Real-time Spatiotemporal Stereo Matching Using the Dual-Cross-Bilateral Grid

Christian Richardt[1] Douglas Orr[1] Ian Davies[1]
Antonio Criminisi[2] Neil A. Dodgson[1]

[1]University of Cambridge, United Kingdom
[2]Microsoft Research Cambridge, United Kingdom

Abstract. We introduce a real-time stereo matching technique based on a reformulation of Yoon and Kweon's adaptive support weights algorithm [1]. Our implementation uses the bilateral grid to achieve a speedup of 200× compared to a straightforward full-kernel GPU implementation, making it the fastest technique on the Middlebury website. We introduce a colour component into our greyscale approach to recover precision and increase discriminability. Using our implementation, we speed up spatial-depth superresolution 100×. We further present a spatiotemporal stereo matching approach based on our technique that incorporates temporal evidence in real time (>14 *fps*). Our technique visibly reduces flickering and outperforms per-frame approaches in the presence of image noise. We have created five synthetic stereo videos, with ground truth disparity maps, to quantitatively evaluate depth estimation from stereo video. Source code and datasets are available on our project website[3].

1 Introduction

In contrast to global stereo matching techniques such as graph cuts [2] or belief propagation [3], Yoon and Kweon's adaptive support weights [1] only aggregate evidence over a finite window size. The effectiveness of their technique is due to aggregation of support over large window sizes and weights that adapt according to similarity and proximity to the central pixel in the support window. Results are good, but the algorithm is slow, taking about one minute to process the Tsukuba images on a current generation CPU. This has prompted people to resort to a separable implementation [4] to achieve interactive frame-rates.

We take a different approach. We rewrite their technique (section 2) as a *dual-cross-bilateral filter* with Gaussian weights (section 3). Based on the bilateral grid (section 3.1), we present a real-time GPU-based implementation (section 3.2) and improve its performance using a dichromatic approach (section 3.3). We show how spatial-depth super-resolution can be accelerated using our technique (section 3.4) and we extend our technique to stereo video (section 3.5). We conclude with results (section 4) and discussion of future work (section 5). Key literature is referred to in-line where it is most relevant.

[3] http://www.cl.cam.ac.uk/research/rainbow/projects/dcbgrid/

Published at ECCV 2010.
The original publication is available at www.springerlink.com.

2 Adaptive Support Weights

We start with a brief summary of Yoon and Kweon's technique. It builds on a winner-take-all stereo pipeline [5] and computes the initial cost space using truncated AD (absolute difference). We write this initial cost space as $C(\mathbf{p}, d)$ where $\mathbf{p} = (x, y)$ are the coordinates of a pixel in the left image and d is some disparity hypothesis. For convenience, let $\overline{\mathbf{p}} = (x - d, y)$ be the corresponding pixel in the right image.

The key idea is to aggregate costs over a large support window of 35×35 pixels for each pixel, where each pixel in the support window is weighted according to similarity and proximity to the central pixel. This is motivated by the *Gestalt* theory of perceptual grouping, with the weight between two pixels given by

$$w(\mathbf{p}, \mathbf{q}) = \exp\left(-\frac{\Delta E(\mathbf{p}, \mathbf{q})}{\gamma_c} - \frac{\|\mathbf{p} - \mathbf{q}\|}{\gamma_p} \right), \tag{1}$$

where ΔE is the Euclidean distance between pixel values in the CIELAB colour space, and the parameters γ_c and γ_p control grouping by similarity and proximity, respectively. Yoon and Kweon use default values of $\gamma_c = 5$ and $\gamma_p = 17.5$.

The aggregated cost space C' is now calculated using

$$C'(\mathbf{p}, d) = \frac{1}{k} \cdot \sum_{\mathbf{q} \in N_{\mathbf{p}}} w(\mathbf{p}, \mathbf{q}) \cdot w(\overline{\mathbf{p}}, \overline{\mathbf{q}}) \cdot C(\mathbf{q}, d)\,, \tag{2}$$

where $k = \sum_{\mathbf{q} \in N_{\mathbf{p}}} w(\mathbf{p}, \mathbf{q}) \cdot w(\overline{\mathbf{p}}, \overline{\mathbf{q}})$ is the normalisation quotient and $N_{\mathbf{p}}$ the set of all pixels in the support window. For the winner-take-all stage, we use Yang *et al.*'s sub-pixel refinement process [6]. We implemented all techniques in this paper using *C for CUDA*, an architecture for general purpose computation on NVIDIA GPUs. We measure run times on an NVIDIA Quadro FX 5800 GPU.

Our straightforward GPU implementation is about 25× faster than reported by Yoon and Kweon and produces comparable results to their publicly-available implementation (on the Middlebury website[4]). However, neither implementation achieves the results reported in the original paper. We believe this to be due to differences in filling in pixels that are invalidated by the left-right consistency check [7]. As we compare different techniques, it is fairest to only compare GPU techniques to other GPU techniques, and also to have all techniques share the same post-processing.

3 Dual-Cross-Bilateral Aggregation

The bilateral filter [8] is a common edge-preserving smoothing filter. One variant, the cross- or joint-bilateral filter [9], smoothes an image with respect to edges in a different image. Yoon and Kweon's technique is another variant that smoothes the cost space while preserving edges in both input images. In the bilateral filtering framework, we call this kind of filter a *dual-cross-bilateral filter* (DCB).

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

We reformulate their approach using Gaussian weights, the *de facto* standard in bilateral filtering. This yields

$$w(\mathbf{p},\mathbf{q}) = G_{\sigma_r}(\Delta E(\mathbf{p},\mathbf{q})) \cdot \sqrt{G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|)}\,, \tag{3}$$

where σ_r and σ_s are similarity and proximity parameters, and $G_\sigma(x) = \exp(\frac{-x^2}{2\sigma^2})$ is the unnormalised Gaussian centred on zero, with standard deviation σ. The square root is applied to the second factor, so that $w(\mathbf{p},\mathbf{q}) \cdot w(\overline{\mathbf{p}},\overline{\mathbf{q}})$ includes the proximity weight exactly once.

The aggregation remains unchanged from equation 2, resulting in

$$C'(\mathbf{p},d) = \frac{1}{k} \cdot \sum_{\mathbf{q}\in N_\mathbf{p}} G_{\sigma_r}(\Delta E(\mathbf{p},\mathbf{q})) \cdot G_{\sigma_r}(\Delta E(\overline{\mathbf{p}},\overline{\mathbf{q}})) \cdot G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|) \cdot C(\mathbf{q},d), \tag{4}$$

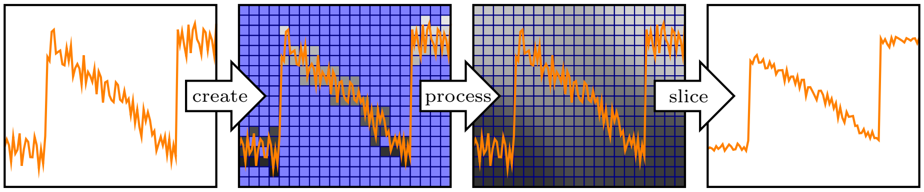
which we compute within a window of 35 × 35 pixels. We use the parameter values $\sigma_r = 10$ and $\sigma_s = 10$, which we found to produce the best results.

The resulting disparity maps are shown in table 2 and the Middlebury evaluation results in table 3. It is notable that our dual-cross-bilateral aggregation improves on our Yoon and Kweon implementation in the *nonocc* (non-occluded pixels) and *all* pixels categories in almost all cases.

3.1 Bilateral Grid

Full-kernel implementations of the bilateral filter are very slow, so several speed-up approaches have been proposed. A separable implementation [11] is too inaccurate for our purposes. Weiss' technique [12] only supports spatial box-filters rather than the Gaussians we use. And Yang *et al.*'s constant-time bilateral filtering [13] does not generalise well to higher dimensions, which we require. We therefore use the bilateral grid [10,14]. It has the interesting property that it runs faster and uses less memory as σ increases.

Consider the example of a greyscale image $I(x,y)$. The bilateral grid embeds it in a 3D space: 2D for spatial coordinates and 1D for pixel values. Each pixel (x,y) is mapped to $(x,y,I(x,y))$ in the bilateral grid Γ. The 1D example in figure 1 illustrates the use of the bilateral grid in three steps.



(a) input signal (b) grid created from signal (c) filtered grid (d) filtered signal

Fig. 1. Illustration of 1D bilateral filtering using the bilateral grid: the signal (a) is embedded in the grid (b), which is processed (c) and sliced to obtain the filtered signal (d). See text for details. Adapted from Chen *et al.* [10].

1. Grid Creation All grid voxels (x, y, c) are first zeroed using $\mathit{\Gamma}(x, y, c) = (0, 0)$. Then, for each pixel (x, y),

$$\mathit{\Gamma}\left(\left[\frac{x}{s_s}\right], \left[\frac{y}{s_s}\right], \left[\frac{I(x,y)}{s_r}\right]\right) \mathrel{+}= (I(x,y), 1)\,. \tag{5}$$

where $[\,\cdot\,]$ is the rounding operator, and s_s and s_r are the spatial and range sampling rates, which are set to σ_s and σ_r respectively. Note that the pixel values and the number of pixels are accumulated using homogeneous coordinates, which make it easy to compute weighted averages in the grid slicing stage.

2. Grid Processing The grid is now convolved with a Gaussian filter, of standard deviation σ_s and σ_r along the space and range dimensions. As the previous step has already sub-sampled the data accordingly, we only need to convolve each dimension with a 5-tap 1D kernel with $\sigma = 1$.

3. Grid Slicing The result is now extracted by accessing the grid coordinates $(x/s_s, y/s_s, I(x, y)/s_r)$ using trilinear interpolation, and dividing the homogeneous vector to access the actual data.

The bilateral grid is amenable to real-time GPU implementation, as demonstrated by Chen *et al.* [10].

3.2 Dual-Cross-Bilateral Grid

Chen *et al.* [10] show that the bilateral grid can also be used for cross-bilateral filtering. This is achieved by using an edge image $E(x, y)$ to determine grid coordinates, but storing the pixel values of the image $I(x, y)$ to be filtered:

$$\mathit{\Gamma}\left(\left[\frac{x}{s_s}\right], \left[\frac{y}{s_s}\right], \left[\frac{E(x,y)}{s_r}\right]\right) \mathrel{+}= (I(x,y), 1)\,. \tag{6}$$

The grid processing remains the same, and the slicing stage accesses the grid at $(x/s_s, y/s_s, E(x, y)/s_r)$.

Recall that our *dual-cross-bilateral* cost aggregation smoothes the cost space while preserving edges in the two input images. To implement it, we extend the bilateral grid to take into account both input images as edge images when calculating grid coordinates, and to accumulate cost space values instead of pixel values. We call our extension the dual-cross-bilateral grid, or *DCB grid*.

For a pixel $\mathbf{p}$ at (x, y) in the left image, and its corresponding pixel $\overline{\mathbf{p}}$ at $(x - d, y)$ in the right image, we create the DCB grid using

$$\mathit{\Gamma}\left(\left[\frac{x}{\sigma_s}\right], \left[\frac{y}{\sigma_s}\right], \left[\frac{L^\star_L(\mathbf{p})}{\sigma_r}\right], \left[\frac{L^\star_R(\overline{\mathbf{p}})}{\sigma_r}\right]\right) \mathrel{+}= (C(\mathbf{p}, d), 1)\,. \tag{7}$$

Instead of image intensities, as in Chen *et al.* [10], we use the lightness component $L^\star$ of the CIELAB colour space, as it is perceptually more uniform and hence more closely models how we perceive greyscale images. However, this also degrades performance compared to the full-kernel DCB, which uses full-colour images. The subscripts L and R indicate the left and right images, respectively.

The result of slicing the DCB grid is the aggregated cost

$$C'(\mathbf{p}, d) = \mathit{\Gamma}\left(\frac{x}{\sigma_s}, \frac{y}{\sigma_s}, \frac{L^\star_L(\mathbf{p})}{\sigma_r}, \frac{L^\star_R(\overline{\mathbf{p}})}{\sigma_r}\right). \tag{8}$$

In our implementation, we tile the 4D bilateral grids for all disparities into one large 2D texture. In the slicing stage, we perform the quadrilinear interpolation by using bilinear texture filtering to fetch the values stored at the surrounding four $([x/\sigma_s], [y/\sigma_s])$ coordinates, and bilinearly interpolate between them.

The run times in table 1 show that the DCB grid runs at 13 *fps* or higher on all data sets, with 70 *fps* on Tsukuba – more than 200× faster than the full-kernel implementation, and more than 165× faster than our GPU implementation of Yoon and Kweon. The disparity maps of all our techniques are shown in table 2 for visual comparison, and evaluated on the Middlebury datasets in table 3.

Table 1. Run time comparison in milliseconds. Our techniques, shown in bold, are benchmarked on an NVIDIA Quadro FX 5800. Asterisks (*) mark run times estimated from reported figures, rounded to one significant digit.

Technique	Tsukuba	Venus	Teddy	Cones
	384×288×16	434×383×20	450×375×60	450×375×60
DCB Grid	**14.2**	**25.7**	**75.8**	**75.0**
Real-time GPU [15]	30*	60*	200*	200*
Reliability DP [16]	42	109	300*	300*
Dichromatic DCB Grid	**188**	**354**	**1,070**	**1,070**
Plane-fit BP [17]	200*	400*	1,000*	1,000*
Y&K (our GPU impl.)	**2,350**	**4,480**	**13,700**	**13,700**
Full-kernel DCB	**2,990**	**5,630**	**17,700**	**17,600**
Yoon & Kweon [1]	60,000	100,000*	300,000*	300,000*

3.3 Dichromatic DCB Grid

The dramatic speedup achieved by the DCB grid comes at some loss of quality. This is because the underlying bilateral grid only works on greyscale images and hence does not differentiate colours that have similar greyscale values, as shown in the examples of figure 2b.

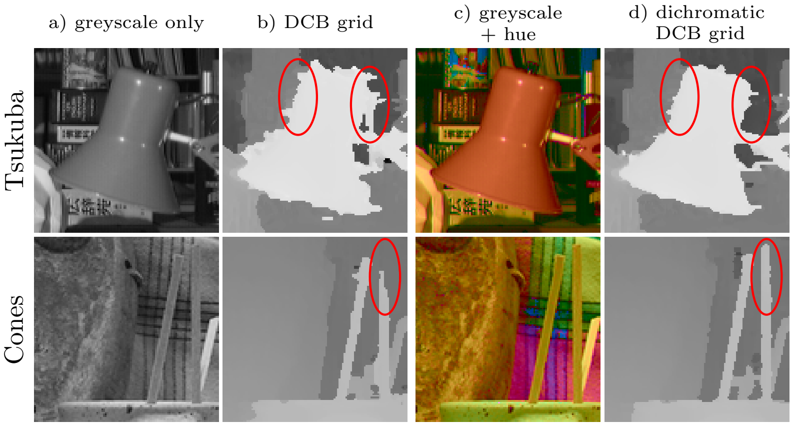


Fig. 2. Comparison of input images (a, c) and disparity maps of the (b) greyscale and (d) dichromatic DCB grids. The input images are displayed as ‘seen’ by the algorithms. Note that the dichromatic DCB grid (d) visibly improves on (b).

A solution is to add additional colour axes to the grid, to increase its colour discriminability. Unfortunately, the memory requirements of the bilateral grid are exponential in the number of dimensions. The *teddy* and *cones* data sets, for example, each have a total memory footprint of

$$60 \text{ disparities} \times \frac{450}{10} \times \frac{375}{10} \times \left(\frac{100}{10}\right)^k \times 8 \text{ bytes} \tag{9}$$

when using the standard parameters $\sigma_s = 10$ and $\sigma_r = 10$, k colour dimensions, and two single-precision floating-point numbers per grid cell. For the DCB grid, where $k=2$, this amounts to 78 MB. However, the best results, with full CIELAB colours in both images ($k=6$), would require a prohibitive 764 GB.

The maximum number of colour dimensions that can be afforded on current generation graphics cards is $k=3$ which equates to 783 MB for *teddy*. This allows one additional colour axis in one of the images, in addition to each image’s greyscale component. The result is a dichromatic technique which can differentiate colours along two colour axes. This is an interesting trade-off between the common monochromatic (greyscale) and trichromatic (*e.g.* RGB) stereo approaches, that has not previously been explored.

We experimented with several colour dimensions (table 4) and found that CIELAB hue h_{ab} provided the highest discriminability.

The results in tables 1 and 3 show that the dichromatic approach improves on the monochromatic DCB grid in all categories (except run time), achieving results comparable (*tsukuba*, *teddy*) or superior (*venus*) to our implementation of Yoon and Kweon, at a 13× speedup. The close-ups in figure 2 also show qualitative improvements.

Table 2. Disparity maps for the Middlebury data sets [5].

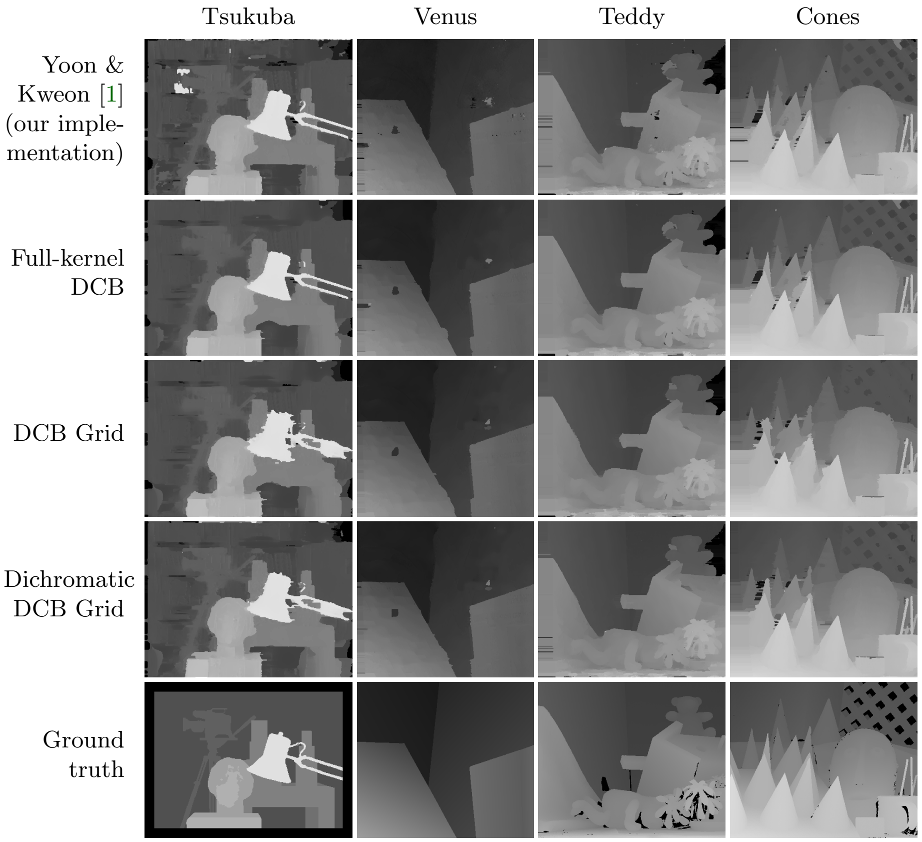


Table 3. Performance comparison of the proposed methods to Yoon & Kweon and selected real-time techniques using the Middlebury stereo benchmark.

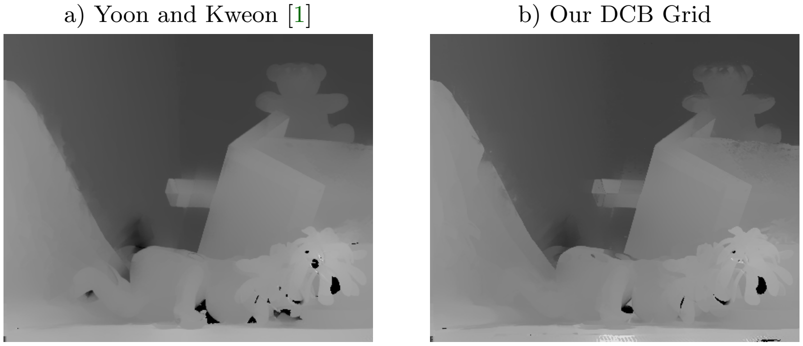
Technique	Rank	Tsukuba			Venus			Teddy			Cones		
		nonocc	all	disc	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc
Plane-fit BP [17]	19.4	0.97	1.83	5.26	0.17	0.51	1.71	6.65	12.1	14.7	4.17	10.7	10.6
Yoon & Kweon [1]	32.8	1.38	1.85	6.90	0.71	1.19	6.13	7.88	13.3	18.6	3.97	9.79	8.26
Full-kernel DCB	47.7	3.96	4.75	12.9	1.36	2.02	10.4	9.10	15.9	18.4	3.34	9.60	8.26
Y&K (our impl.)	48.2	4.39	5.29	8.10	1.30	2.07	8.31	9.39	16.3	18.4	3.68	9.96	8.42
Dichrom. DCB Grid	52.9	4.28	5.44	14.1	1.20	1.80	9.69	9.52	16.4	19.5	4.05	10.4	10.3
Real-time GPU [15]	56.2	2.05	4.22	10.6	1.92	2.98	20.3	7.23	14.4	17.6	6.41	13.7	16.5
Reliability DP [16]	59.7	1.36	3.39	7.25	2.35	3.48	12.2	9.82	16.9	19.5	12.9	19.9	19.7
DCB Grid	64.9	5.90	7.26	21.0	1.35	1.91	11.2	10.5	17.2	22.2	5.34	11.9	14.9

Table 4. Performance comparison of the dichromatic DCB grid with various colour properties using the Middlebury stereo benchmark. Judging by rank, as computed by the Middlebury website, the best technique is CIELAB hue, h_{ab}.

Technique	Rank	Tsukuba			Venus			Teddy			Cones		
		nonocc	all	disc	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc
$h_{ab} = \text{atan2}(b^\star, a^\star)$	**48.6**	**4.28**	5.44	14.1	1.20	1.80	9.69	9.52	16.4	19.5	4.05	**10.4**	**10.3**
HSL saturation	49.0	4.44	**5.37**	**12.9**	**1.05**	**1.58**	**8.29**	9.46	16.4	19.4	4.30	10.7	11.3
$C^\star_{ab} = \sqrt{a^{\star 2} + b^{\star 2}}$	49.9	4.97	5.94	16.7	1.15	1.75	8.65	9.55	16.4	19.9	**4.00**	**10.4**	10.5
$s_{ab} = C^\star_{ab}/L^\star$	50.0	4.36	5.45	12.9	1.19	1.86	9.32	**9.41**	**16.3**	**19.2**	4.41	10.8	11.6
$b^\star$	50.8	4.79	5.83	16.2	1.25	1.84	10.1	9.53	**16.3**	19.6	4.28	10.7	11.6
$a^\star$	52.0	5.36	6.49	18.3	1.24	1.84	9.13	9.62	16.5	19.9	4.28	10.5	11.3
HSL hue	51.2	4.62	5.85	14.9	1.30	1.87	10.4	9.83	16.6	20.2	4.18	10.7	11.1

3.4 Spatial-Depth Super-Resolution

Yoon and Kweon's method is also used in other contexts such as spatial-depth super-resolution. Yang *et al.* [6] use it as a central component in their system. Starting from a low-resolution depth map, they iteratively upsample it to the full resolution of the input images using Yoon and Kweon's cost aggregation. We use the same algorithm with our DCB grid and achieve a speedup of more than 100×. Figure 3 compares results, run times and errors.



	fps	nonocc	all	disc
a) Yoon and Kweon [1]	0.15	17.1	17.4	41.7
b) Our DCB Grid	17.0	19.9	19.2	42.5

Fig. 3. Comparison of Yoon & Kweon's and our cost aggregation techniques in Yang *et al.*'s spatial-depth super-resolution on 8× sub-sampled *teddy*. Our technique is more than 100× faster, at only a small loss of quality.

3.5 Temporal DCB Grid

Stereo videos pose different challenges to stereo images: the application of techniques on a per-frame basis is insufficient to achieve flicker-free and temporally coherent disparity maps. Given the success of the DCB grid method, we turned our attention to adding time as an extra dimension, inspired by approaches that aggregate costs over a 3D spatiotemporal support window [18,19]. Our experiments consider a time window of five frames, which we found to work well.

For each frame of the video, the DCB grid is created and processed as described in section 3.2, but the slicing is based on the grids of the last $n = 5$ frames, each weighted by w_i:

$$C'(\mathbf{p}, d) = \sum_{i=0}^{n-1} w_i \cdot \mathit{\Gamma}_i \left(\frac{x}{\sigma_s}, \frac{y}{\sigma_s}, \frac{L^{\star}_{L}(\mathbf{p})}{\sigma_r}, \frac{L^{\star}_{R}(\overline{\mathbf{p}})}{\sigma_r} \right), \tag{10}$$

where $i = 0$ indicates the current frame, $i = 1$ the previous frame and so on. The original spacetime stereo approaches use constant weights ($w_i = 1$). We use Gaussian weights, $w_i = \exp\left(-i^2/2\sigma_t^2\right)$ with $\sigma_t = 2$, which extends the DCB grid into the time dimension. We also tried Paris' adaptive exponential decay [20], but did not see any improvements compared to our simpler technique.

Note that we cannot use the dichromatic and temporal extensions at the same time, as we have insufficient memory to handle 6 dimensions of data (4 GB of GPU memory). Results of qualitative and quantitative nature are discussed next.

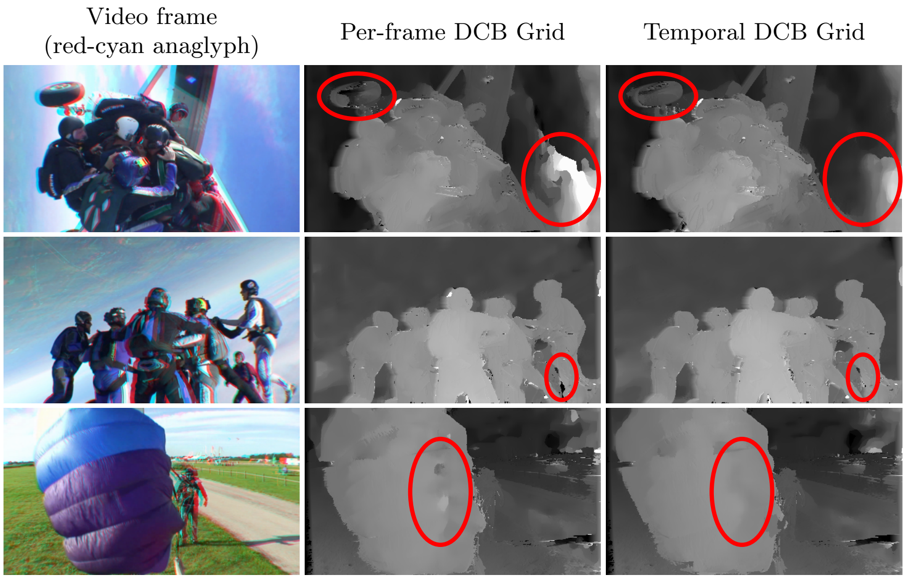
4 Results

All results in this paper were created using an NVIDIA Quadro FX 5800 GPU with 4 GB video memory, on a 2.4 GHz Intel Quad Core CPU with 4 GB RAM. Disparity maps created using our per-frame techniques are shown in table 2 and compared to other techniques in tables 1 and 3. Like Yoon and Kweon, we include left-right post-processing when reporting performance figures, but exclude it in run time measurements.

Our DCB grid is currently the fastest stereo correspondence approach on the Middlebury stereo evaluation website. A faster technique by Yang *et al.* [21] is not listed, as it has not been evaluated on the new Middlebury data sets, and we hence cannot compare to it fairly.

We improved the performance of the DCB grid using a dichromatic technique, drawing on a second colour axis to increase colour discriminability. Our results demonstrate that partial-colour solutions can improve stereo results, and we believe that this idea has more general applicability.

Tables 1 and 3 also show an interesting trade-off: both 'Real-time GPU' [15] and 'Reliability DP' [16] are slower than the DCB grid, but faster than the dichromatic DCB grid, with performance being inversely related: the dichromatic DCB grid outperforms both 'Real-time GPU' and 'Reliability DP' which in turn outperform the DCB grid. Yang *et al.*'s plane-fit BP [17] outperforms our dichromatic DCB grid at similar run times, but their technique occupies both CPU and GPU, whereas our techniques leave the CPU available for other tasks.



© Eric Deren, Dzignlight Studios.

Fig. 4. Disparity maps for selected frames of the 'skydiving' stereo video. Note that our temporal DCB grid visibly reduces errors (see highlighted regions).

4.1 Stereo Videos

We evaluated the temporal DCB grid qualitatively using real stereo videos and quantitatively on synthetic stereo videos with ground truth disparities, where we also compared it against per-frame techniques.

Qualitative Evaluation Figure 4 shows frames from the 'skydiving' video[5]. We processed it at a resolution of 480×270 with 40 disparities, without the left-right consistency check. On our machine, the per-frame DCB grid runs at 16 *fps* and the temporal DCB grid at 14 *fps*. As can be seen in the supplementary video, the temporal DCB grid visibly reduces flickering compared to the per-frame method.

Stereo Videos with Ground Truth Disparities The quantitative evaluation of disparity maps from stereo videos is hindered by the general lack of ground truth disparity maps. We created a set of five stereo videos with ground truth disparity maps (see figure 5) and make them available on our project website:

book	– turning a page of an old book (41 frames)
street	– camera pans across a street view (100 frames)
tanks	– camera flies along a grid of tanks (100 frames)
temple	– rotating Mayan temple (100 frames)
tunnel	– moving through a winding corridor (100 frames)

[5] http://www.dzignlight.com/stereo/skydiving.shtml

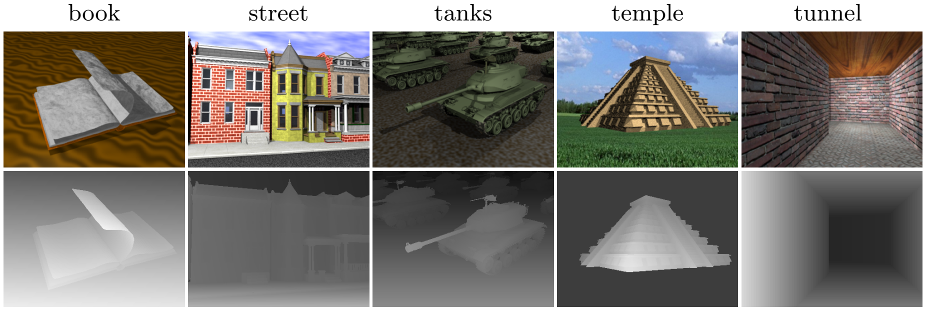


Fig. 5. Selected frames and disparity maps from our synthetic stereo videos.

We generated the sequences using Blender, an open source modeller. Each frame is 400×300 pixels in size with a disparity range of 64 pixels. The 'book', 'tanks' and 'temple' objects were taken from the Official Blender Model Repository[6], while the *tunnel* scene was our own design. For the 'street' sequence, we combined models and materials by Andrew Kator and Jennifer Legaz[7]. We added two parallel cameras to each scene with a small lateral offset between them, to provide the left and right views, and used the Blender node system to render disparity maps from the point of view of each camera.

Quantitative Evaluation We compared the temporal DCB grid against per-frame techniques using our synthetic ground truth videos. We processed all video frames by all techniques and used the same left-right consistency post-processing as earlier. Our ground truth stereo videos do not contain any noise, but real videos do. For this reason, we investigated the robustness to noise of per-frame techniques and the temporal DCB grid. We simulate thermal imaging noise by adding zero-centred Gaussian noise to all colour channels of the input frames.

The performance and run times of our implementations are shown in table 5. We summarise the level and variability of errors using the mean and standard deviation of the percentage of bad pixels across frames.

The best results are produced by the temporal DCB grid which significantly outperforms the per-frame techniques on all datasets except *tunnel*, on which it shows the least variation in error. Our per-frame DCB grid techniques come second and third, and our full-kernel implementations are placed last.

We believe that the poor performance of the temporal DCB grid on the *tunnel* video is because it has a lot of texture, so that simple per-frame approaches work well, while our temporal technique tends to over-smooth. Nevertheless, it reduces flickering visibly in all videos, as can be seen in the supplementary videos.

[6] http://e2-productions.com/repository/

[7] Licensed under CC-BY 3.0, available at http://www.katorlegaz.com/3d_models/.

Table 5. Performance comparison of the proposed methods on our synthetic stereo videos with additive Gaussian noise ($\sigma = 20$). Shown are the average and standard deviation of the percentage of bad pixels (threshold is 1), and per-frame run times. For most datasets, the temporal DCB grid has the least mean error.

Technique	Time	Book		Street		Tanks		Temple		Tunnel	
	in ms	mean	stdev	mean	stdev	mean	stdev	mean	stdev	mean	stdev
Temporal DCB Grid	90	**44.0**	2.02	**25.9**	**2.00**	**31.4**	6.06	**31.7**	1.82	36.4	**7.88**
DCB Grid	51	52.2	2.04	32.5	2.33	36.0	6.16	39.5	1.91	**25.7**	11.1
Dichromatic DCB Grid	782	58.9	1.83	39.2	2.62	47.8	12.0	43.0	1.73	32.9	12.0
Full-kernel DCB	13,200	65.9	1.45	49.1	3.13	53.5	6.15	52.0	**1.28**	43.0	11.7
Y&K (our impl.)	9,770	84.2	**1.24**	56.1	2.67	87.7	**2.01**	72.8	1.80	58.4	11.7

It is also notable that our temporal DCB grid has a run time that is sublinear in the number of frames: it only takes 76% longer than the per-frame DCB grid to process a five frames window instead of a single frame.

Plots of the error levels at standard deviations between 0 and 100 (out of 255) are shown in figure 6. The graphs show that the temporal DCB grid improves on the per-frame technique at increased noise levels in all cases. In particular, it is superior for all noise levels in the *street* and *temple* sequences, and starting from noise levels of 5–45 for the other sequences. We assume that it is the integration of temporal evidence across several frames that makes this improvement possible.

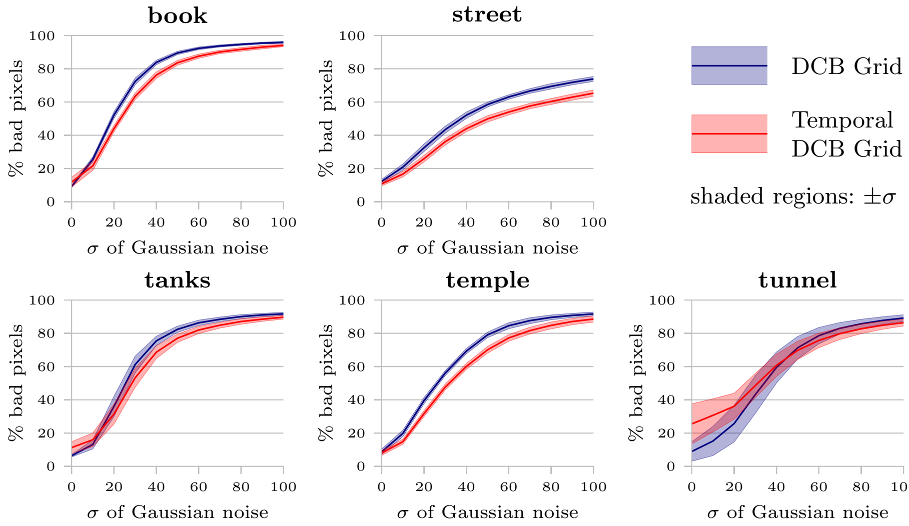


Fig. 6. Error versus noise curves for ground truth stereo videos: the temporal DCB grid performs better than the per-frame DCB grid at higher noise levels. Please also refer to the supplementary videos for a visual comparison.

5 Discussion

Rewriting Yoon and Kweon's adaptive support weights as a dual-cross-bilateral filter with Gaussian weights allows us to use the bilateral grid for acceleration, and to incorporate temporal information into the stereo matching process. Our DCB grid achieves real-time frame-rates through a speedup of more than 200× compared to a full-kernel GPU implementation, at only a small loss of precision. Our DCB grid is currently the fastest method on the Middlebury stereo website. The source code for our techniques, our ground truth stereo videos and further supplementary materials are available from our project website.

The speed of the DCB grid makes it versatile. Techniques building on Yoon and Kweon's method automatically benefit from a large speedup. We showed this by applying it to Yang *et al.*'s spatial-depth super-resolution, achieving a speedup of 100×, with minimal loss of quality.

Future Work

Using our dichromatic DCB grid, we showed that colour is a useful component in achieving high quality disparity maps. However, the enormous memory requirements of the bilateral grid effectively inhibit filtering in full colour. Recent work by Adams *et al.* [22] proposes a method with linear memory requirements. They agree that the bilateral grid is currently the fastest bilateral filtering technique for 4 dimensions when using a filter standard deviation of 10, as we do. However, full-colour filtering, using a total of 8 dimensions, would be about four times as fast with their technique, with significantly reduced memory requirements.

We hope that our new ground truth stereo videos provide a useful resource for research in depth estimation from stereo videos. There is a need for specialised stereo video correspondence techniques that incorporate temporal evidence to resolve ambiguities. With this in mind, it will be necessary to set up a stereo video evaluation website, perhaps as part of the Middlebury vision website.

In addition, we are interested in investigating suitable evaluation metrics for assessing stereo videos. We used the mean and standard deviation of the bad pixel percentage. However, it might be useful to consider other metrics that objectively quantify flickering and temporal coherence in disparity videos.

Acknowledgements

We are grateful to Andrew Fitzgibbon for helpful discussions as well as suggesting the temporal DCB grid extension. We further thank the anonymous reviewers for their valuable feedback, and NVIDIA for donating the Quadro graphics card through their CUDA Centre of Excellence at the University of Cambridge.

Christian Richardt and Ian Davies were supported by the Engineering and Physical Sciences Research Council (EPSRC). Douglas Orr was supported as an undergraduate research intern by Presenccia, an Integrated Project funded under the European Sixth Framework Programme (FP6-FET-27731).

References

1. Yoon, K.J., Kweon, I.S.: Adaptive support-weight approach for correspondence search. PAMI **28** (2006) 650–656
2. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. PAMI **23** (2001) 1222–1239
3. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient belief propagation for early vision. IJCV **70** (2006) 41–54
4. Gong, M., Yang, R., Wang, L., Gong, M.: A performance study on different cost aggregation approaches used in real-time stereo matching. IJCV **75** (2007) 283–296
5. Scharstein, D., Szeliski, R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. IJCV **47** (2002) 7–42
6. Yang, Q., Yang, R., Davis, J., Nistér, D.: Spatial-depth super resolution for range images. In: Proc. CVPR. (2007)
7. Egnal, G., Wildes, R.P.: Detecting binocular half-occlusions: Empirical comparisons of five approaches. PAMI **24** (2002) 1127–1133
8. Tomasi, C., Manduchi, R.: Bilateral filtering for gray and color images. In: Proc. ICCV. (1998) 839–846
9. Paris, S., Kornprobst, P., Tumblin, J., Durand, F.: A gentle introduction to bilateral filtering and its applications. In: SIGGRAPH Classes. (2008) Course material available online at http://people.csail.mit.edu/sparis/bf_course.
10. Chen, J., Paris, S., Durand, F.: Real-time edge-aware image processing with the bilateral grid. ACM Trans. Graph. **26** (2007) 103
11. Pham, T., van Vliet, L.: Separable bilateral filtering for fast video preprocessing. In: Proc. ICME. (2005)
12. Weiss, B.: Fast median and bilateral filtering. ACM Trans. Graph. **25** (2006) 519–526
13. Yang, Q., Tan, K.H., Ahuja, N.: Real-time $O(1)$ bilateral filtering. In: Proc. CVPR. (2009)
14. Paris, S., Durand, F.: A fast approximation of the bilateral filter using a signal processing approach. IJCV **81** (2009) 24–52
15. Wang, L., Liao, M., Gong, M., Yang, R., Nistér, D.: High-quality real-time stereo using adaptive cost aggregation and dynamic programming. In: Proc. 3DPVT. (2006) 798–805
16. Gong, M., Yang, Y.H.: Near real-time reliable stereo matching using programmable graphics hardware. In: Proc. CVPR. (2005) 924–931
17. Yang, Q., Engels, C., Akbarzadeh, A.: Near real-time stereo for weakly-textured scenes. In: Proc. BMVC. (2008)
18. Davis, J., Nehab, D., Ramamoorthi, R., Rusinkiewicz, S.: Spacetime stereo: a unifying framework for depth from triangulation. PAMI **27** (2005) 296–302
19. Zhang, L., Snavely, N., Curless, B., Seitz, S.M.: Spacetime faces: high resolution capture for modeling and animation. ACM Trans. Graph. **23** (2004) 548–558
20. Paris, S.: Edge-preserving smoothing and mean-shift segmentation of video streams. In: Proc. ECCV. (2008) 460–473
21. Yang, R., Pollefeys, M., Li, S.: Improved real-time stereo on commodity graphics hardware. In: Proc. CVPR Workshops. (2004) 36–36
22. Adams, A., Baek, J., Davis, A.: Fast high-dimensional filtering using the permutohedral lattice. Comp. Graph. Forum **29** (2010) 753–762