Cooperative Stereo Matching with Color-Based Adaptive Local Support*

Roland Brockers

Jet Propulsion Laboratory - California Institute of Technology
4800 Oak Grove Drive, Pasadena, California
brockers@jpl.nasa.gov

Abstract. Color processing imposes a new constraint on stereo vision algorithms: The assumption of constant color on object surfaces used to align local correlation windows with object boundaries has improved the accuracy of recent window based stereo algorithms significantly. While several algorithms have been presented that work with adaptive correlation windows defined by color similarity, only a few approaches use color based grouping to optimize initially computed traditional matching scores. This paper introduces the concept of color-dependent adaptive support weights to the definition of local support areas in cooperative stereo methods to improve the accuracy of depth estimation at object borders.

1 Introduction

A closer look at recent publications in stereo vision shows that the old separation between slow algorithms, employing a large amount of computational power to achieve the most accurate results, and fast real-time algorithms concentrating on efficiency, becomes more and more blurred. Recent improvements in computer hardware seem to open space for an incorporation of additional computational power to real-time algorithms to increase accuracy by new improvement steps beyond the well known fixed window correlate-and-winner-takes-all scheme of former real-time approaches. Examples are algorithms using local improvement steps that follow the disparity estimation [1] or basic dynamic programming based optimization methods [2].

Another group of algorithms with the potential to significantly improve real-time approaches are algorithms using color grouping for an advanced adaptive correlation [3,2]. Surprisingly, grouping neighboring pixels that are assumed to be located on the same object surface by the similarity of their color is relatively new in stereo vision. Recent algorithms use this constraint to obtain local adaptive correlation windows which are better aligned to object borders, resulting in better correlation accuracy at disparity discontinuities [4,5,6,3]. Grouping is achieved either by color segmentation [4,5,6] or by calculating color-dependent correlation weights [3] to control the influence of pixels inside a correlation window on the matching score. Unfortunately, relatively large correlation windows are needed by these algorithms to eliminate ambiguities (in [3] the typical window size is 33x33 pixels, in [6] results are given for 51x51 windows), which results in a higher computational cost making these approaches unsuitable for

* This research was supported by DFG grant BR-3583/1-1.

X. Jiang and N. Petkov (Eds.): CAIP 2009, LNCS 5702, pp. 1019–1027, 2009.

(a) (b) (c)

Fig. 1. Disparity maps for *Teddy* image detail (a): fixed local support leads to object blurring (b), while adaptive local support maintains accurate object borders (c)

real-time applications. To reduce the window size significantly and benefit from a faster implementation we transfer the idea of color grouping to a cooperative optimization algorithm which is highly suited to hardware implementation.

Cooperative approaches are a relatively old group of stereo methods, originally motivated by the biological model of the human visual cortex (cp. [7]). They optimize initially calculated matching scores in a three dimensional disparity label space $\phi(x, y, d)$ by iterative local cooperation of neighboring labels using mainly the stereoscopic continuity and uniqueness constraint coded in the variable update function. While the implementation of the uniqueness constraint differs among algorithms, all cooperative approaches implement the continuity constraint by coupling labels of neighboring pixels within local support areas. In traditional approaches these are fixed local support areas, which can be calculated very efficiently, but entail a common disadvantage: the coupling of labels across object borders, resulting in enlargement of foreground objects. When no adjustments are made, cooperative algorithms will always extend object surfaces from areas with high initial matches into regions with low matching scores, e.g. occluded areas or low textured regions of the input images (cp. Fig. 1). To cope with this general problem of fixed-window based algorithms, alignment mechanisms were proposed in the literature to get more accurate object borders during post-processing [1]. However, within an optimization algorithm, automated adjustment inside the optimization is preferable. Good results can be achieved e.g. with methods performing an initial color-based segmentation to group pixels belonging to the same object surface [4,5,6]. In this paper we show that a computationally expensive pre-segmentation is not necessary and that the grouping can be coded within pre-calculated weights of local support areas used in a cooperative optimization suitable for later hardware acceleration.

The proposed algorithm is described in section 2 and 3. In section 4 the new algorithm is applied to common test scenes with ground truth from the Middlebury stereo vision page [8] to demonstrate the quality of the approach and to compare the results with other related algorithms. Finally, section 5 summarizes the results.

2 Matching Costs

The structure of the proposed algorithm is similar to traditional stereo algorithms (cp. [9]). As an initial guess a correlation based similarity measure quantifies the similarities of potential corresponding pixel pairs. Matching scores are calculated using a

sampling insensitive version of normalized cross correlation which correlates the gray-scale intensities of the rectified input images to achieve tolerance to noise in local chromaticity (eq. 1-3). Because of the later optimization it is possible to choose a very small correlation window size (e.g. 3x3 pixels). This reduces calculation time and minimizes the influence of errors near disparity discontinuities caused by a non-constant disparity inside the correlation window.

$$s_0(x,d) = \frac{m_{x,d}}{s_{l,x} s_{r,x+d}} \tag{1}$$

$$m_{x,d} = \sum_{\tilde{x}} \varphi\Big\{ i_l(x+\tilde{x}), \bar{i_l}(x) \Big\} \varphi\Big\{ i_r(x+d+\tilde{x}), \bar{i_r}(x+d) \Big\} \tag{2}$$

$$s_{a,k} = \sqrt{\sum_{\tilde{x}} (\varphi\left\{ i_a(k+\tilde{x}), \bar{i_a}(k) \right\})^2} \tag{3}$$

To achieve insensitivity to image sampling artifacts a difference operator φ is introduced to calculate the difference between the intensity of a particular pixel $i(x+\tilde{x})$ and the mean intensity $\bar{i}(x)$ inside the correlation window with an adapted version of the sampling insensitive dissimilarity measure of Birchfield and Tomasi [10]. As the intensity mean already includes an averaging among several pixel intensities, only $i(x+\tilde{x})$ is expected to be appreciably influenced by sampling effects. Therefore φ calculates the minimum difference between $i(x+\tilde{x})$, linearly interpolated in an interval of $\pm 1/2$ pixel, and the window mean (eq. 5) and uses this difference to calculate the deviation term for the cross correlation (eq. 4).

$$\varphi\left\{ i(x_1), \bar{i}(x_2) \right\} = i(x_m) - \bar{i}(x_2) \tag{4}$$

$$\text{with } |i(x_m) - \bar{i}(x_2)| = \min_{x_1 - \frac{1}{2} \le x \le x_1 + \frac{1}{2}} |i(x) - \bar{i}(x_2)| \tag{5}$$

Note, that this definition of the difference operator minimizes the absolute differences while maintaining the sign of the difference as this is essential for the calculation of cross correlation.

3 Cooperative Optimization

In a second step, a cooperative optimization process adjusts the probability values of the similarity measure to calculate an optimal solution with respect to the implicitly coded stereoscopic constraints. For efficiency reasons, the cooperative optimization is formulated as an iterative cost minimization approach with a global cost function containing only squared cost terms (cp. [11]). To simplify equations, all labels in the disparity space are ordered in a single order parameter vector

$$\boldsymbol{\xi} = (\xi_{(1,d_{min})}, ..., \xi_{(n,d_{min})}, \xi_{(1,d_{min}+1)}, ..., \xi_{(n,d_{max})})^T \tag{6}$$

where the first index $i \in [1, ..., n]$ enumerates all pixels in the reference view.

The global cost function P defines two different cost terms for each variable $\xi_{(i,d)}$:

$$P(\boldsymbol{\xi}) = c_1 \sum_{d=d_{min}}^{d_{max}} \sum_{i=1}^{n} (\xi_{(i,d)} - \xi_{(i,d)_0})^2 + c_2 \sum_{d=d_{min}}^{d_{max}} \sum_{i=1}^{n} \sum_{j \in U_i} \gamma_{ij} (\xi_{(i,d)} - \xi_{(j,d)})^2 \quad (7)$$

Costs are generated when a variable $\xi_{(i,d)}$ differs from its initial value $\xi_{(i,d)_0}$ given by the similarity measure $s_0(x, d)$ (correlation confidence) or if the variable values within a local neighborhood U are diverging (continuity) (eq. 7).

The local support area U_i of a pixel i is defined by a local window surrounding i where the influences of neighboring pixels j are weighted by individual adaptive support weights γ_{ij} which are explained in the following section. c_1 and c_2 are positive constants to trim the final global cost function. Because of the squared cost terms, equation 7 has only one minimum, characterizing the global solution for the optimization problem (cp. [11]). It is calculated numerically with the gradient descent method, where the update function for the iteration is defined as:

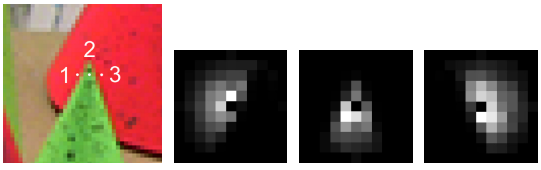
$$\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_k - \lambda \nabla P(\boldsymbol{\xi}_i), \quad \lambda > 0 \quad (8)$$

with

$$\frac{\delta P}{\delta \xi_{i,d}} = [2c_1 + 4c_2 \sum_{j \in U_i} \gamma_{ij}] \, \xi_{(i,d)} - 2c_1 \xi_{(i,d)_0} - 4c_2 \sum_{j \in U_i} \gamma_{ij} \xi_{(j,d)}. \quad (9)$$

After convergence, the valid disparity for each pixel in the reference view is selected by a winner-takes-all maximum search over all variables attached to the same pixel (uniqueness).

Computationally, this approach has some advantages compared to other cooperative approaches like [7] or [12]. The avoidance of local competition leads to a simple, linear first derivate of the cost function (eq. 9), making it possible to use a fast standard minimization method to compute the global cost minimum. All calculations are local, which implies a high potential for parallelization in a manner amenable to hardware implementations. Additionally, all calculations consist only of additions and multiplications which is particularly important for FPGA implementation. Finally, the independent definition of the local neighborhood U_i with its appropriate weights makes it possible to implement different kinds of local support. In the proposed algorithm, we define U_i as a circular 2D window in a constant disparity level with pre-calculated adaptive weights γ_{ij}.



(a) (b) (c) (d)

Fig. 2. Adaptive local support: (a) detail of *Cones* scene with marked window positions; (b-d) adaptive weights local support areas for position 1-3 ($D = 11$, $\sigma_c = 6$, $\sigma_r = 2.04$)

3.1 Adaptive Local Support

The local support weights γ_{ij} define the amount of support a variable gets from its neighbors depending on color similarity c_{ij} and spatial distance r_{ij} of the underlying pixels (eq. 10). The definition of the individual weight terms essentially follows Tomasi and Manduchi's definition of bilateral filters [13]. To reflect perceptual metrics, the color difference is defined by the Euclidean distance of the two color vectors $\mathbf{c}_i$ and $\mathbf{c}_j$ of pixel i and j in CIELab color space, calculated from the reference view after applying a bilateral filter to remove local noise. The spatial distance is determined by the Euclidean distance between the image coordinate vectors of i and j. All distances are weighted by Gaussian weighting functions (eq. 11 and 12).

$$\gamma_{ij} = r_{ij} \cdot c_{ij} \tag{10}$$

$$r_{ij} = e^{-\frac{1}{2}\left(\frac{\delta_e(\mathbf{x}_i,\mathbf{x}_j)}{\sigma_r}\right)^2} = e^{-\frac{1}{2}\frac{(x_i-x_j)^2+(y_i-y_j)^2}{\sigma_r^2}} \tag{11}$$

$$c_{ij} = e^{-\frac{1}{2}\left(\frac{\delta_e(\mathbf{c}_i,\mathbf{c}_j)^4}{\sigma_c^2}\right)} = e^{-\frac{1}{2}\frac{((L_i-L_j)^2+(a_i-a_j)^2+(b_i-b_j)^2)^2}{\sigma_c^2}} \tag{12}$$

This differs from the weight definition in [3] where the authors choose a Laplacian kernel for the weighting function, which is less optimal because of lower compactness and smaller weights in the immediate proximity of the center pixel. Note that the color distance in (12) has an exponent of 4 for a sharper separation of color edges.

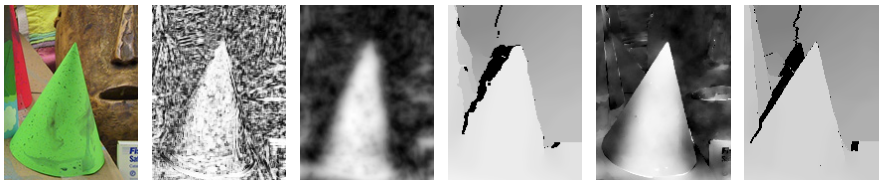
To define a local support area U_i for pixel i, adaptive support weights γ_{ij} are calculated for all neighboring pixels j inside a fixed local neighborhood of circular shape with diameter D to minimize the mean spatial distance between supporting pixels and the center. While all variables in U_i contribute to the development of the center variable i, the center variable itself is excluded from the support window to inhibit self amplification. Figure 2 shows an example of local support areas in an area of color transition in the *cones* scene of the Middlebury data set.

3.2 Occlusion Detection and Sub-pixel Precision

After the optimization process is complete, occlusions are explicitly detected by searching the disparity map for pixels that point to the same corresponding pixel in the non-reference view or are in succession in a cyclopean view, using the optimized correspondence probabilities (cp. [11]). To calculate sub-pixel precise disparities, the relaxation process is applied once again to the 2D pixel-precise disparity map similar to [11].

4 Experimental Results

In the following we evaluate our algorithm with images from the Middlebury test data set to demonstrate the capabilities of the new approach. Figure 3 illustrates the effect of adaptive local support on variable values before and after the optimization. After initialization with the similarity measure, variables linked to the correct object disparity



(a) (b) (c) (d) (e) (f)

Fig. 3. Adaptive local support compared with fixed local support: (a) detail of reference view; (b) similarity measure in a constant disparity level of $d = 49$; (c) variable values after optimization with fixed local support (circ. Gaussian weighted, $D = 7$); (d) disparity map for fixed local support; (e) variable values after optimization with adaptive weights local support ($\sigma_c = 6$, $\sigma_r = 1.84$, $D = 7$); (f) resulting disparity map

already contain high values, but a large amount of noise can be observed as well due to the small correlation window size of 3x3 pixels (Fig. 3b). After the optimization most of the ambiguity is resolved. If using fixed local support, high variable values are propagated to neighboring variables disregarding of object boundaries (Fig. 3c), leading to misalignment of object borders and calculated disparity edges in the final disparity map (Fig. 3d). With adaptive local support, cross boundary coupling is significantly reduced, resulting in sharp object contours that are aligned with the real object boundaries in both the variable activity map (Fig. 3e) and the disparity map (Fig. 3f).

For quantitative comparison with related stereo algorithms, we applied our algorithm to all four Middlebury test scenes according to [9] ($\delta_d = 1.0$, constant parameter set and back filling of detected occlusions). The calculated disparity maps are shown in Figure 4 and the corresponding error percentages of false matches are illustrated in Table 1. Compared to the *CostRelax* approaches, which use basically a similar optimization method, adaptive local support clearly outperforms fixed local support. With the new similarity measure, accuracy can be further improved, most visibly in low resolution images like *Tsukuba*, which are more prone to sampling artifacts. Compared to other methods, which also use color-based adaptive grouping, the proposed algorithm achieves good results. In Figure 5 the error development during iteration is illustrated for the *Tsukuba* scene. Fig. 5b points out a major advantage of local optimization methods: The quick decrease of overall errors allows an early termination of the iteration

Table 1. Percentage of bad matching pixels for the Middlebury data set ($\delta_d = 1$) for non occluded pixels (non), all pixels (all) and near discontinuities (disc) (*cp. [8] visited in 03/2009)

Algorithm	**Tsukuba**			**Venus**			**Teddy**			**Cones**		
	non	all	disc	non	all	disc	non	all	disc	non	all	disc
SegmentSupport [6]*	1.25	1.62	6.68	0.25	0.64	2.59	8.43	14.2	18.2	3.77	9.87	9.77
AdaptiveWeight [3]*	1.38	1.85	6.90	0.71	1.19	6.13	7.88	13.3	18.6	3.97	9.79	8.26
Our Method (ncc+BT, 3x3)	**2.91**	**3.49**	**11.4**	**0.60**	**1.11**	**6.45**	**7.92**	**13.7**	**20.9**	**3.59**	**9.43**	**10.3**
Our Method (regular ncc, 3x3)	**4.60**	**5.10**	**13.0**	**0.69**	**1.62**	**6.11**	**8.06**	**16.0**	**21.0**	**3.60**	**12.3**	**10.4**
CostRelax [14] (3D fixed local supp.)*	4.76	6.08	20.3	1.41	2.48	18.5	8.18	15.9	23.8	3.91	10.2	11.8
CostRelax [11] (2D fixed local supp.)	6.33			1.44			9.60			5.24		

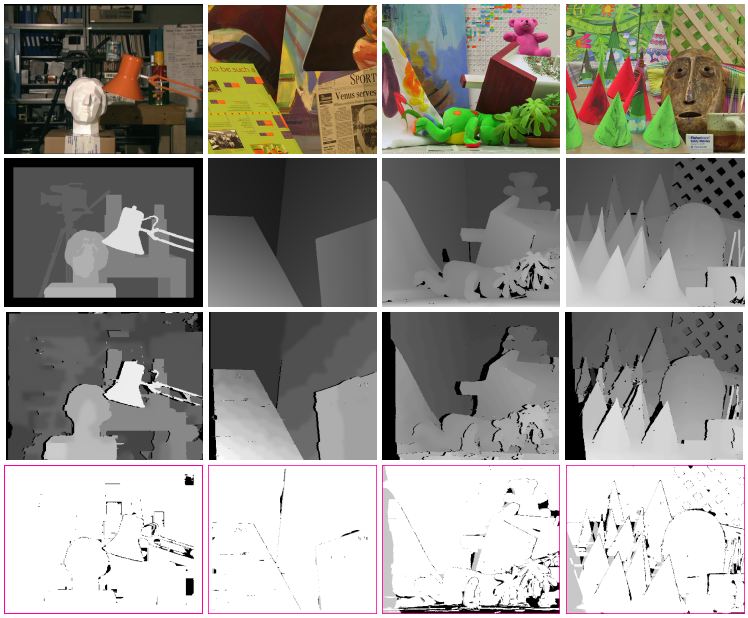


Fig. 4. Results for *Tsukuba*, *Venus*, *Teddy* and *Cones* scene; from top to bottom: left reference view; ground truth; disparity map calculated with proposed algorithm, with black labeled occlusions; error map with false matches (black) and unconsidered errors in true occluded areas (gray) ($c_1 = 0.5, c_2 = 10, c_3 = 0.5, c_4 = 0.2, \sigma_c = 6, \sigma_r = 8, D = 5$, 400 iterations)

in time critical applications, giving a calling process the ability to provide only the momentary available calculation time for stereo calculation while still benefiting from significantly improved disparity maps. When comparing fixed and adaptive local support, the biggest difference is visible at object boundaries (Fig. 5a). Where the old fixed support algorithm generates significant object blurring during iterations, adaptive local support decreases errors over time and maintains accurate object borders.

However, difficulties remain in un-textured areas. This is in part due to the relatively small window size of local support (e. g. $D = 5$ for results in Table 1) and the strict

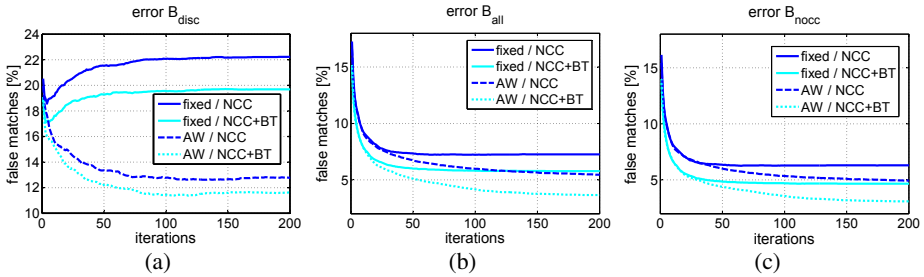


Fig. 5. Error during iteration for the *Tsukuba* scene: Adaptive weights local support (AW) compared with fixed local support (fixed); (a) error near discontinuities; (b) error for all pixels; (c) error in true non-occluded regions. For the definition of different regions see [8].

limitation on non-global optimization. In future work we plan to investigate variable window sizes that adapt to the local amount of texture to deal with these difficulties. In our tests, the algorithm was coded in standard non-optimized C++ code using full float precision and run on an 2.4GHz Intel Core2Duo T7700. The calculation time for the *Tsukuba* scene with 100 iterations was 20s on a single core and 11.5s on both cores. In a future FPGA based implementation we expect to run the algorithm in near real-time.

5 Conclusion

Traditional cooperative stereo algorithms are well known for producing blurry object borders due to a fixed coupling of neighboring pixels to implement the stereoscopic continuity constraint. In this paper we demonstrated that color-based adaptive local support can be used to align support areas with object borders and, thus keep accurate object boundaries throughout the cooperative optimization process. Implemented in a simple and fast relaxation algorithm which uses a new sampling insensitive normalized cross correlation for initial matching, our results show that optimization with local adaptive support generates results comparable with other algorithms that also use adaptive local grouping, while avoiding large correlation windows or computationally expensive pre-segmentation.

References

1. Hirschmüller, H., Innocent, P.R., Garibaldi, J.: Real-time correlation-based stereo vision with reduced border errors. Int. J. Comp. Vision 47, 229–246 (2002)
2. Wang, L., Liao, M., Gong, M., Yang, R., Nister, D.: High-quality real-time stereo using adaptive cost aggregation and dynamic programming. In: Proc. Int. Symp. on 3D Data Processing, Visualization, and Transmission (3DPVT), pp. 798–805 (2006)
3. Yoon, K.J., Kweon, I.S.: Adaptive support-weight approach for correspondence search. IEEE Trans. PAMI 28, 650–656 (2006)
4. Zhang, Y., Kambhamettu, C.: Stereo matching with segmentation-based cooperation. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002. LNCS, vol. 2351, pp. 556–571. Springer, Heidelberg (2002)
5. Klaus, A., Sormann, M., Karner, K.: Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. In: Proc. Int. Conf. Pat. Recogn. (ICPR), pp. III 15–18 III (2006)
6. Tombari, F., Mattoccia, S., Di Stefano, L.: Segmentation-based adaptive support for accurate stereo correspondence. In: Mery, D., Rueda, L. (eds.) PSIVT 2007. LNCS, vol. 4872, pp. 427–438. Springer, Heidelberg (2007)
7. Marr, D., Poggio, T.: Cooperative computation of stereo disparity. Science 194, 283–287 (1976)
8. Middlebury stereo vision page, `http://vision.middlebury.edu/stereo`
9. Scharstein, D., Szeliski, R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. Int. J. Comp. Vision 47, 7–42 (2002)
10. Birchfield, S., Tomasi, C.: A pixel dissimilarity measure that is insensitive to image sampling. IEEE Trans. PAMI 20, 401–406 (1998)
11. Brockers, R., Hund, M., Mertsching, B.: Stereo matching with occlusion detection using cost relaxation. In: Proceedings of the IEEE International Conference on Image Processing, ICIP, pp. III–389– III–392 (2005)

12. Zitnick, C.L., Kanade, T.: A cooperative algorithm for stereo matching and occlusion detection. IEEE Trans. PAMI 22, 675–684 (2000)
13. Tomasi, C., Manduchi, R.: Bilateral filtering for gray and color images. In: Proc. IEEE Int. Conf. on Comp. Vision (ICCV), pp. 839–846 (1998)
14. Brockers, R., Hund, M., Mertsching, B.: Stereo vision using cost-relaxation with 3d support regions. In: Proc. Image and Vision Comp. New Zealand (IVCNZ), pp. 96–101 (2005)