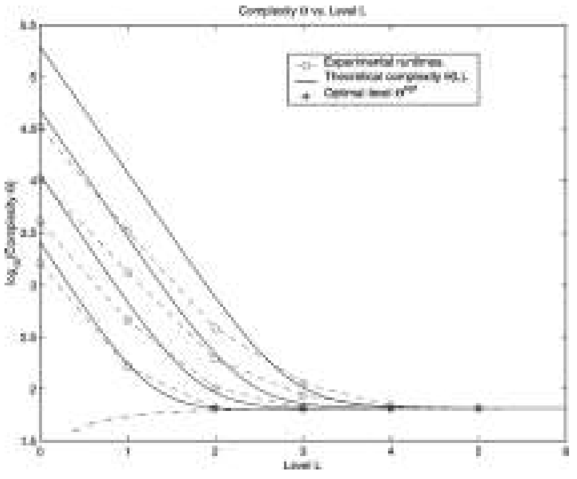

Figure 7. Logarithmic plot of the complexity Θ versus L.

An analytical expression for the complexity $\Theta(L)$, function of L, can be found. It reaches a unique global minimum for $L = L^{opt}$. This optimal number of levels is only a function of the initial disparity search range δ_{range}. The complexity at this optimal level is

$$\Theta(L = L^{opt}) = O\left(WH\left(65 - \frac{4071}{(\delta_{range} + 5)^2}\right)\right) \tag{12}$$

So, if δ_{range} is large enough, we claim that the complexity is independent of the disparity search range. This is illustrated in Fig. 7 which shows a logarithmic plot of the complexity Θ versus L. The tests are performed on large images: 1404×1092. The figure compares the calculated complexity with experimentally measured runtimes for 4 values of the disparity range $\delta_{range} = 51, 107, 219, 443$. As we can see, the theoretical curves (solid without bullets) decrease exponentially towards an almost constant complexity, where they reach a unique minimum. Practical results (dashed with bullets) show the same tendency. All these minima for a varying δ_{range} parameter, as described by Eq. (12), are connected by the solid line, that reaches asymptotically $WH\log_{10}(65)$.

We end this section with a practical implementation for calculating the optimal level of downsampling. The real formula, a result of the analytical optimization problem ($\delta_{s\,range} = 7$), is

$$L^{opt} = \log_2\left(\frac{(\delta_{range} + 5)(\sqrt{4924} - 45)}{242}\right) \tag{13}$$

A good approximation of this curve is given by:

$$L^{opt} \approx \log_2\left(\frac{(\delta_{range} - 2 + \delta_{s\,range})}{2(\delta_{s\,range} - 1)}\right)$$
$$= \log_2\left(\frac{(\delta_{range} + 5)}{12}\right) \qquad (14)$$

This is nothing else than the result of the solution of $\delta^{(L)}_{range} = \delta_{s\,range} = 7$. So we keep on downsampling as long as δ_{range} is greater than $\delta_{s\,range}$.

6. Symmetry

Recently, the focus of our work has shifted towards the symmetry of the algorithm. Symmetry means that the same match sequence is found if the left and the right image are swapped. If we consider Fig. 1 again and if we swap the images, would the same match sequence still be found? To have a symmetric algorithm, 3 conditions have to be fulfilled. First, the costfunction and the corresponding dissimilarity function must be symmetric. The dissimilarity functions that are presented in this paper are symmetric. The costfunction of Eq. (16) is assymetric due to the last term. A modified symmetric costfunction is given in the following equation.

$$\kappa(DS) = \sum_{xL=0}^{W-1}(D(xL, xR)|_{match}) + \sum_{xL=0}^{W-1}(\beta|_{occl})$$
$$+ \sum_{xR=0}^{W-1}(\beta|_{occl}) + \sum_{xL=0}^{W-1}(\alpha|DS(xL)$$
$$- DS_{-1}(xL)|\, IG^Y(xL) IG^Y(xR)) \qquad (15)$$

Secondly, the same nodes must be considered in the search tree. Figure 8 shows that this is not the case for the algorithm described before. In this figure, the search tree is shown near a left occlusion. In the right image, when the left and right image are swapped, the matches are indicated in gray. In the left image, the same matches as in the right image are considered (gray), but there are also additional ones (dark gray). This is because of the vertical direction of refinement. To avoid this, the boundary given by constraint $C5$ (Eq. (9)) must be more tight near an occlusion. Current work consists in swapping this direction of refinement in the presence of occlusions.

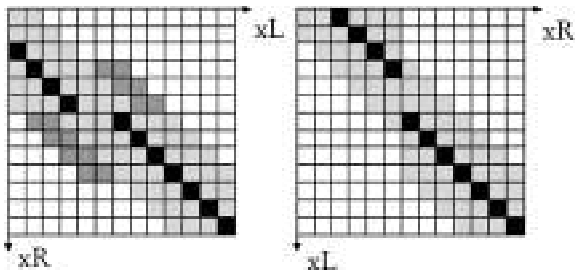

Figure 8. Symmetric nodes must be considered.

Third, the same search tree must be built during execution of the algorithm. This means that in case of an ambiguity (2 paths with the same cost), the same path must be taken, independently of which image is considered first. This means that at every stage in the search algorithm, we will make a hard decision. Nevertheless, the existence of an ambiguity provides us with extra information regarding the reliability of that match. We will call this soft information, in contradiction with the hard decision that must be taken.

7. Comparison

It is instructive to compare our stereo algorithm with other stereo algorithms. First, it is compared with the algorithm of Koch (1996) and Falkenhagen (1994). They developed an algorithm that searches for correspondences by matching blocks (for instance NCC). Local boundary conditions are taken into account, but no global cost function is minimized. Interpolation of disparities is non-existent. To increase the robustness of the local optimisation problem, and to reduce the corona effect, one estimates the disparity in a large region (large robust block comparison), after which these disparities are refined with smaller blocks. Although this kind of hierarchy is comparable with the one used in our algorithm, it arises from a fundamentally different reason, i.e. a lack of robustness of the local optimisation problem. It is important to see that the acceleration of our algorithm, under certain conditions proven by Birchfield (Birchfield and Tomasi, 1998), has no effect on the outcome! The complexity of the algorithm described in Falkenhagen (1994) is $O(W \cdot H \cdot \delta^2_{range})$.

Secondly, our algorithm is compared with Birchfield's algorithm (Birchfield, 1999). A modified version is implemented in Intels' OpenCV. The core of this algorithm is very similar to our algorithm, but there is a major difference in how the acceleration is performed. Birchfield's fast algorithm (complexity $O(W \cdot H \cdot \delta_{range} \log(\delta_{range})))$ used a method to prune bad nodes in the search tree. The disadvantage is that the exact minimum (best path) is no longer guaranteed, and to achieve an acceptable minimum (path), extra constraints are put on the cost function. It can no longer be chosen independently of the algorithm, and Birchfield had to postpone several tasks to a postprocessing step. For instance, in Birchfield's algorithm, propagation of disparities between scanlines, intensity gradient information and processing occlusions are done during postprocessing. To make all this happen, a lot of postprocessing parameters are used without rules to tune them.

Finally, a comparison is made with the class of mincut-maxflow stereo algorithms (Roy and Cox, 1998; Boykov et al., 1999). We compared our algorithm with the implementation of Roy and Cox (1998). A complete 2D optimisationproblem is solved, making use of mincut-maxflow problem. The algorithm offers high quality disparity maps, comparable with the one obtained by our algorithm. The disadvantage is its complexity $O(W \cdot H \cdot \delta_{range}^2 \cdot \log(W\delta_{range}))$ and its memory use (more than 3.5 Gbyte for Mars images given in the Results section).

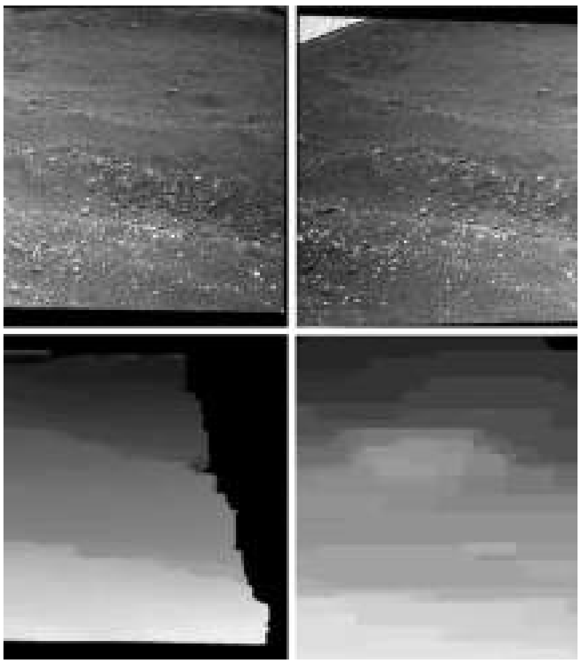
Figure 9. Mars images. Original left and right image are shown at the top. The disparity map is presented at the bottom left and a detail of the disparity map near a stone on Mars' surface is shown at the bottom right.

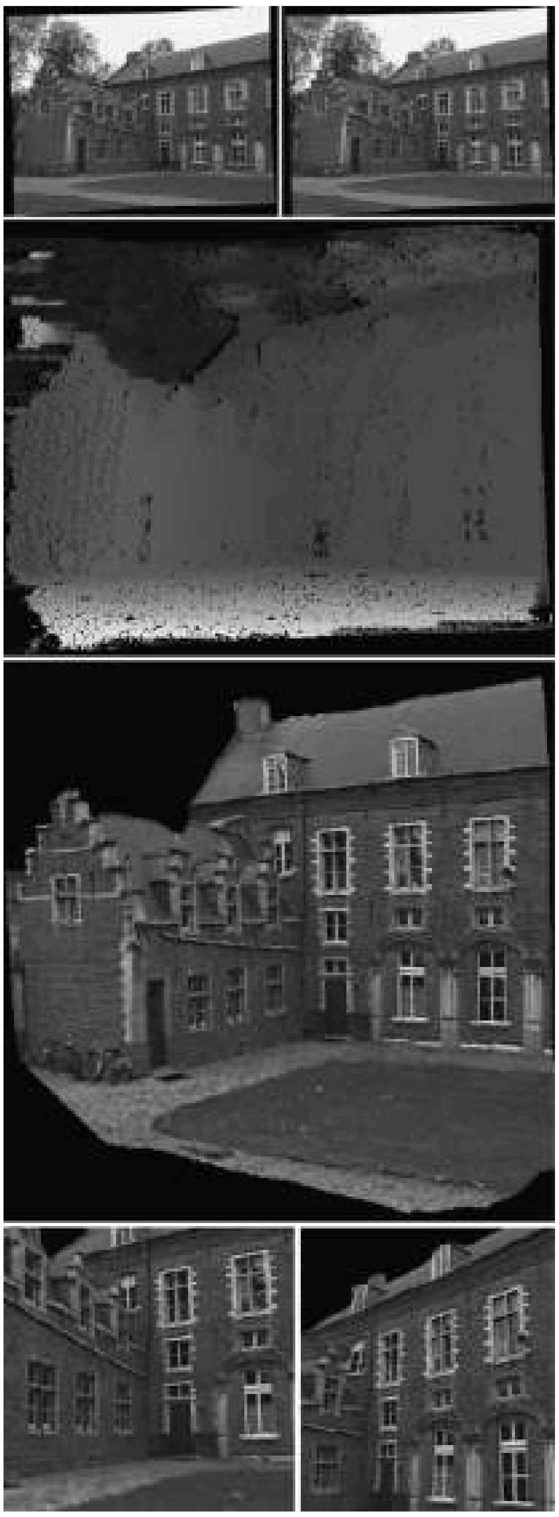
Figure 10. Images from the 'Arenbergkasteel' Original left and right image are shown at the top. The disparity map is presented in the middle together with a 3D model and 2 details of this model.

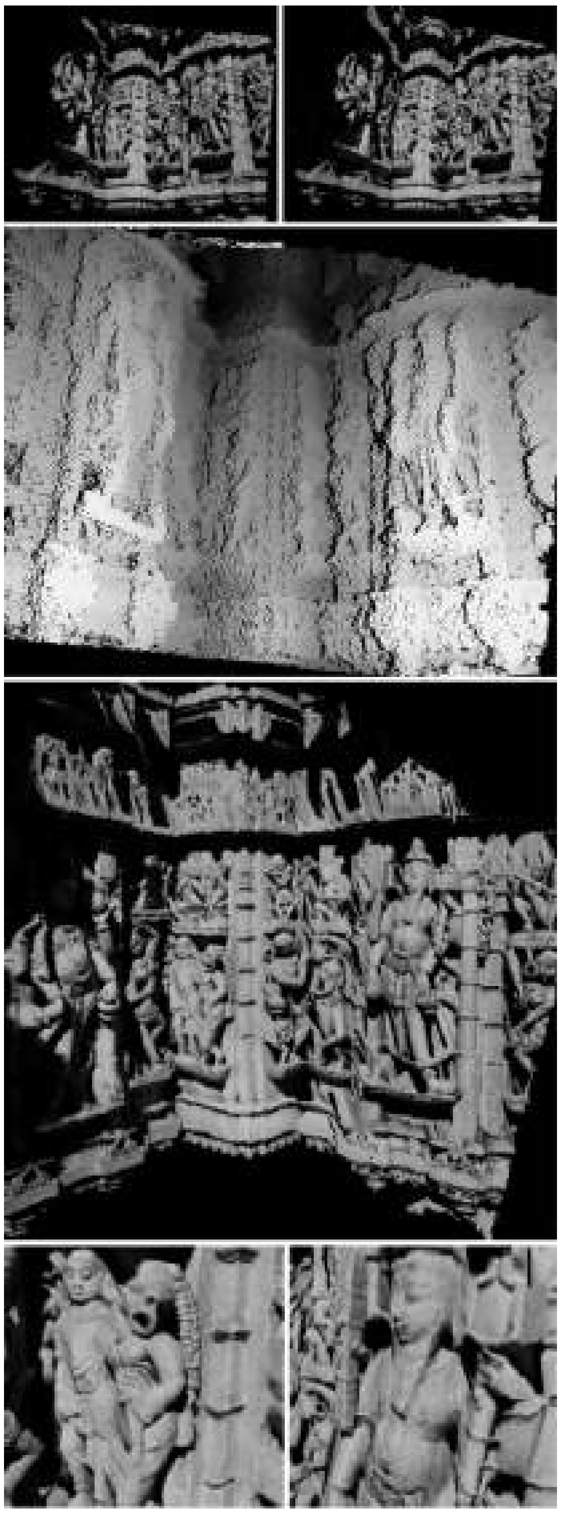

Figure 11. Images from an Indian temple Original left and right image are shown at the top. The disparity map is presented in the middle together with a 3D model and 2 details of this model.

8. Results

The first example is provided by ESA (European Space Agency). Figure 9 shows images of the testbed of Mars' surface. These are large images (1404 $\times$ 1092) with large disparity ranges (500 and more). This was the actual reason to search for a new algorithm which resulted in the hierarchical algorithm, since most existing algorithms failed. This is a good example to illustrate the memory needs and speed of our algorithm. The images were downsampled 4 times, and the total processing

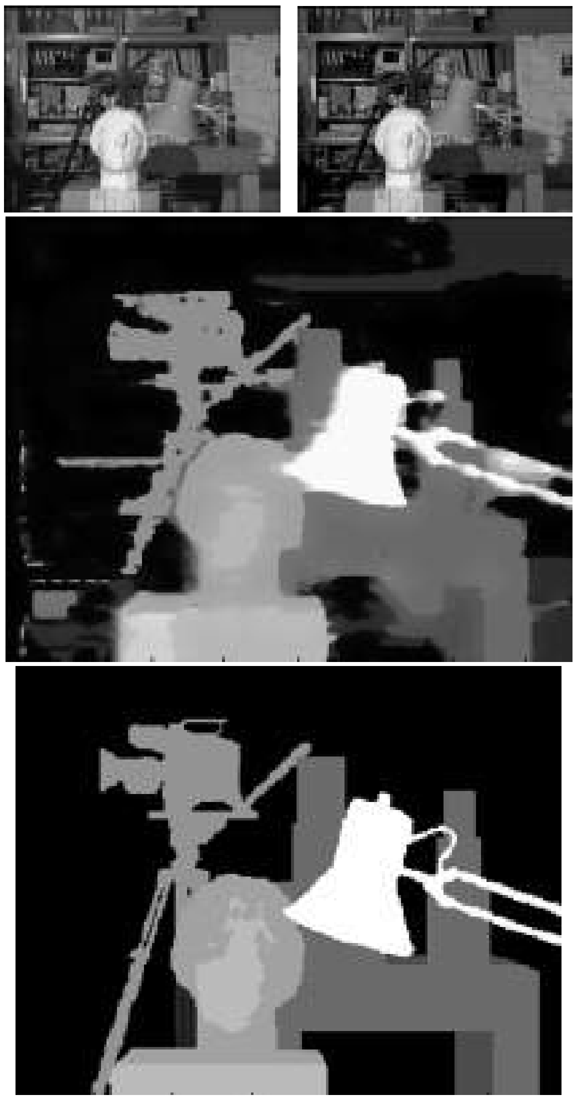

Figure 12. Tsukuba dataset for testing accuracy. Original left and right image are shown at the top. The disparity map is presented in the middle. The ground truth can be found at the bottom.

time did not exceed 70 seconds. The amount of memory used was 150 kbyte. Because of imperfections of the cameras, yielding an intensity profile in the images, the more expensive NCC dissimilarity function is used.

A second testcase includes a set of images of the Arenbergcastle in Leuven. These two images (792×635) can be found at the top of Fig. 10. It is a good example to show the accuracy of the disparitymap. There are lots of depth discontinuities and lots of details. The hierarchical algorithm achieves sharp edges (roof, crenels, ...) and a good overall quality. The total runtime is 16 seconds.[4]

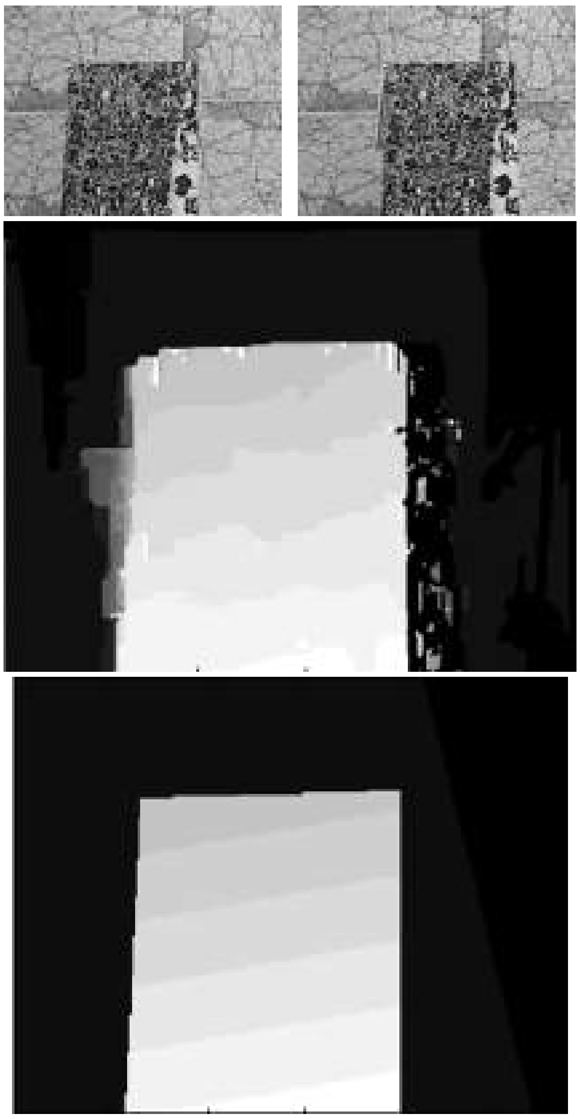
Figure 13. Slanted dataset for testing accuracy. Original left and right image are shown at the top. The disparity map is presented in the middle. The ground truth can be found at the bottom.

A third testcase includes a set of images of an Indian temple (Fig. 11). Lots of small occlusions and depth discontinuties are present in the images. Again, the accuracy of the disparitymap is very high. Many details (ornaments, faces, hair, ...) are clearly visible in the 3D model which shows also a good overall quality. The total runtime is 30 seconds.

The next image is provided by the University of Tsukuba as a dataset for testing accuracy. The depth map ground truth is also given in Fig. 12. A lot of details, including the camera in the background, are visible in the disparitymap. Because of the small images (384×288), downsampling was not needed during the execution, that took 4 seconds of processing time.

The last image is provided by the Microsoft Research, again as a dataset for testing accuracy. The depth map ground truth is also given in Fig. 13. The black spots in the disparitymap represent occlusions. In previous images, they were interpolated. Because of the large occlusion present in the image, this is not done here. The images (284×216) are downsampled only once during execution that took only 2 seconds of processing time.

9. Conclusion

The algorithm presented in this paper is based on the hierachical solution of the stereo correspondence problem, using subsampling. The complexity is $O(W \cdot H)$, which is independent of the disparityrange and lower than other classical stereo algorithms which have a complexity of $O(W \cdot H \cdot \delta_{range}^2)$. A careful implementation yields minimum memory usage, so that the computation fits in the cache of most modern processors. The disparity maps show a high quality, thanks to a new and more complex cost function. This cost function is completely decoupled from the optimisation problem, so it is easy to switch cost functions to make a tradeoff between quality and execution time.

Acknowledgment

Geert Van Meerbergen and Marc Pollefeys are respectively research assistant and post-doctoral fellow of the Fund for Scientific Research—Flanders (Belgium). The financial support of the FWO project G.0223.01 and the IUAP/P4/24 project are also gratefully acknowlegded.

Notes

1. The underlined disparities are interpolated. This is necessary for the hierarchical implementation.
2. If a raw disparity map is available, it can be used here as offset disparity map.
3. It suffices to search in a $\delta_{range}^{(1)}$ that is $\frac{\delta_{s\ range}}{2}$ smaller than $\frac{\delta_{range}}{2}$. This disparity range will be corrected (and enlarged) while up-sampling and refining the disparity map.
4. The tests were performed on an intel Pentium III workstation 700 MHz.

References

Birchfield, S. 1999. Depth and motion discontinuities. Ph.D. Thesis, Department of Electrical Engineering, Stanford University.

Birchfield, S. and Tomasi, C. 1998. A pixel dissimilarity measure that is insensitive to image sampling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(4):401–406.

Birchfield, S. and Tomasi, C. 1999. Depth and motion discontinuities. *International Journal of Computer Vision*, 35(3):269–293.

Boykov, Y., Veksler, O., and Zabih, R. 1999. Fast approximate energy minimization via graph cuts. In *Proc. Seventh International Conference on Computer Vision*, pp. 377–384.

Cox, I., Hingorani, S., and Rao, S. 1996. A maximum likelihood stereo algorithm. *Computer Vision and Image Understanding*, 63(3).

Dhond, U. and Aggarwal, J. 1989. Structure from stereo—A review. *IEEE Trans. Syst., Man and Cybern.*, 19:1489–1510.

Falkenhagen, L. 1994. Depth estimation from stereoscopic image pairs assuming piecwise continuous surfaces. In *Image Processing for Broadcast and Video Production*, Y. Paker and S. Wilbur (Eds.). Springer: Great Britain, pp. 115–127.

Falkenhagen, L. 1997. Hierarchical block-based disparity estimation considering neighbourhood constraints. In *Proceedings International Workshop on SNHC and 3D Imaging*, Rhodes, Greece, pp. 115–122.

Hartley, R., Gupta, R., and Chang, T. 1992. Stereo from uncalibrated cameras. In *Proc. Conference Computer Vision and Pattern Recognition*, pp. 761–764.

Intille, S.S. and Bobick, A.F. 1994. Disparity-space images and large occlusion stereo. In *Proc. Third European Conference Computer Vision*, pp. 179–186.

Koch, R. 1996. Automatische oberflachenmodellierung starrer dreidimensionaler Objekte aus stereoskopischen Rundum-Ansichten. Ph.D. Thesis, University of Hannover, Germany. Also published as Fortschritte-Berichte VDI, Reihe 10, Nr. 499, VDI Verlag, 1997.

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

Ohta, Y. and Kanade, T. 1985b. Stereo by intra- and inter-scanline search using dynamic programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-7(2):139–154.

Okutomi, M. and Kanade, T. 1992. A locally adaptive window for signal processing. *International Journal of Computer Vision*, 7:143–162.

Roy, S. and Cox, I. 1998. A maximum-flow formulation of the N-camera stereo correspondence problem. In *Proceedings of the International Conference on Computer Vision*, Bombay, India, 1998.