HAL
open science

Bilaterally Weighted Patches for Disparity Map Computation

Laura Fernández Julià, Pascal Monasse

To cite this version:

Laura Fernández Julià, Pascal Monasse. Bilaterally Weighted Patches for Disparity Map Computation. Image Processing On Line, 2015, 5, pp.73-89. 10.5201/ipol.2015.123 . hal-01131412

HAL Id: hal-01131412

https://enpc.hal.science/hal-01131412v1

Submitted on 16 Mar 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

2014/07/01 v0.5 IPOL article class

Published in Image Processing On Line on 2015–03–11.
Submitted on 2014–08–29, accepted on 2015–02–23.
ISSN 2105–1232 © 2015 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials, software, datasets and online demo at
http://dx.doi.org/10.5201/ipol.2015.123

Bilaterally Weighted Patches for Disparity Map Computation

Laura F. Julià[1], Pascal Monasse[1]

[1] Université Paris-Est, LIGM (UMR CNRS 8049), ENPC, F-77455 Marne-la-Vallée
({fernandl,monasse}@imagine.enpc.fr)

Abstract

Visual correspondence is the key for 3D reconstruction in binocular stereovision. Local methods perform block-matching to compute the disparity, or apparent motion, of pixels between images. The simplest approach computes the distance of patches, usually square windows, and assumes that all pixels in the patch have the same disparity. A prominent artifact of the method is the "foreground fattening effect" near depth discontinuities. In order to find a more appropriate support, Yoon and Kweon introduced the use of weights based on color similarity and spatial distance, analogous to those used in the bilateral filter. This paper presents the theory of this method and the implementation we have developed. Moreover, some variants are discussed and improvements are used in the final implementation. Several examples and tests are presented and the parameters and performance of the method are analyzed.

Source Code

The `C++` implementation for this algorithm is available in the IPOL web page of this article[1]. Compilation and usage instructions are included in the `README.txt` file of the archive. Note that an optional rectification step can be launched before running the algorithm.

Keywords: stereovision; bilateral filter

1 Introduction

Stereo vision matching algorithms aim at estimating the depth of a scene given two photographs taken from different points of view. When the camera makes a fronto-parallel motion between the two images, depth estimation is done by estimating the disparity of each pixel, that is its apparent displacement from the first image (reference image) to the second image (target image). This disparity permits to recover the 3D position of the point that was photographed in this pixel since disparity is inversely proportional to depth [1].

The core of disparity map computation methods is pixel correspondence. In order to compute the disparity of a pixel in the reference image, the corresponding pixel in the target image has to be

[1] http://dx.doi.org/10.5201/ipol.2015.123

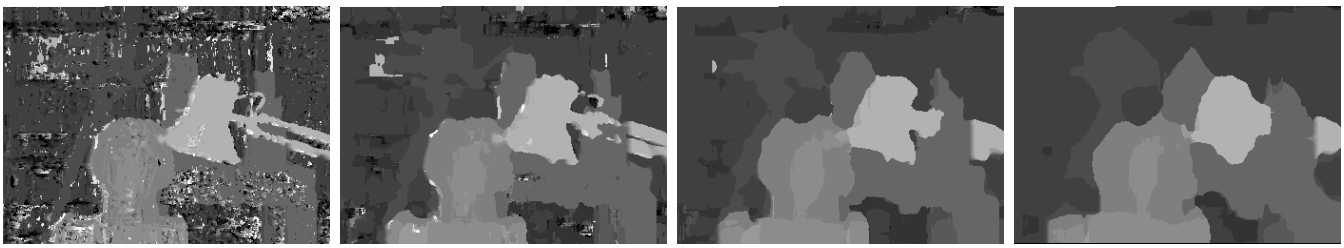
(a) Reference image. (b) Target image. (c) Ground-truth disparity map.

Figure 1: *Tsukuba* image pair and its ground truth disparity map: the intensity is proportional to the apparent horizontal displacement, the black border meaning "no data".

found. With rectified pairs of images the search of the corresponding pixel is limited to the pixels on the same horizontal line in the target image. Therefore, the correspondence search can be done measuring the similarity between pixels and their neighborhoods and choosing the best match.

A key challenge in the correspondence search is image ambiguity, which results from the ambiguous local appearances of image pixels due to image noise and insufficient (or repetitive) texture. By using local support windows, called patches, the image ambiguity is reduced efficiently while the discriminative power of the similarity measure is increased. In this approach, it is implicitly assumed that all pixels in a patch are at a similar depth in the scene and, therefore, that they have similar disparities. The artifact resulting from the violation of this assumption is the foreground-fattening phenomenon.

This effect appears when a pixel is wrongly assigned the disparity corresponding to other pixels in its patch. For this reason, although taking larger patches leads to smoother results, it also increases the fattening effect, producing errors near the depth discontinuities (this phenomenon occurring on the data of Figure 1 is illustrated in Figure 2).



(a) 3×3 (b) 9×9 (c) 20×20 (d) 35×35

Figure 2: Resulting disparity maps for *Tsukuba* obtained with a simple block-matching method (using SAD, sum of absolute differences, for patch similarity) for several patch sizes. The visible dilation of foreground shapes appearing with the increase of the patch size is the fattening phenomenon.

To avoid this phenomenon, adaptive-window methods try to find an optimal support patch for each pixel. In this article, we study the correspondence search method proposed by Yoon and Kweon [9]. The method consists in computing the support weights of the pixels in a patch using color similarity and geometric proximity to the center. The weights permit using larger patches, getting better results in homogeneous regions, while not provoking the fattening effect near depth discontinuities.

The weights introduced by this method are analogous to the ones used in the bilateral filter applied to image denoising [8]. Recall that the resulting image I_{BF} obtained by applying the bilateral filter

to the original image I in a pixel p is computed as

$$I_{BF}(p) = \frac{\sum_{q \in N_p \cap I} w(p,q)\, I(q)}{\sum_{q \in N_p \cap I} w(p,q)}, \tag{1}$$

where $N_p = [x-r, x+r] \times [y-r, y+r]$ is a neighborhood of $p = (x,y)$ and the weights are parameterized by positive real numbers σ_d and σ_r

$$w(p,q) = \exp\Big(-\frac{\|p-q\|^2}{2\sigma_d^2} - \frac{\|I(p)-I(q)\|^2}{2\sigma_r^2} \Big). \tag{2}$$

We have slightly abused the notation in (1), by noting also I the rectangle domain of the image I.

2 The Method

Let us note I the reference image and $\tilde{I}$ the target image of a rectified stereo pair. Then, $I_c(p)$ will be the color intensity in the color channel c ($c = r$ for red, $c = g$ for green, $c = b$ for blue) of the pixel p in the reference image I. Also, given a pixel $p = (x,y)$ in the image I and a disparity d, then $\tilde{p}_d$ will be the pixel in $\tilde{I}$ with coordinates $\tilde{p}_d = (x+d, y)$. The method is composed of three parts: adaptive support-weight computation, dissimilarity computation based on the support-weights and disparity selection.

2.1 Weight Assignation Based on Gestalt Grouping

Like in other block-matching methods, we take into account a support window of neighbor pixels to compare it to another patch in the target image. But not all the pixels in these patches may be relevant to do the comparison. Ideally, only pixels belonging to the same object and at the same depth should be taken into account. Considering the difficulty of explicitly segmenting an image, a pragmatic approach is adopted, assigning weights or probabilities to the neighbor pixels, trying to imitate the human visual system mechanism for grouping similar points. Therefore, the similarity of pixels in the patch is measured and weights corresponding to the probability to be grouped with the central pixel are assigned.

The strongest Gestalt grouping principles of the visual system are the color similarity and the spatial proximity [6]. Following these two principles, the support-weight of a pixel p with respect to another pixel q can be written as

$$w(p,q) = f(\Delta c_{pq}, \Delta g_{pq}) = w_{col}(\Delta c_{pq}) \cdot w_{pos}(\Delta g_{pq}), \tag{3}$$

where Δc_{pq} and Δg_{pq} are the color distance and the spatial distance between p and q, respectively. The function f gives the strength of grouping by similarity and proximity, and if we consider Δc_{pq} and Δg_{pq} as independent events, the strength can be measured separately by w_{col} and w_{pos}.

Yoon and Kweon [9] justify their choice of w_{col} with the perceptual difference between two colors,

$$D(c_p, c_q) = 1 - \exp\Big(-\frac{\Delta c_{pq}}{14} \Big), \tag{4}$$

which is defined in the CIE Lab color space, and where Δc_{pq} is the Euclidean distance. Based on this, they define the similarity strength as

$$w_{col}(\Delta c_{pq}) = \exp\Big(-\frac{\Delta c_{pq}}{\gamma_{col}} \Big), \tag{5}$$

depending on a fixed positive parameter γ_{col}.

In our implementation we used colors in the RGB space and the L^1 distance

$$\Delta c_{pq} = \frac{1}{3}\|I(p) - I(q)\|_1 = \frac{1}{3}\sum_{c\in\{r,g,b\}} |I_c(p) - I_c(q)|. \tag{6}$$

Working in the CIE Lab space implies extra computations and, in addition, our tests yield better results using RGB and the L^1 norm. The computation advantage of using the L^1 norm comes from the fact that it is easy to tabulate: if color channels are integers between 0 and 255, the L^1 norm is an integer between 0 and $3 \cdot 255$, whose exponential values can be tabulated. Our tests exhibit a huge impact on computation time when tabulating: tabulation divides the running time by a factor 13.

Following the same strategy, the proximity strength is defined as

$$w_{pos}(\Delta g_{pq}) = \exp\Big(-\frac{\Delta g_{pq}}{\gamma_{pos}}\Big), \tag{7}$$

where γ_{pos} should be the patch radius and Δg_{pq} is the Euclidean distance

$$\Delta g_{pq} = \|p - q\|_2 = \sqrt{(x - x')^2 + (y - y')^2} \tag{8}$$

where $p = (x, y)$ and $q = (x', y')$. Putting (3), (5) and (7) together we have

$$w(p, q) = w_{col}(p, q) \cdot w_{pos}(p, q) = \exp\Big(-\Big(\frac{\Delta c_{pq}}{\gamma_{col}} + \frac{\Delta g_{pq}}{\gamma_{pos}}\Big)\Big). \tag{9}$$

In Figure 3 some examples of weighted patches are shown for the *Tsukuba* image.

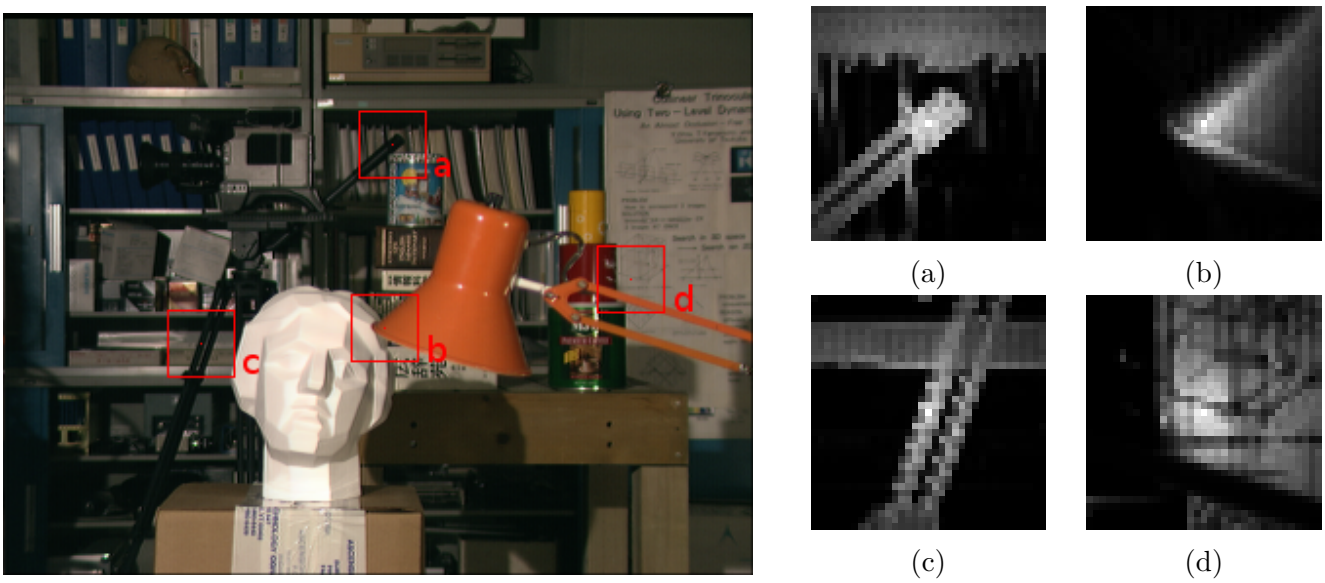


Figure 3: Resulting weights for four different 35×35 patches in the *Tsukuba* left image, with $\gamma_{col} = 12$ and $\gamma_{pos} = 17$.

2.2 Dissimilarity Computation and Disparity Selection

For the dissimilarity computation step of the method we have to take into account the weights that we have just defined. In the computation of the matching cost between pixels, a raw matching cost e

based on the color similarity is combined with the corresponding support-weights in both reference and target patches.

The dissimilarity between two pixels p (in the reference image) and $\tilde{p}_d$ (in the target image with disparity d) is defined as

$$E(p,\tilde{p}_d) = \frac{\sum_{q\in N_p\cap I, \tilde{q}_d\in N_{\tilde{p}_d}\cap\tilde{I}} \operatorname{comb}(w(p,q), w(\tilde{p}_d,\tilde{q}_d))\, e(q,\tilde{q}_d)}{\sum_{q\in N_p\cap I, \tilde{q}_d\in N_{\tilde{p}_d}\cap\tilde{I}} \operatorname{comb}(w(p,q), w(\tilde{p}_d,\tilde{q}_d))}. \tag{10}$$

The function comb : $\mathbb{R}^2 \to \mathbb{R}$ combines the weights of tentative corresponding pixels q and $\tilde{q}_d$. In the original article [9], comb is the product of its arguments, but we consider variants later on. A faster algorithm is the asymmetric case, where $\operatorname{comb}(w_1, w_2) = w_1$ because the weights in the target image need not be computed. Notice that the former combination leads to a spatial weighting by $\exp(-\|p-q\|_2/\gamma_{pos})^2 = \exp(-2\|p-q\|_2/\gamma_{pos})$, while the latter gives a weight $\exp(-\|p-q\|_2/\gamma_{pos})$. To have consistent values in all cases, we modified slightly the definition by pulling the proximity term outside the combination function

$$E(p,\tilde{p}_d) = \frac{\sum_{q\in N_p\cap I, \tilde{q}_d\in N_{\tilde{p}_d}\cap\tilde{I}} w_{pos}(p,q)^2 \operatorname{comb}(w_{col}(p,q), w_{col}(\tilde{p}_d,\tilde{q}_d))\, e(q,\tilde{q}_d)}{\sum_{q\in N_p\cap I, \tilde{q}_d\in N_{\tilde{p}_d}\cap\tilde{I}} w_{pos}(p,q)^2 \operatorname{comb}(w_{col}(p,q), w_{col}(\tilde{p}_d,\tilde{q}_d))}. \tag{11}$$

In (11), the raw matching cost e between pixels q and $\tilde{q}_d$ is defined as the truncated (by a positive parameter τ_{col}) absolute difference of color

$$e(q,\tilde{q}_d) = e_c(q,\tilde{q}_d) = \min\Big\{\frac{1}{3}\sum_{c\in\{r,g,b\}} |I_c(q) - \tilde{I}_c(\tilde{q}_d)|\ ,\ \tau_{\text{col}}\Big\}. \tag{12}$$

However, results improve by introducing another term in the raw matching cost taking into account the similarity of the x-derivative value between both pixels [7][2]. Analogously to e_c, define

$$e_g(q,\tilde{q}_d) = \min\big\{|\nabla_x I(q) - \nabla_x \tilde{I}(\tilde{q}_d)|\ ,\ \tau_{\text{grad}}\big\}, \tag{13}$$

where $\nabla_x I$ is the x-derivative computed as in the article by Tan and Monasse [7] and τ_{grad} is a threshold for the gradient difference. Hence, we redefine the total raw matching cost as a linear combination of e_c and e_g,

$$e(q,\tilde{q}_d) = (1-\alpha)\cdot e_c(q,\tilde{q}_d) + \alpha\cdot e_g(q,\tilde{q}_d) \tag{14}$$

with $\alpha \in [0,1]$.

Several choices for the combination function comb (some symmetric, one asymmetric) will be tested, but the choice for the demo is the multiplication of its arguments. A pixel's final disparity is selected by the winner-takes-all method,

$$d_p = \operatorname*{argmin}_{d\in S_d} E(p,\tilde{p}_d), \tag{15}$$

where $S_d = \{d_{\min},\ldots,d_{\max}\}$ is the set of all possible disparities, which is assumed known.

[2]The use of the x-derivative, among other choices, is due to the horizontal apparent motion, since a horizontal edge (with large y-derivative) is not discriminative because of the aperture problem

3 Implementation

We have implemented this algorithm in C++. Our implementation is based on the code by Tan and Monasse [7], the main differences being in the file `disparity.cpp`.

The pseudo-code for the disparity map computation using adaptive weights is presented in Algorithm 1. The key to a fast algorithm is to compute only once the support weight patches in the target image $\tilde{I}$, based on the trivial observation

$$(x+1) + S_d = x + (S_d \cup \{d_{\max}+1\} \setminus \{d_{\min}\}) . \tag{16}$$

In other words, the patches in $\tilde{I}$ that are compared to pixel $(x+1, y) \in I$ are the same as for pixel $(x, y) \in I$, with the addition of the new patch centered on $(x+1+d_{\max}, y) \in \tilde{I}$ and the subtraction of the one centered on $(x+d_{\min}, y)$. If the patches of $\tilde{I}$ centered at pixels $\{(x+d, y) : d \in S_d\}$ are stored in an array `weights`, we can simply update the array to pixel $x+1$ by storing at index $x+d_{\min}$, whose patch centered on $(x+d_{\min}, y)$ is no longer useful, the new patch centered on $(x+1+d_{\max}, y)$. This is why in Algorithm 1, the lines 7-8 initialize the patches centered on pixels $\{(0+d, y) : d \in S_d\}$ (except for $d_{\max}$, computed in the x loop) that will be used for the pixel $(0, y)$ of I. Then, for each new x, only the patch of $\tilde{I}$ centered on $(x+d_{\max}, 0)$ needs be computed at line 11. Of course, this storage is useless in the asymmetric case, when only the weights of I are used.

Another factor to accelerate the implementation is the tabulation of the similarity and proximity strengths in order to accelerate the computation of weights in function `support` (Algorithm 2). As observed above, this is possible only when we use the L^1 color distance. The same code is able to use different combinations of weights in the dissimilarity computation[3], the function `costCombined` applies Equation (11) for the chosen combination. Also, the raw matching costs are computed only once for every pair of pixels. For every possible disparity value the matching costs of pixels in the image are precomputed and stored in a new image, as explained in Algorithm 3. These images form the layers of the cost volume, defined as

$$cost[d](p) = e(p, \tilde{p}_d). \tag{17}$$

In the original article [9], the authors do not specify whether they use a post-processing method to detect and fill occlusions and smooth the resulting disparity map, but it seems that they do. In our implementation we have used the post-processing presented and implemented by Tan and Monasse [7]. The code is exactly the one presented in the article and explained there. It proceeds by first detecting inconsistent pixels in left-right disparity maps, replacing their disparity with a simple scan line based filling, and finally applying a median filter with weights taken from the original image.

4 Results

4.1 Study of Parameters

The parameters of the method are summed up in Table 1, where the default values used in our experiments are also specified. The values that we have used are not the same as the default parameters of Yoon and Kweon [9] since they had to be adapted to a different color space and norm and the new matching cost formula. We chose the values that gave the better results for the Middlebury benchmark images. Moreover, tuning the parameters for each image individually can lead to better results in the final disparity map.

We tested our implementation varying the parameters (one at a time) and studied the error in the resulting disparity map. We compared the resulting disparity maps for the Middlebury benchmark

[3] This is a compile-time option, the dynamic use of a function pointer resulting in slower code.

Algorithm 1: Disparity map computation, function `disparityAW`

input : Color images I and $\tilde{I}$, disparity range $S_d = \{d_{\min}, \ldots, d_{\max}\}$, patch radius r, parameters γ_{pos}, γ_{pol}.
output: Disparity maps $d_1(\cdot)$, $d_2(\cdot)$ for images I and $\tilde{I}$ respectively.

```
1  Tabulate color similarity strengths distC(·), array of size 3·255+1.   // Equations (5), (6)
2  Tabulate proximity strengths distP(·), array of size (2r+1)^2.         // Equations (7), (8)
3  Compute array cost of #S_d cost images for all disparities.            // Algorithm 3
4  E_1(·) ← +∞   E_2(·) ← +∞
5  foreach 0 ≤ y < height do
6      Create array weights of #S_d patches.
7      foreach d ∈ [d_min, d_max − 1] do                                  // Weighted target patches
8          weights[d − d_min] ← support(Ĩ, d, y, distC).                  // Algorithm 2
9      foreach 0 ≤ x < width do
10         W_1 ← support(I, x, y, distC).                                 // Algorithm 2
11         weights[x + d_max − d_min (mod #S_d)] ← support(Ĩ, x + d_max, y, distC).  // Algorithm 2
12         foreach d ∈ S_d do
13             W_2 ← weights[x + d − d_min (mod #S_d)]
14             c ← cost[d − d_min]
15             Compute dissimilarity E combining W_1, W_2, c, and distP.  // Equation (11)
16             if E < E_1(x, y) then
17                 E_1(x, y) ← E
18                 d_1(x, y) ← d
19             if E < E_2(x + d, y) then
20                 E_2(x + d, y) ← E
21                 d_2(x + d, y) ← −d
```

// The blue lines are not executed in the asymmetric combination of weights.

Algorithm 2: Weighted patch computation, function `support`

input : Color image I, central pixel p, patch radius r, color similarity strength table $distC$.
output: $(2r+1) \times (2r+1)$ image of color-based weights $weight$ centered on p in I.

```
1  weight(.) ← 0
2  foreach pixel q ∈ N_p ∩ I do
3      Compute δ ← ‖I(p) − I(q)‖_1
4      weight(q) = distC[δ]
```

Algorithm 3: Cost volume computation

input : Color images I and $\tilde{I}$, disparity range $S_d = \{d_{\min}, \ldots, d_{\max}\}$, parameters α, τ_{col}, τ_{grad}.
output: Cost image vector *cost*.

Compute x-derivatives $\nabla_x I$ and $\nabla_x \tilde{I}$.
foreach $d \in S_d$ **do**
  Create cost image $cost[d](\cdot)$
  foreach $p \in I$ **do**
    $cost_c \leftarrow \tau_{\text{col}} \quad cost_g \leftarrow \tau_{\text{grad}}$
    if $\tilde{p}_d \in \tilde{I}$ **then**
      $cost_c \leftarrow \min(\frac{1}{3}\Delta c_{p\tilde{p}_d}, \tau_{\text{col}})$ // Equation (12)
      $cost_g \leftarrow \min(|\nabla_x I(p) - \nabla_x \tilde{I}(\tilde{p}_d)|, \tau_{\text{grad}})$ // Equation (13)
    $cost[d](p) \leftarrow (1-\alpha)cost_c + \alpha\, cost_g$ // Equation (14)

Parameter	Value	Description
α	0.9	Value in $[0,1]$ for the combination of the color and the gradient similarity. The bigger α, the more important the gradient term is with respect to the color term.
γ_{col}	12	Positive parameter for the color similarity strength.
γ_{pos}	17.5	Positive parameter for the proximity strength.
r	17	Radius of the patch considered in the correspondence search.
τ_{col}	30	Color threshold for the matching cost computation.
τ_{grad}	2	Gradient threshold for the matching cost computation.

Table 1: Default parameters of the algorithm.

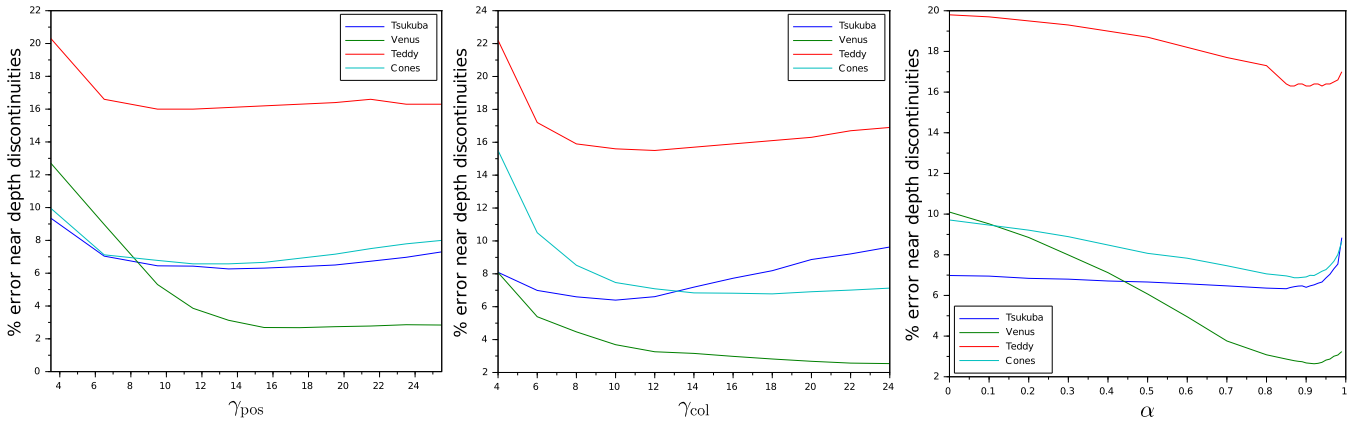
with their ground truth and computed the percentage of "bad" pixels (pixels whose absolute disparity error is greater than 1) for all the pixels in the image, the pixels in non-occluded areas and pixels near depth discontinuities.

Seeing that varying some parameters had a greater effect on the error near depth discontinuities than in other pixels, we show in Figure 4a the variation of this error for the four Middlebury benchmark images when varying γ_{pos}, γ_{col} and α. For γ_{pos}, we varied γ_{pos} alongside the patch radius r to see the error evolution when we changed the size of the patch. The error stabilizes for $\gamma_{pos} > 15.5$ ($r = 15$) so we preserved the default value of Yoon and Kweon [9] $\gamma_{pos} = 17.5$ ($r = 17$) and patch size 35×35.

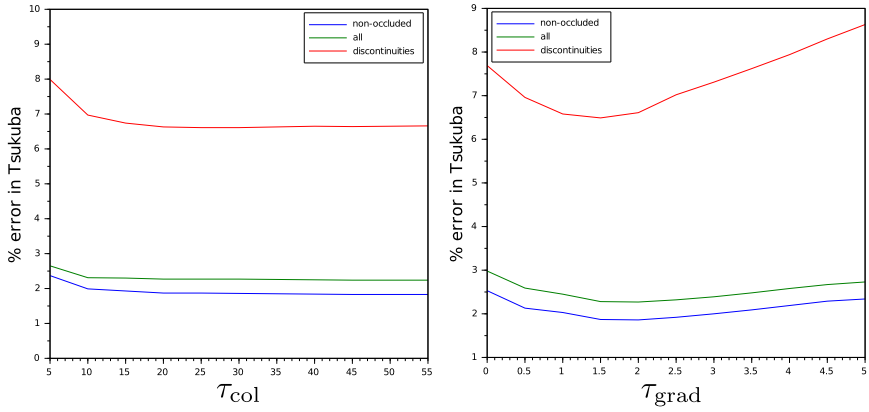
The essence of the method lies in a good choice for the parameter γ_{col}. As we increase γ_{col}, we take less into account the color distance w.r.t. the central pixel, so we take more pixels in the window into consideration for the comparison. This translates to the results in a first improvement of the accuracy when γ_{col} is increased, but the error grows eventually when too many pixels with different disparities are taken into account for high γ_{col}. As γ_{col} gets very large, we are using regular block matching. To find a compromise between all the images we choose $\gamma_{col} = 12$.

The addition of the gradient term proves to be very beneficial and choosing $\alpha = 0.9$ we get the best results. Notice that although this term is highly helpful, we should not disregard the color term, since the error increases sharply after $\alpha = 0.95$. Figure 5 shows the influence of α in the resulting disparity map. We can see that increasing α we erase some of the errors in the background and recover the edges a lot better, with less fattening effect.

Regarding the color and the gradient thresholds, their effect on the error is similar for all four images, so we chose to illustrate their influence by showing in Figure 4b the results for the *Tsukuba*

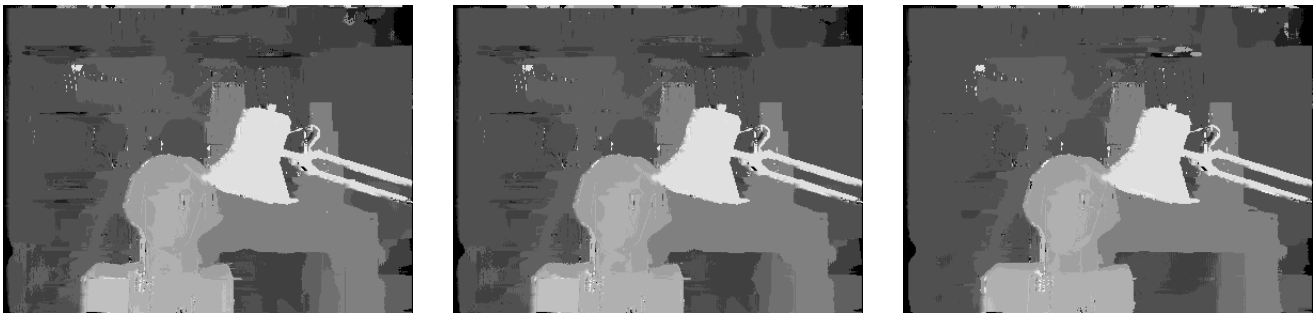


(a) Influence of the parameters γ_{pos}, γ_{col} and α on the error percentage near depth discontinuities for the four reference images.



(b) Influence of the thresholds τ_{col} and τ_{grad} on the error percentage in *Tsukuba* image.

Figure 4: Influence of the parameters on the error percentage.



(a) $\alpha = 0$, total error $= 5.18\%$. (b) $\alpha = 0.5$, total error $= 4.82\%$. (c) $\alpha = 0.9$, total error $= 4.46\%$.

Figure 5: Disparity maps for *Tsukuba* for different values of α without the post-processing step.

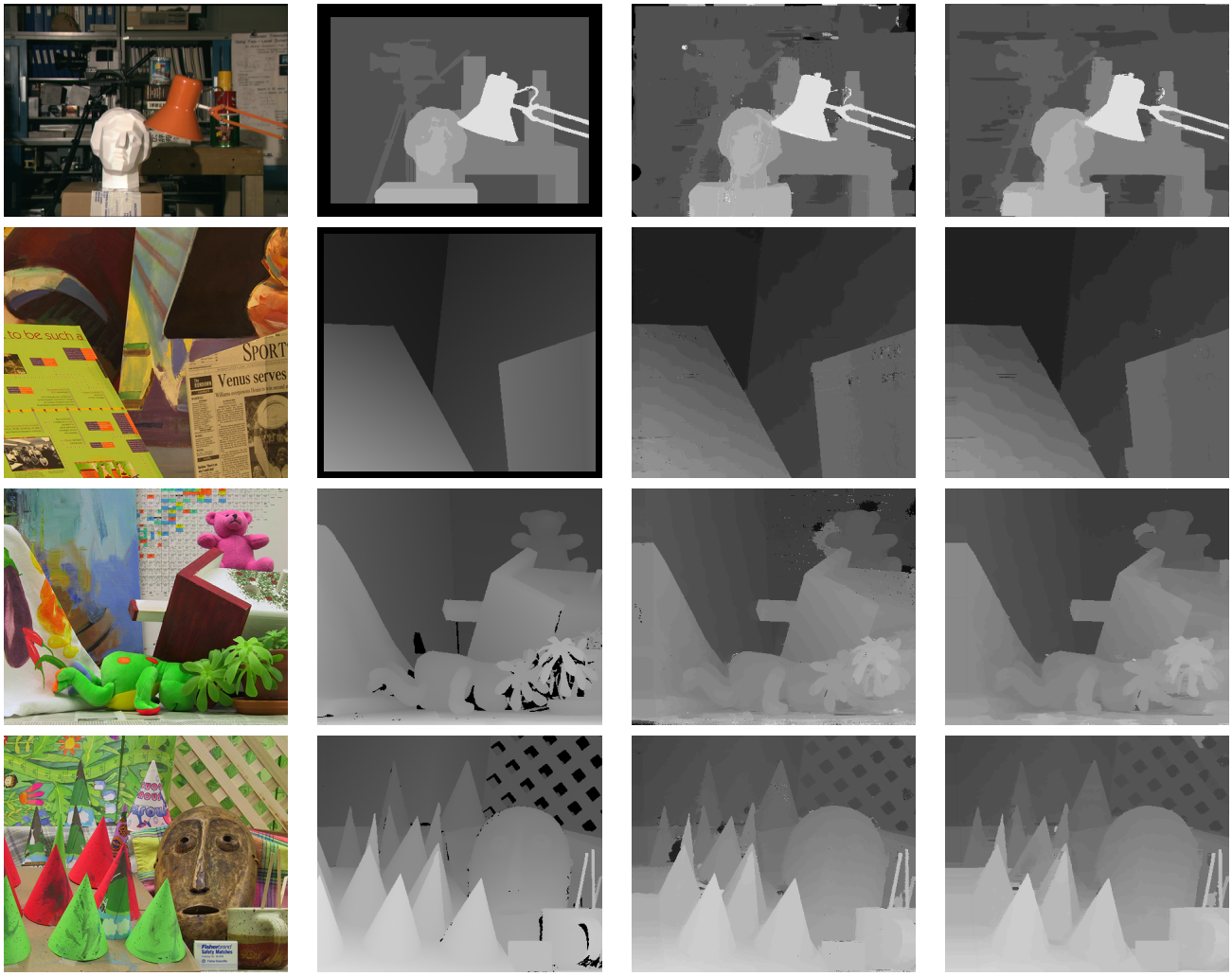
image at all pixels, pixels near depth discontinuities and non-occluded pixels. It is clear that while the error decreases when τ_{col} increases and stabilizes after $\tau_{col} = 20$, the influence of τ_{grad} is greater and its optimal value is $\tau_{grad} = 2$.

4.2 Middlebury Benchmark Disparity Maps

In this section the results for the Middlebury benchmark are presented and compared to those presented in the original article. The parameters values used in all cases are the ones specified in Table 1. The disparity maps have been improved and smoothed by a final post-processing step as in

the article by Tan and Monasse [7].

In Figure 6, we can notice that whereas we get a few additional isolated errors compared to the original method and we are not able to compute the disparity in some small regions, our implementation recovers the depth discontinuities much better and with less fattening effect. As a result, our implementation achieves better results in general as it is shown in Table 2.



Left image | Ground truth | Our results | Results from [9]

Figure 6: Disparity maps and comparison with the results of Yoon and Kweon [9].

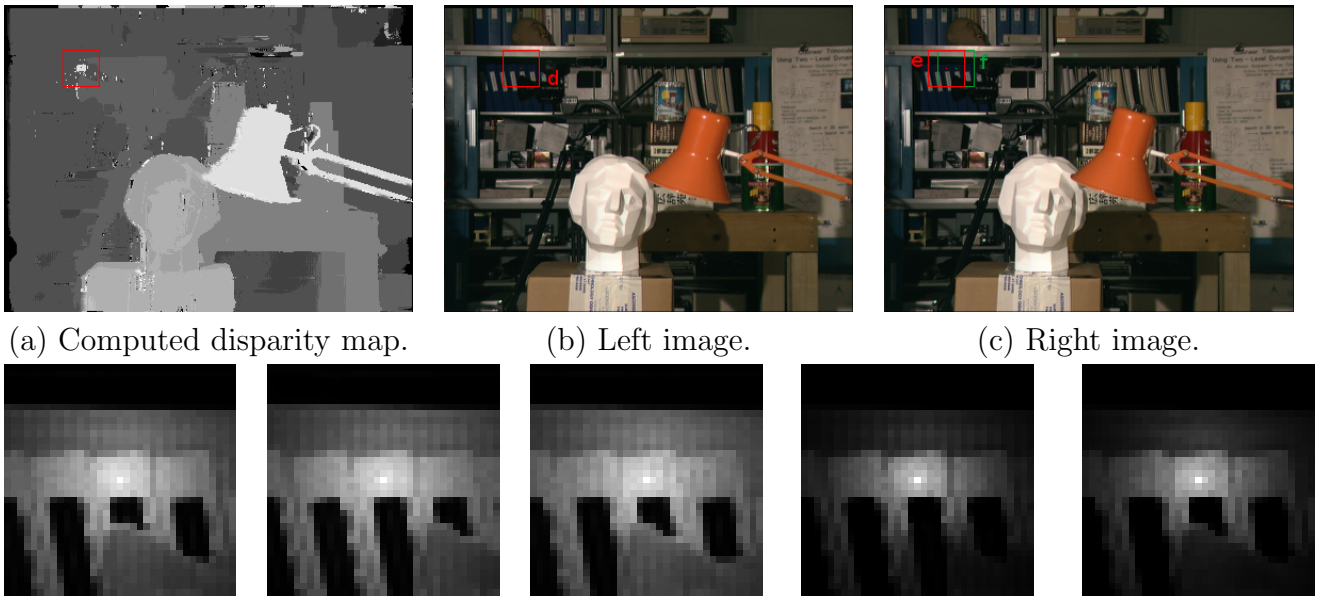
		Tsukuba			Venus			Teddy			Cones		
	Rank	non	all	disc	non	all	disc	non	all	disc	non	all	disc
Original article [9]	80.2	**1.38**	**1.85**	6.90	0.71	1.19	6.13	7.88	**13.3**	18.6	3.97	9.79	8.26
Ours w/o post-proc.	90.4	2.50	4.46	7.25	1.29	2.86	4.61	7.60	17.0	17.0	3.13	13.8	8.29
Ours with post-proc.	**63.7**	1.86	2.27	**6.61**	**0.65**	**1.02**	**3.15**	**6.56**	14.4	**15.5**	**2.48**	**8.81**	**6.91**

Table 2: Error percentages on Middlebury stereo benchmark (with tolerance of ± 1 pixel for disparity). The rank is the algorithm average rank in the Middlebury website. In **bold** are the best (i.e., lowest) values of the three rows for each column. Our results overcome the ones from [9] in most of the cases. The results are significantly improved by the post-processing.

4.3 Weights at Error Pixels

The adaptive weights method combined with the use of the gradient gives good results and it has a good performance at disparity discontinuities. Nevertheless, it is not able to overcome some of the usual problems of local methods. Indeed, most of the errors produced by this method are found in repetitive areas, homogeneous regions or isolated points with a very different color from their neighbors. Examples of these errors are presented in Figures 7, 8 and 9, and the weights around the erroneous pixels are shown in order to understand their cause.

Figure 7 is an example of repetition (stroboscopic effect). The binders in the shelf are organized one after another and they look all similar. The method fails to find the good correspondence in the case of some pixels in that region since it ignores the pixels that are not part of the repetition. In fact, the pixels are assigned to the previous binder.



(a) Computed disparity map. (b) Left image. (c) Right image.

(d) $w(p_d, .)$ (e) $w(p_e, .)$ (f) $w(p_f, .)$ (g) $w(p_d, .)w(p_e, .)$ (h) $w(p_d, .)w(p_f, .)$

Figure 7: Example of repetition. The patch `d` of center p_d in (b) is matched to patch `e` of center p_e in (c), but the true correspondence is in fact patch `f` of center p_f. Images (d), (e) and (f) show the weights in the patches. (g) is the combinations of the weights (d) and (e). (h) is the combination of (d) and (f). The two combined weights are similar and since the color distances in both correspondences are also similar because of the repetition, the assignation fails to find the correct disparity.

Figure 8 is an example of isolated point. Below the bust in *Tsukuba* image there are some letters on the box that get a bad disparity. The problem in this case is the presence of some points in that area that have a color really different from their neighbors so the weights in the reference image (8a) are scattered. Consequently, the possible combinations of weights are also scattered and the similarity measure takes into account few pixels. Ultimately, the correspondence is assigned to a combination of weights that consists in almost all the weight concentrated in the central pixel. This effect was observed in a survey paper [3].

Figure 9 shows that the method, even with its large 35×35 windows, is not immune to the aperture problem, that is, the window is not large enough to contain significant horizontal gradient. In the background of the image there is a bookcase. The method assigns significant weights only to those pixels on the shelf (9a), which is a homogeneous region of the image. Because of that, the method fails to find the correct correspondence.

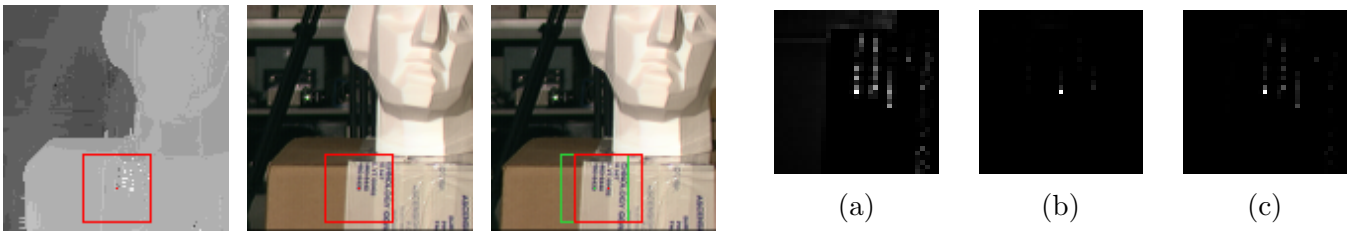


Figure 8: Example of isolated point. (a) Weighted patch of the reference pixel. (b) Combined weights of the detected correspondence. (c) Combined weights of the true correspondence.

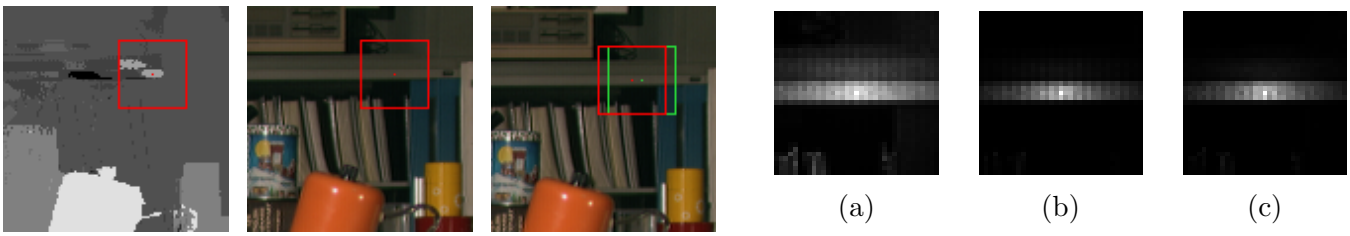


Figure 9: Example of aperture problem, where the patch is not discriminative enough, because it has no significant non-horizontal edge. (a) Weighted patch of the reference pixel. (b) Combined weights of the detected correspondence. (c) Combined weights of the true correspondence.

4.4 Variants of the Method

In our implementation we have modified the original color space used in the similarity strength (5), but we can study how the method works in the original space. One could also study other ways to combine the weights of the two patches when we compute the dissimilarity (11) for a certain pair of pixels. These variants of the method are studied in the following sections.

4.4.1 CIE Lab Space

Using the CIE Lab color space takes more computations due to the conversion of the values originally in RGB. Although we have not used this space in our implementation, some results using the CIE Lab space are presented below. For the implementation of the method in this space we used the following conversion:[4] if we have a color $c = (r, g, b)$ in RGB coordinates, first we have to convert it to XYZ coordinates,

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.950 \end{pmatrix} \begin{pmatrix} r/2.55 \\ g/2.55 \\ b/2.55 \end{pmatrix}$$

and then to CIE Lab,

$$\begin{cases} L = 116 \cdot f(\dfrac{Y}{Y_n}) - 16 \\ a = 500 \cdot (f(\dfrac{X}{X_n}) - f(\dfrac{Y}{Y_n})) \\ b = 200 \cdot (f(\dfrac{Y}{Y_n}) - f(\dfrac{Z}{Z_n})) \end{cases} \quad \text{with } f(t) = \begin{cases} t^{\frac{1}{3}} & \text{if } t > (\frac{6}{29})^3 \\ \frac{1}{3}\left(\frac{29}{6}\right)^2 t + \frac{4}{29} & \text{otherwise.} \end{cases}$$

where $(X_n, Y_n, Z_n) = (95.047, 100.00, 108.883)$ is the white point. The parameters used are $\gamma_{col} = 3.5$, $\tau_{\text{col}} = 10$, $\gamma_{pos} = 17.5$ and $\alpha = 0.9$. We can observe in Figure 10 that the results near discontinuities

[4]Formula taken from http://www.cs.rit.edu/~ncs/color/t_convert.html

are a little worse than using RGB and we get inaccurate edges. Apart from this, the numerical results are quite close to the ones using the RGB space (Table 3), some of them being even better (non-occluded pixels of *Venus* image).

	Tsukuba			Venus			Teddy			Cones		
	non	all	disc	non	all	disc	non	all	disc	non	all	disc
RGB space	1.86	2.27	6.61	0.65	1.02	3.15	6.56	14.4	15.5	2.48	8.81	6.91
CIE Lab space	1.85	2.60	7.94	0.50	1.25	5.33	7.53	15.1	18.3	3.35	9.83	9.58

Table 3: Error comparison between our implementation using CIE Lab and RGB color spaces.

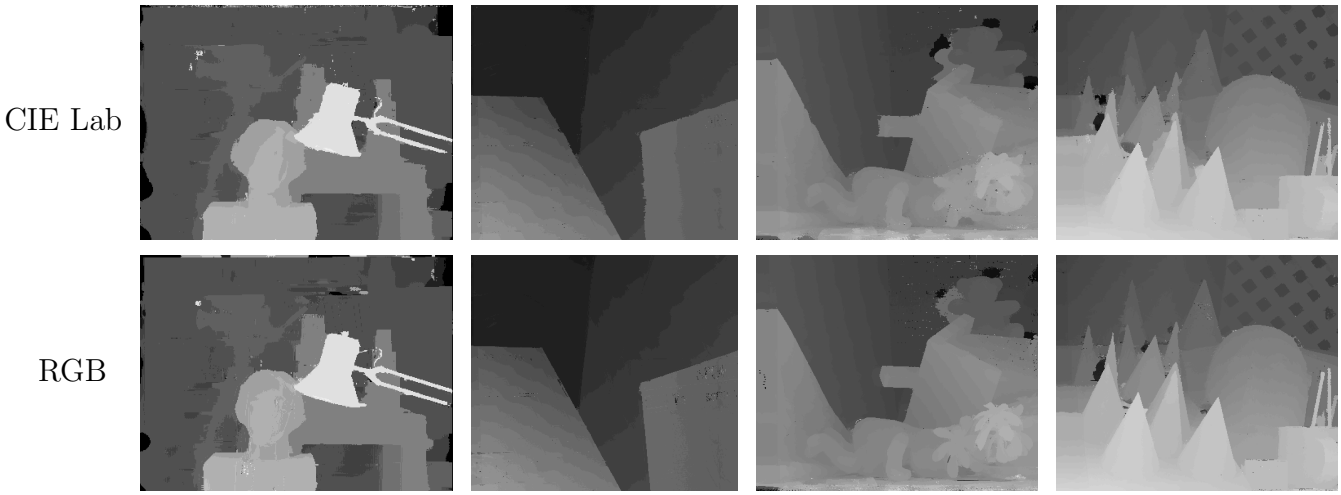


Figure 10: Comparison of color spaces for the Middlebury benchmark images.

4.4.2 Combination of Weights

To compute each patch pair matching cost, the presented method computes the product of weights of the target and reference patches, see Equation (11). In this section other possibilities are explored. The simplest approach is to consider only the weights of one patch (asymmetric weights), for example, those of the reference patch, which amounts to $w(\tilde{p}_d, \tilde{q}_d) = 1$ in Equation (11). Another approach, similar to taking the product is to take the sum of weights

$$E(p, \tilde{p}_d) = \frac{\sum_{q \in N_p \cap I, \tilde{q}_d \in N_{\tilde{p}_d} \cap \tilde{I}} \big(w(p,q) + w(\tilde{p}_d, \tilde{q}_d)\big) e(q, \tilde{q}_d)}{\sum_{q \in N_p \cap I, \tilde{q}_d \in N_{\tilde{p}_d} \cap \tilde{I}} \big(w(p,q) + w(\tilde{p}_d, \tilde{q}_d)\big)}. \tag{18}$$

The parameters that give the best results for these two approaches and used in our tests are $\gamma_{pos} = 18$, $\gamma_{col} = 4$ and $\tau_{\text{col}} = 20$. Finally, in order to take the union of supports and penalize non-coincident regions, we can consider taking the maximum of weights

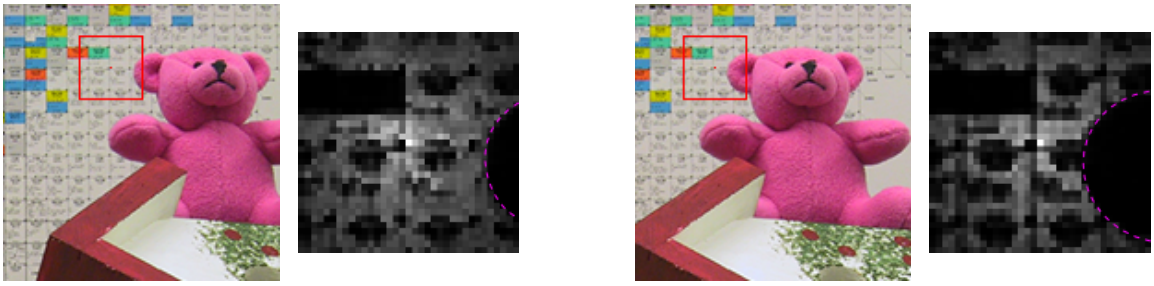
$$E(p, \tilde{p}_d) = \frac{\sum_{q \in N_p \cap I, \tilde{q}_d \in N_{\tilde{p}_d} \cap \tilde{I}} \max\big(w(p,q), w(\tilde{p}_d, \tilde{q}_d)\big) e(q, \tilde{q}_d)}{\sum_{q \in N_p \cap I, \tilde{q}_d \in N_{\tilde{p}_d} \cap \tilde{I}} \max\big(w(p,q), w(\tilde{p}_d, \tilde{q}_d)\big)}. \tag{19}$$

For this approach the best results are achieved by using $r = 14$, $\gamma_{pos} = 14$, $\gamma_{col} = 4$ and $\tau_{\text{col}} = 15$.

Figure 11 shows that near depth discontinuities, the best combination is the product: it puts a tiny weight on the region occupied by the ear of the teddy bear in left or right image. A program

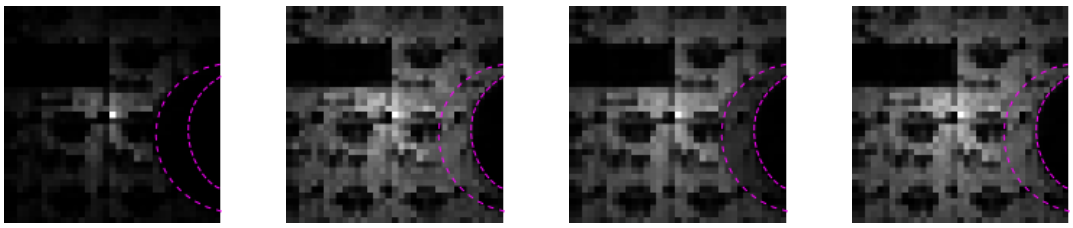
included in the source code archive, `show_weights`, permits to output the weighted patches with different combinations.

Visually, the results for the other alternatives are really similar to the ones given by the product (Figure 12). However, when considering the error (Table 4), the asymmetric and maximum combinations are a little worse in the discontinuities than the product and the sum (except for *Tsukuba*). Surprisingly, using asymmetric weights does not reduce the accuracy as much as we could expect and it proves to be a good substitute if we need to speed up computations. Nevertheless, the maximum has a worse performance than the other two variants, giving many errors and low accuracy in depth discontinuities.



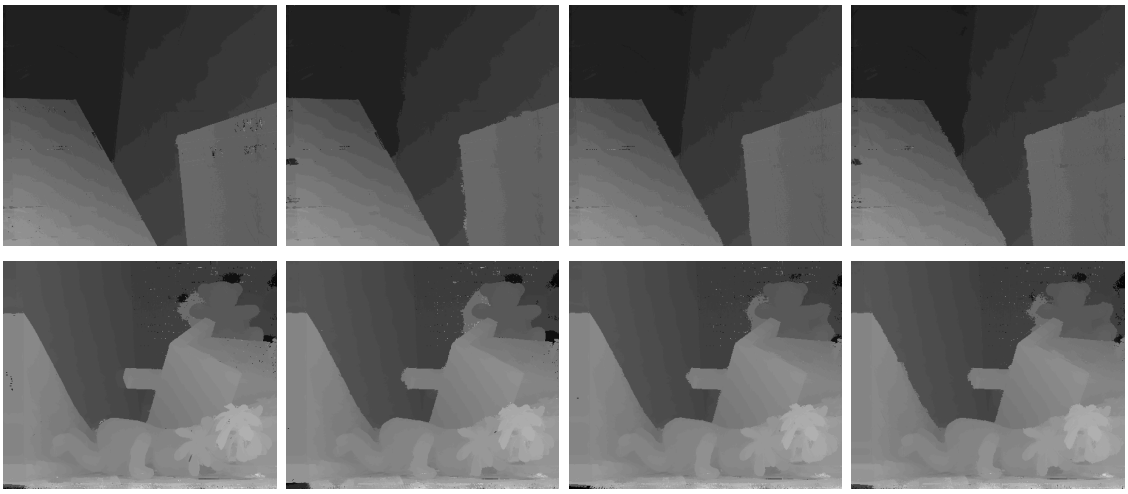
Patch and weights centered at pixel (307,44) in *Teddy* left image.

Patch and weights centered at the corresponding matching pixel (292,44) in *Teddy* right image.



(a) Product. (b) Asymmetric. (c) Sum. (d) Maximum.

Figure 11: Example of the resulting weights for the matching cost computation of two patches using different combinations of weights.



(a) Product. (b) Asymmetric. (c) Sum. (d) Maximum.

Figure 12: Disparity maps (with post-processing) comparison using diverse weights combinations for *Venus* and *Teddy* images.

	Tsukuba			Venus			Teddy			Cones		
	non	all	disc	non	all	disc	non	all	disc	non	all	disc
Product	1.86	2.27	6.61	0.65	1.02	3.15	6.56	14.4	15.5	2.48	8.81	6.91
Asymmetric	1.95	2.41	7.99	0.74	1.42	8.12	6.90	14.6	17.0	3.21	9.90	9.03
Sum	2.28	2.68	8.56	0.65	1.11	5.03	6.82	14.6	16.8	3.01	9.65	8.49
Maximum	2.74	3.14	10.8	1.62	2.23	11.7	7.72	15.2	18.6	3.38	9.72	9.35

Table 4: Error percentages on Middlebury benchmark for different weight combination functions.

4.5 Other Examples

We have tested the algorithm with other images and obtained satisfying results. Some examples using the default parameters are presented in Figure 13. The method is able to recover fine structures, for instance, most of the bars, brushes and holes in *Art* image are respected. In addition, homogeneous and textured regions are not a problem in most of the cases (*Flowerpots* and *Baby3* images). Nonetheless, we get some errors in the *Laundry* image in the region of the window.

5 Conclusion

In this article we have studied and implemented Yoon and Kweon's method [9] for disparity map computation. This adaptive support-weight approach was the first published block-matching method that put weights in the blocks. Others have followed, but a survey [3] found that the method of Yoon and Kweon [9] yields the best results. Very similar results, but with much faster computation, can be obtained using a guided filter of the cost volume [4], which can also be tested in IPOL [7]. Hosni et al. [2] propose to use an explicit segmentation of the image in order to speed up computations. Moreover, one of the best performing stereo matching methods [5] uses a faster approximation of the presented algorithm that selects cross-shaped neighborhoods. However, it remains to be seen in what measure the better results are due to the adaptive neighborhoods on their own.

Image Credits

All images by the authors (license CC-BY-SA) except:

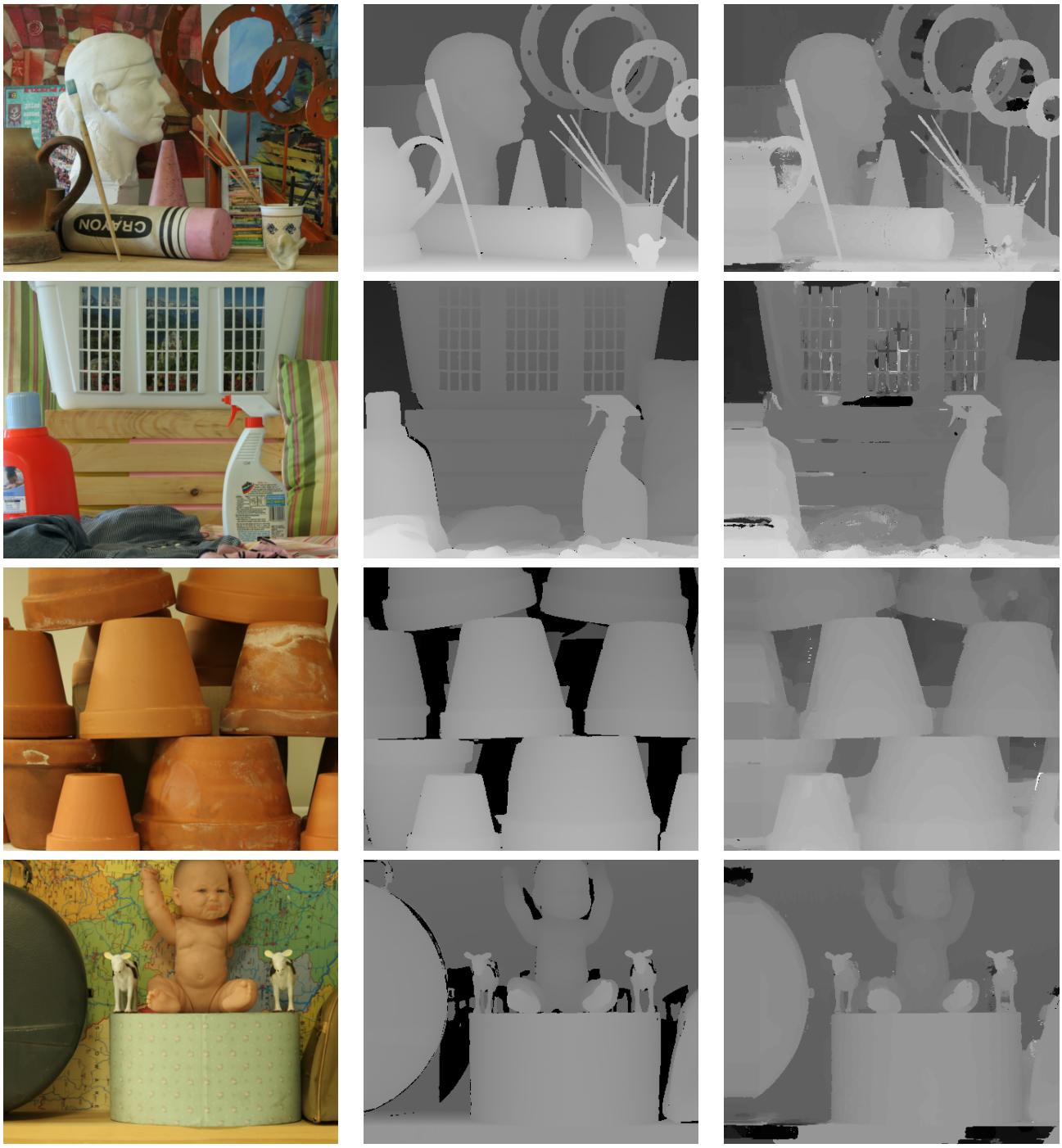
Middlebury.

References

[1] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*, Cambridge University Press, 2nd edition ed., 2004. ISBN 9780521540513.

[2] A. Hosni, M. Bleyer, and M. Gelautz, *Near real-time stereo with adaptive support weight approaches*, in International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT), 2010, pp. 1–8. http://publik.tuwien.ac.at/files/PubDat_190923.pdf.

[3] ——, *Secrets of adaptive support weight techniques for local stereo matching*, Computer Vision and Image Understanding, 117 (2013), pp. 620–632. http://dx.doi.org/10.1016/j.cviu.2013.01.007.



Left image Ground truth Our result

Figure 13: Resulting disparity maps for other images (from top to bottom: Art, Laundry from Middlebury 2005; Flowerpots, Baby3 from Middlebury 2006).

[4] A. Hosni, C. Rhemann, M. Bleyer, C. Rother, and M. Gelautz, *Fast cost-volume filtering for visual correspondence and beyond*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 35 (2013), pp. 504–511. http://dx.doi.org/10.1109/TPAMI.2012.156.

[5] X. Mei, X. Sun, M. Zhou, S. Jiao, H. Wang, and X. Zhang, *On building an accurate stereo matching system on graphics hardware*, in IEEE International Conference on Computer Vision Workshops (ICCV Workshops), 2011, pp. 467–474. http://dx.doi.org/10.1109/ICCVW.2011.6130280.

[6] W. Metzger, *Gesetze des Sehens*, W. Kramer Frankfurt am Main, 1936. ISBN 3782910478.

[7] P. Tan and P. Monasse, *Stereo Disparity through Cost Aggregation with Guided Filter*, Image Processing On Line, 4 (2014), pp. 252–275. http://dx.doi.org/10.5201/ipol.2014.78.

[8] C. Tomasi and R. Manduchi, *Bilateral filtering for gray and color images*, in IEEE Sixth International Conference on Computer Vision, 1998, pp. 839–846. http://dx.doi.org/10.1109/ICCV.1998.710815.

[9] K.-J. Yoon and I.S. Kweon, *Adaptive support-weight approach for correspondence search*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 28 (2006), pp. 650–656. http://dx.doi.org/10.1109/TPAMI.2006.70.