Supplementary Material
CBMV: A Coalesced Bidirectional Matching Volume for Disparity Estimation

Konstantinos Batsos kbatsos@stevens.edu
Changjiang Cai ccai1@stevens.edu
Philippos Mordohai mordohai@cs.stevens.edu
Stevens Institute of Technology

In this Supplementary Material, we include more visualizations of our method, as well as details and results omitted in the main text.

Matching Volume Computation Our method begins by computing four individual matching functions: CENSUS, NCC, ZSAD and SOBEL. The matching windows were determined experimentally. The window combinations we found that work best are: 11×11 for CENSUS, 3×3 for NCC and 5×5 for ZSAD and SOBEL. The choice of the 3×3 NCC window might seem odd, since the winner-take-all disparity maps are noisy. Our priority, similar to [3], is to generate a matching volume that is amenable to optimization. A lower initial error rate does not guarantee the lowest possible error rate after cost-optimization. In fact, we found that the 3×3 NCC window led to better results after optimization and post-processing than larger windows.

Figure 5 shows the left and right views of the Piano and PlaytableP scenes taken from the Middlebury 2014 dataset with corresponding disparity maps. The left and right disparity maps are generated from CBMV before optimization and post-precessing.

Figure 6 shows the final disparity maps computed after optimization and post-processing.

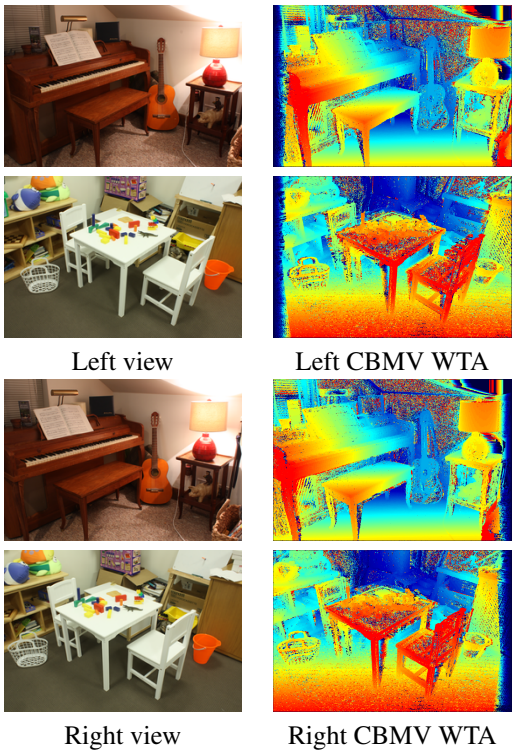


Figure 5. Left and right views of Piano and PlaytableP scenes of the Middlebury 2014 dataset. The disparity maps are computed in a winner take all fashion on the raw CBMV.

Volumetric Confidence Estimation (Video Included) During the confidence estimation process, we extract confidence measures for every pixel and every disparity under consideration. To better illustrate these volumes, we include a video of the volumetric cost and confidence maps computed for Census and the final CBMV for the Piano scene of the Middlebury 2014 dataset. The video can be viewed using the VLC media player on all operating systems [1].

The video is gray scale. Each frame is a constant disparity slice of a cost or confidence volume of the final CBMV volume. Dark regions denote areas with low confidence while bright regions denote areas with high confidence. The left dark margin that grows bigger as time elapses includes pixels that are out of the field of view of the right image as disparity values increase. The white curve that goes through the scene denotes pixels for which the feature value is highest at the given disparity level. Constant high or low values for the entire duration of the videos represent ambiguous areas without texture. As an example, at the top right corner of the Piano scene the intensity of all pixels is constant, thus the volumes are roughly uniform across disparity in that area.

The confidence measures we use are computed using ratios of cost values, and transformations of the entire cost curve for a pixel. They capture properties that are not readily available to a CNN operating on image patches and reduce ambiguity, as can be seen in the video.

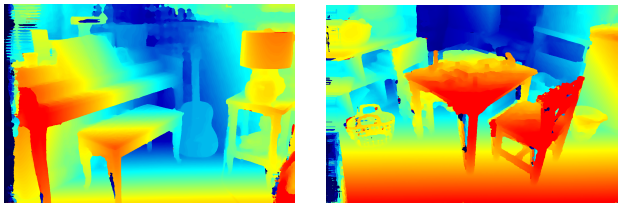


Figure 6. Final disparity maps, for the left view of the Piano and PlaytableP scenes of the Middlebury 2014 dataset, generated after optimization and post-precessing.

dataset	Middlebury	KITTI 2012	KITTI 2015
SGM parameters			
π_1	1.3	1.1	2.4
π_2	10	27	24.25
q_1	2.2	3	3.3
q_2	3	2	1.4
$alpha$	1.4	2	2.9
τ_{so}	0.12	0.08	0.16
CBCA parameters			
L_1	6	5	3
τ_1	0.07	0.13	0.06
$cbca_{i1}$	0	1	4
$cbca_{i2}$	4	0	4
Median filter			
$media_w$	3	5	9
$median_i$	1	1	1
Bilateral filter			
$blur_\sigma$	2.2	7	5
$blur_\tau$	1.3	5	4.75

Table 4. Cost-optimization and post-processing hyper-parameters for the three dataset, Middlebury 2014, KITTI 2012 and KITTI 2015

Optimization and Post-processing In this section we give details about the cost-optimization and post-processing pipeline along with hyper-parameters excluded from the main paper. The hyper parameter settings change depending on the dataset as in [3]. The pipeline follows the one proposed by [3] and is depicted in Fig. 2. For a more detailed description of the building blocks and hyper-parameters we refer the reader to the work of Mei et al. [2]. We follow a similar notation to avoid confusion. A complete list of our hyper-parameter values is shown in Table 4.

The SGM hyper-parameters are:

- π_1 cost penalty, for disparities deviating by ± 1.
- π_2 cost penalty, for larger disparity changes.
- τ_{so} color change difference. This threshold is used to reduce the penalties π_1 and π_2 close to object boundaries.
- q_1 , q_2 are the penalty reduction coefficients, such that: π_1, π_2, if $D_1 < \tau_{so}, D_2 < \tau_{so}$.
  $\pi_1 = \pi_1/(q_1 q_2), \pi_2 = \pi_2/(q_1 q_2)$ if $D_1 > \tau_{so}, D_2 > \tau_{so}$.
  $\pi_1 = \pi_1/q_1, \pi_2 = \pi_2/q_1$ otherwise.
  D_1 and D_2 denote the color intensity difference in the horizontal and vertical directions respectively.
- $alpha$ is the π_1 reduction coefficient for the vertical directions [3].

The CBCA hyper-parameters are:

- L_1 is the maximum size of the cross arms in the vertical and horizontal directions.
- τ_1 is the threshold that controls the cross size based on the image intensity difference.
- $cbca_{i1}$ number of CBCA iterations before SGM.
- $cbca_{i2}$ number of CBCA iterations after SGM.

The median filter hyper-parameters are:

- $median_w$ the size of the median filter window.
- $median_i$ the number of iterations.

The bilateral filter hyper-parameters are:

- $blur_\sigma$ is the standard deviation of the normal distribution used in the kernel computation.
- $blur_\tau$ is the disparity difference threshold. When disparities deviate more than this threshold, they are not considered in the filtering process.

For the bilateral filter a Gaussian kernel is used. The size of the blur window is computed as $2\lceil 3blur_\sigma \rceil + 1$.

Ablation Studies To determine the hyper-parameters for our method and how each component affects the performance and end result, we conducted a number of ablation studies. We have included some representative results here since including all results would lead to clutter.

A number of matching function combinations was attempted. We show the validation error rates on KITTI 2012 with constant hyper-parameters. Using Census+NCC the error is 4.91%, Census+NCC+ZSAD the error is 4.76%, Census+NCC+ZSAD+Sobel (final) the error is 4.36%. Changing the confidence measures leads to the following error rates on the KITTI 2015 validation set: no ratio or likelihood 5.29%, likelihood only 5.02%, both ratio and likelihood (final) 4.78%.

The post-processing steps after SGM cost optimization result in the following improvements. For KITTI 2012 the error decreases from 5.2% to 4.33%; for KITTI 2015, from 5.68% to 4.78%; and for Middlebury 2014 the error decreases form 12.1% to 11.7%.

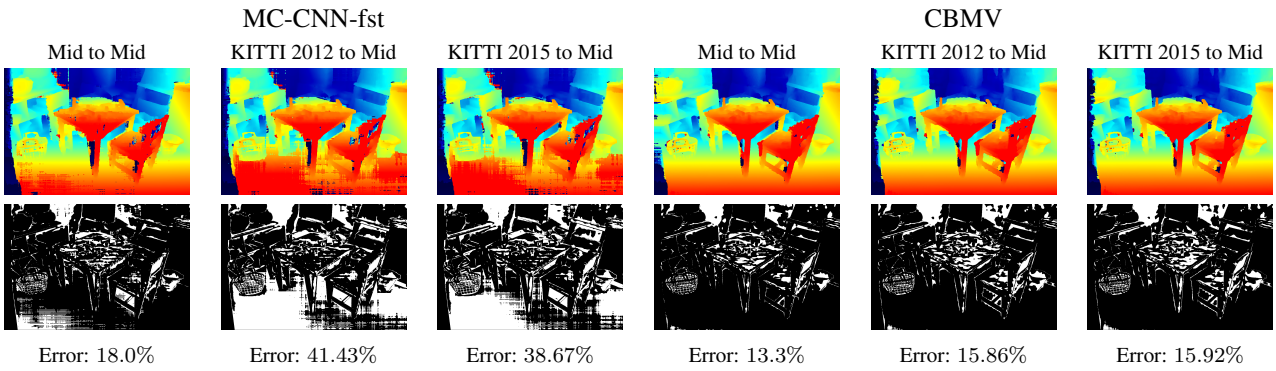


Figure 7. MC-CNN-fst vs CBMV generalization experiment. Mid stands for Middlebury. Dataset A to Dataset B denotes that the model was trained on Dataset A and tested on Dataset B. Top row: qualitative results of MC-CNN-fst and CBMV on the Playtable scene of Middlebury 2014 dataset. Bottom row: Corresponding error maps with quantitative results.

Generalization MC-CNN is the current state of the art method in terms of cost computation. MC-CNN's greatest advantage is that it learns how to match small image patches in a purely data-driven manner. However this advantage is also a weakness. MC-CNN directly exposes the learning algorithm to the training images, thus it overspecializes to the training domain. In our method the learning algorithm is exposed to the initial estimates of generic block matching algorithms. Figure 7 shows quantitative and qualitative results of MC-CNN-fst and CBMV for the Playtable scene of Middlebury 2014 dataset. Part of this figure is Fig. 4 in the main paper. During this experiment the *only change* was the model with which the cost volume was computed. The cost-optimization and disparity refinement parameters were kept the same. As an example, for the Middlebury dataset we used the hyper-parameters that we would use if we were training and testing on the Middlebury dataset. The MC-CNN models were downloaded from the web, and are publicly available on the official github repository of MC-CNN. Figure 8 shows generalization results of CBMV on the KITTI 2012 and KITTI 2015 benchmarks.

References

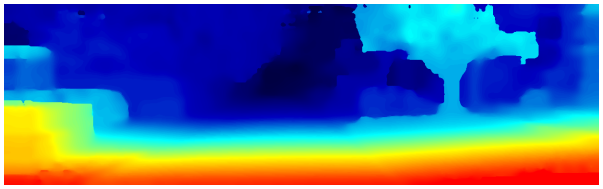
[1] VideoLAN VLC media player. https://www.videolan.org/vlc/index.html. 1

[2] X. Mei, X. Sun, M. Zhou, S. Jiao, H. Wang, and X. Zhang. On building an accurate stereo matching system on graphics hardware. In *ICCV Workshops*, pages 467–474, 2011. 2

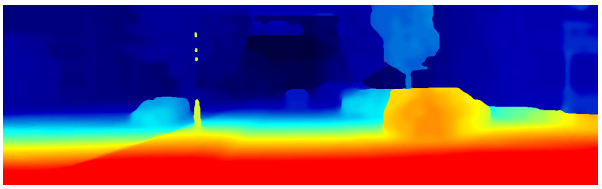
[3] J. Žbontar and Y. LeCun. Stereo matching by training a convolutional neural network to compare image patches. *The Journal of Machine Learning Research*, 17(65):1–32, 2016. 1, 2

Left image, KITTI 2012

Left image, KITTI 2015



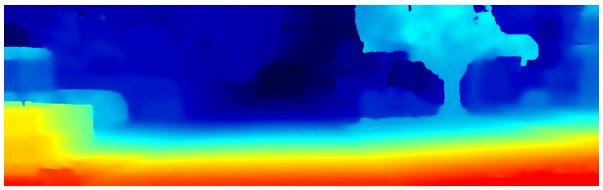
Mid to KITTI 2012



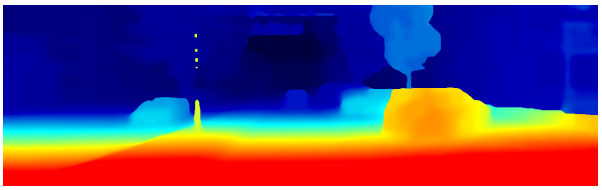
Mid to KITTI 2015

Error: 1.73%

Error: 1.48%



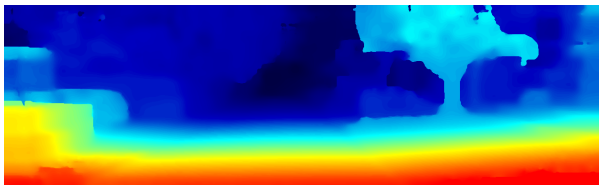
KITTI 2012 to KITTI 2012



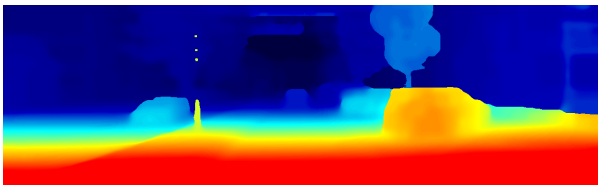
KITTI 2012 to KITTI 2015

Error: 1.99%

Error: 1.45%



KITTI 2015 to KITTI 2012



KITTI 2015 to KITTI 2015

Error: 2.10%

Error: 1.46%

Figure 8. CBMV generalization experiment on KITTI 2012 and KITTI 2015. Mid stands for Middlebury. Qualitative and quantitative results show that out method can generalize well across datasets. During this experiment we *only* changed the learned models, and we *did not* perform any additional hyper-parameter tuning.