Real-Time Stereo Vision: Making More Out of Dynamic Programming

Jan Salmen[1], Marc Schlipsing[1], Johann Edelbrunner[1], Stefan Hegemann[2], and Stefan Lüke[2]

[1] Institut für Neuroinformatik, Ruhr-Universität Bochum, 44780 Bochum, Germany
{Jan.Salmen,Marc.Schlipsing,
Hannes.Edelbrunner}@neuroinformatik.rub.de
[2] Continental AG, Division Chassis & Safety, Germany
{stefan.hegemann,stefan.lueke}@continental-corporation.com

Abstract. Dynamic Programming (DP) is a popular and efficient method for calculating disparity maps from stereo images. It allows for meeting real-time constraints even on low-cost hardware. Therefore, it is frequently used in real-world applications, although more accurate algorithms exist. We present a refined DP stereo processing algorithm which is based on a standard implementation. However it is more flexible and shows increased performance. In particular, we introduce the idea of multi-path backtracking to exploit the information gained from DP more effectively. We show how to automatically tune all parameters of our approach offline by an evolutionary algorithm. The performance was assessed on benchmark data. The number of incorrect disparities was reduced by $40\,\%$ compared to the DP reference implementation while the overall complexity increased only slightly.

1 Introduction

Stereo vision's task is to estimate depth by calculating a *disparity map* for two input images. It has actively been investigated for decades and still is a vivid topic in computer vision – mainly driven by a wide range of possible fields of application like robotics, automotive systems, surveillance and augmented reality just to name a few.

Depending on the application, very different requirements and objectives, i.e. runtime and performance, are relevant. If there are only weak constraints on these issues, a look at the Middlebury Stereo site[1] established by Scharstein and Szeliski [1,2], which compares results for a lot of different approaches, allows for finding a state-of-the-art algorithm.

However, if problem specifications become more restrictive, e.g. in terms of real-time performance on limited hardware resources, there are much fewer options as requirements may include low memory usage, mainly linear memory access, few complex operations (like floating point instructions), and parallelizability. DP approaches are perfectly suitable in this context, providing all the properties listed while their performance is sufficient for most real-world applications.

[1] http://vision.middlebury.edu/stereo/

X. Jiang and N. Petkov (Eds.): CAIP 2009, LNCS 5702, pp. 1096–1103, 2009.

In this work we will focus on how to tap the full potential of the basic, pixel-wise DP algorithm itself. We modify it to be better parameterizable and perform automatic offline optimization of those parameters. Additionally, we introduce a new idea to exhaustively benefit from the information gained by the DP – namely through a multi-path backtracking.

This article is organized as follows. In the forthcoming section, we present related work on DP stereo methods. The algorithm we chose as reference is presented in more detail in section 3. In section 4 modifications to this algorithm proposed by us are presented. Section 5 describes how the parameters of the new approach are optimized for a given task using the Middlebury benchmark images as an example. Finally we discuss our results in section 6.

2 Related Work

DP was first used for *edge-based* approaches, for example by Ohta and Kanade [3]. Geiger et al. [4] were among the first to propose a DP stereo method based on *pixel-wise* intensity differences. Bobick and Intille [5] presented the algorithm whose implementation serves as reference for this study. It is described in detail in section 3.

In [6] the usage of *MMX*, Assembler code, and some other techniques allow for building a real-time DP based stereo system. No results can be found in the Middlebury benchmark, but the results reported for two of the four current test images show a slightly better performance than the reference implementation does.

In [7] Gong et al. introduce a so called *reliability-based* DP algorithm. Tracing more than just the best path after the optimization step they provide a reliability measure for each disparity. A similar technique will be applied in this work.

Kim et al. [8] present a DP based algorithm which identifies possible disparities for each pixel by comparing orientation filter responses. The first (horizontal) DP optimization step is then performed only taking into account those candidates. Costs caused by that optimization step are incorporated in an energy function which is finally optimized in vertical direction. Computation takes several seconds on the Middlebury benchmark images. The idea of identifying candidates and choose among them can be found in our approach, too, but it is conducted in a different manner.

In [9] adaptive cost aggregation in the vertical direction (considering color information) is used to improve the DP results. Real-time capability is achieved with a GPU-based implementation.

Veksler introduces a DP algorithm that works on a tree structure instead of image rows [10]. Her implementation takes less than $1\ s$ and performs almost $20\,\%$ better on the Middlebury benchmark data than the basic method.

3 Reference Implementation

As reference and baseline for comparison, we consider the algorithm ranked 49th in the Middlebury evaluation under the name of 'DP'. The source code is freely available as part of the *StereoMatcher* framework. Because the goal in [1] was to evaluate the optimization technique itself, the DP approach presented in [11] was applied, but without

shiftable windows and *ground control points*. In the course of this section the algorithm is described.

3.1 Cost Calculation

The matching costs d_{SAD} for two pixels P and Q are based on the *sum of absolute differences (SAD)* which is

$$d_{\text{SAD}}(P,Q) = d_R(P,Q) + d_G(P,Q) + d_B(P,Q) \tag{1}$$

for RGB images, where $d_R(P,Q) = |P_R - Q_R|$ with P_R as red component of pixel P etc. The cost for matching two pixels is interpolated in the range of a half pixel as proposed by Birchfield and Tomasi [5]. The result is the cost matrix C of size $X \times N$ for every image row, where X is the length of the row and N the number of allowed disparities.

3.2 Cost Aggregation

Values in the cost matrix are not aggregated. Thus, optimization is only based on the interpolated pixel-to-pixel intensity differences.

3.3 Optimization

The optimization is performed independently for each image row. A path with minimal total cost through the matrix C has to be found (cf. [11]). While the naive technique, i.e. calculating costs of all possible paths independently, would lead to a computational complexity of $O(N^X)$, DP solves this in $O(NX)$. The algorithm works in two phases: *forward step* and *backtracking*.

During the forward step, three possible states for each pixel and each disparity are managed simultaneously: M (matched), V (vertical occlusion) and D (diagonal occlusion). Pixels in matched state are assigned the corresponding cost from C, Pixels in states V or D are penalized by c_{occ}. Starting an occlusion is additionally penalized by $\rho_I(\Delta I)$ which depends on the actual intensity gradient ΔI and is realized as

$$\rho_I(\Delta I) = \begin{cases} c_{smooth} & \text{if } \Delta I < t_I \\ c_{smooth} \cdot p & \text{if } \Delta I \geq t_I. \end{cases} \tag{2}$$

The algorithm on the left side in figure 1 contains pseudo code for one part of the forward step and storage of the best transition leading to state M.

In the backtracking phase the minimal cost path is followed backwards along the stored transitions. At the same time, pixels in the result image are set to the corresponding disparity values or marked occluded.

3.4 Refinement

Pixels marked as occluded are filled from left to right. No sub-pixel refinement is performed.

4 Proposed Algorithm

Our implementation provides modifications of the reference algorithm. We add more flexibility and aim at utilizing the information gained from DP more efficiently.

4.1 Cost Calculation

We propose calculating a *weighted Euclidean distance* in the RGB-space

$$d_{\text{Euklid}}(P,Q) = \sqrt{w_R \cdot d_R^2(P,Q) + w_G \cdot d_G^2(P,Q) + w_B \cdot d_B^2(P,Q)}, \qquad (3)$$

where the three components are weighted individually by w_R, w_G, and w_B respectively.

The Birchfield cost measure is not applied because experiments in [1] did not show pay-off concerning efficiency.

4.2 Cost Aggregation

Slight aggregation in vertical direction, between neighboring scanlines, is performed. To realize that, we store the cost matrices for the current row and the two adjacent ones. A vertical Gaussian filter with weights $(1, 2, 1)$ is applied previous to optimization in order to calculate a smoothed match matrix.

4.3 Optimization

Extended parametrization. We introduce more parameters to the DP technique to allow for more flexibility: Different penalties for diagonal and vertical occlusions (c_D, c_V) and for starting these occlusions (p_D, p_V) are provided. Additionally, we introduce rewards r_D and r_V for transitions leading from occluded to matched state as counterpart to the penalties. As the penalty p_D and reward r_D e.g. would cancel each other out, this is only useful if all these costs are gradient-dependent. Therefore, for every parameter $\bullet$ named above, there is also a corresponding one used if $\Delta I \geq t_I$, marked as $\hat{\bullet}$.

Multi-path backtracking. It is noteworthy that a lot of information is generated during optimization, but most of it is disregarded in the classic approach: During the forward step only one transition per cell is chosen, not taking into account (almost) equally good solutions. For backtracking, just one possible path ending is considered.

We explicitly aim for incorporating more information for optimization. So, during the forward step of DP, *almost-optimal* transitions are stored. A transition is almost-optimal if its cost does not differ more than Δ_c from the best one. Figure 1 shows the resulting changes in pseudo code.

After the forward step, in the classic DP approach the only solution is found in the last column, corresponding to a path ending with minimal total costs m. We propose to additionally trace alternative path endings with costs $c \leq \tau \cdot m$ (with $\tau > 1$). Depending on the information from the forward step, more than one transition can be considered at every point during backtracking. All resulting possible paths through the DP array can be found by depth-first search, keeping track of already visited nodes in order to

```
cMM ← A(x − 1, d, M) + C(x, d);
cVM ← A(x − 1, d, V) + C(x, d);
cDM ← A(x − 1, d, D) + C(x, d);
cMin ← min(cMM, cVM, cDM);
A(x, d, M) ← cMin;
if cMM = cMin then
    optTrans(x, d) ← MM⃗;
else if cVM = cMin then
    optTrans(x, d) ← VM⃗;
else if cDM = cMin then
    optTrans(x, d) ← DM⃗;
```

```
cMM ← A(x − 1, d, M) + C(x, d);
cVM ← A(x − 1, d, V) + C(x, d) − r_D;
cDM ← A(x − 1, d, D) + C(x, d) − r_V;
cMin ← min(cMM, cVM, cDM);
A(x, d, M) ← cMin;
if cMM ≤ cMin + Δ_c then
    isPosTrans(x, d, MM⃗) ← true;
else
    isPosTrans(x, d, MM⃗) ← false;
if cVM ≤ cMin + Δ_c then
    isPosTrans(x, d, VM⃗) ← true;
else
    isPosTrans(x, d, VM⃗) ← false;
if cDM ≤ cMin + Δ_c then
    isPosTrans(x, d, DM⃗) ← true;
else
    isPosTrans(x, d, DM⃗) ← false;
```

Fig. 1. Treatment of state M for pixel at x and disparity d during forward step of DP in the reference implementation (left) and in our approach (right)

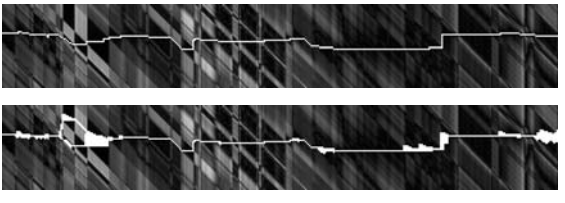

Fig. 2. Cost matrix for a single image row with path found by basic DP (top) and same matrix with results from multi-path backtracking (bottom)

avoid multiple processing. Figure 2 shows an example for optimization results in the reference implementation and in our approach.

If a pixel at (x, d) is traversed with state M, the disparity d is possible in this image row at position x and this information is stored, resulting in a sparse representation of candidate disparities for every pixel. As the assignment of disparities is not definite as in the reference implementation, an additional selection step is necessary.

Vertical optimization. To chose from the sparse number of candidate disparities, an additional optimization step is performed. The same idea as used for the horizontal optimization step is realized here: an assignment from the possible disparities in a column with minimal cost is chosen. If there was not found any possible disparity for a pixel at all during backtracking, every disparity is allowed. The cost function enforces

smoothness by penalizing neighbor disparities that differ by 1 with λ and disparities that differ more with μ. Again, this optimization problem can be solved with DP, but here, it is significantly less complex due to the smaller amount of possible solutions. Resulting disparity values are accepted if the corresponding matching cost is less than C_{max}. Otherwise, the pixel is marked as occluded.

4.4 Refinement

Occluded regions in the disparity image are filled in the same manner as in reference implementation. Furthermore, disparities for the left border of the reference image are not calculated, because not all matching costs can be computed there. This border region is filled from the right side.

5 Experiments

Along with our modifications and in favor of more flexibility the total number of parameters increase from four to 21, cf. table 1 for an overview. To tune the parameters offline, we use the *Covariance Matrix Adaption evolution strategy (CMA-ES)* [12], a variable-metric evolutionary algorithm which represents the "state-of-the-art in evolutionary optimization in real-valued optimization" [13]. As a baseline for comparison, we also optimize the four parameters of the reference approach the same way.

5.1 Experimental Setup

We optimized the parameters for the benchmark images from the Middlebury evaluation site as an example. As objective the *average number of bad pixels* as defined in [1] was minimized. Using the CMA-ES implementation from the *Shark* open-source machine learning library [14], five optimization trials were conducted for the basic approach and five for our modified approach.

5.2 Results

For the basic approach we obtained $14.5\,\%$ bad pixels as best solution with $c_{occ} = 28.8$, $c_{smooth} = 31.7$, $p = 1.5$ and $t_I = 5.1$. The perfomance is very similar to that of the reference implementation using the Birchfield-Tomasi measure.

For our modified approach, the best solution for the parameter settings given in table 1 results in $8.8\,\%$ bad pixels. Figure 3 shows results for two of the test images.

Table 1. Optimized parameter settings for our approach

Parameter	p_D	c_D	p_V	c_V	r_D	r_V	t_I	w_R	w_G	w_B	
Best	30.7	27.4	−5.3	-12.9	-2.6	3.6	45.9	0.32	0.62	0.06	
Parameter	$\hat{p}_D$	$\hat{c}_D$	$\hat{p}_V$	$\hat{c}_V$	$\hat{r}_D$	$\hat{r}_V$	Δ_c	τ	λ	μ	C_{max}
Best	43.9	19.0	−16.7	−13.7	−1.9	4.0	1.95	1.17	22.6	57.5	76.2

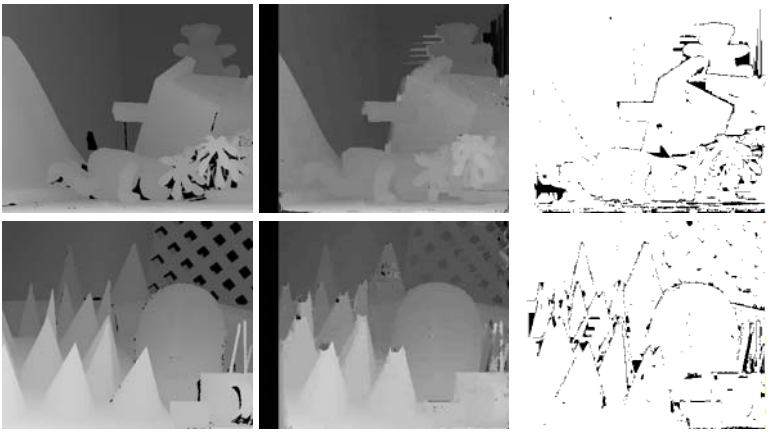

Fig. 3. Left to right: ground-truth disparity images, results obtained from our approach (border and occlusions shown black), and error images for Middlebury test images Teddy and Cones

Table 2. Results for Middlebury benchmark data sets

Rank	Algorithm	Tsukuba			Venus			Teddy			Cones			Avg. percent bad pixels
		nocc	all	disc	nocc	all	disc	nocc	all	disc	nocc	all	disc	
38	**Our approach**	2.0	3.8	9.8	3.3	4.7	13.0	6.5	13.9	16.6	5.2	13.7	13.4	8.83
⋮														
42	RealTimeGPU [9]	2.1	4.2	10.6	1.9	3.0	20.3	7.2	14.4	17.6	6.4	13.7	16.5	9.82
43	CostRelax [15]	4.8	6.1	20.3	1.4	2.5	18.5	8.2	15.9	23.8	3.9	10.2	11.8	10.6
44	ReliabilityDP [7]	1.4	3.4	7.3	2.4	3.5	12.2	9.8	16.9	19.5	12.9	19.9	19.7	10.7
45	TreeDP [10]	2.0	2.8	10.0	1.4	2.1	7.7	15.9	23.9	27.1	10.0	18.3	18.9	11.7
⋮														
50	**DP [1]**	4.1	5.0	12.0	10.1	11.0	21.0	14.0	21.6	20.6	10.5	19.1	21.1	14.2

See table 2 for detailed results and comparsions. Thus, our approach shows better performance in the Middlebury evaluation than most other algorithms using DP and it is the best one performing pixel-wise DP.

Processing test images *Tsukuba* takes $0.2\ s$, *Venus* $0.4\ s$, *Teddy* $0.8\ s$ and *Cones* $0.8\ s$ on a standard desktop PC with $1.8\ GHz$.

6 Conclusion

We considered a freely available implementation of a standard stereo algorithm based on DP. The technique is very popular due to its applicability to a large variety of real-world problems.

We showed how to modify and extend it in order to provide higher performance and more flexibility. In particular we focused on utilizing available information during DP more efficiently. At the same time the computational complexity did not increase significantly.

Offline optimization of all algorithm parameters for benchmark images as an example showed that an error reduction of $40\,\%$ compared to the reference implementation has been allowed. The algorithm proposed is widely applicable, escpecially for real-time applications.

References

1. Scharstein, D., Szeliski, R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. International Journal of Computer Vision 47, 7–42 (2002)
2. Scharstein, D., Szeliski, R.: High-accuracy stereo depth maps using structured light. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 195–202 (2003)
3. Ohta, Y., Kanade, T.: Stereo by intra- and inter-scanline search using dynamic programming. IEEE Transactions on Pattern Analysis and Machine Intelligence 7, 139–154 (1985)
4. Geiger, D., Ladendorf, B., Yuille, A.L.: Occlusions and binocular stereo. In: Sandini, G. (ed.) ECCV 1992. LNCS, vol. 588, pp. 425–433. Springer, Heidelberg (1992)
5. Birchfield, S., Tomasi, C.: Depth discontinuities by pixel-to-pixel stereo. International Journal of Computer Vision 35, 1073–1080 (1999)
6. Forstmann, S., Kanou, Y., Ohya, J., Thuering, S., Schmitt, A.: Real-time stereo by using dynamic programming. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshop, vol. 3, p. 29 (2004)
7. Gong, M., Yang, Y.H.: Near real-time reliable stereo matching using programmable graphics hardware. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 924–931 (2005)
8. Kim, J.C., Lee, K.M., Choi, B.T., Lee, S.U.: A dense stereo matching using two-pass dynamic programming with generalized ground control points. In: Proceedings IEEE Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 1075–1082 (2005)
9. Wang, L., Liao, M., Gong, M., Yang, R., Nister, D.: High-quality real-time stereo using adaptive cost aggregation and dynamic programming. In: Proceedings of the International Symposium on 3D Data Processing, Visualization and Transmission, pp. 798–805 (2006)
10. Veksler, O.: Stereo correspondence by dynamic programming on a tree. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 384–390 (2005)
11. Bobick, A.F., Intille, S.S.: Large occlusion stereo. International Journal of Computer Vision 33(3), 181–200 (1999)
12. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation 9(2), 159–195 (2001)
13. Beyer, H.G.: Evolution strategies. Scholarpedia 2(8), 1965 (2007)
14. Igel, C., Glasmachers, T., Heidrich-Meisner, V.: Shark. Journal of Machine Learning Research 9, 993–996 (2008)
15. Brockers, R., Hund, M., Mertsching, B.: Stereo vision using cost-relaxation with 3D support regions. In: Image and Vision Computing, New Zealand (2005)