Supplementary material for “Real-time self-adaptive deep stereo”

Alessio Tonioni, Fabio Tosi, Matteo Poggi, Stefano Mattoccia, Luigi di Stefano
Department of Computer Science and Engineering (DISI)
University of Bologna, Italy
{alessio.tonioni, fabio.tosi5, m.poggi, stefano.mattoccia, luigi.distefano }@unibo.it

1. Detailed *MADNet* structure

We report here a detailed description of the three modules that compose *MADNet*. We start from Fig. 1 depicting details of our pyramidal convolutional feature extractor. *MADNet* will deploy two of them with parameter sharing to extract features independently on the left and right frames (green and red pyramid in Figure 2 (a) in the main paper).

Then, for each one of the 6 resolutions considered in *MADNet*, we build one disparity estimation module as described in Fig. 2. Let $\mathcal{D}_n$ be a disparity estimation module for resolution n. The first operation performed is the computation of a cost volume (*i.e.*, correlation layer[4]) between corresponding convolutional features at the same resolution extracted from the left and right image ($\mathcal{F}_n^l$ and $\mathcal{F}_n^r$ respectively). If a lower resolution disparity is available (*e.g.*, $\mathcal{D}_{n+1}$), the features from the right image can be partially aligned to the one on the left before the cost volume computation by using a warping layer [5]. With this strategy we aim to encode in the cost volume useful information for the refinement of the lower resolution disparity $\mathcal{D}_{n+1}$. Then the final input to the module is obtained by concatenating the cost volume obtained and the up-scaled lower res disparity. At the lowest resolution in our network ($\mathcal{D}_6$) we ignore the warping and up-sampling steps (since we do not have a $\mathcal{D}_7$) and directly create a cost volume between $\mathcal{F}_6^l$ and $\mathcal{F}_6^r$. At the highest resolution considered by our model ($\mathcal{D}_2$) we deploy a residual refinement module, depicted in Fig. 3. Here we use atrous convolutions and residual connections to get the final disparity estimation at the same resolution as its input. To recover full resolution, we up-sample the output of the refinement module using bi-linear interpolation.

2. Implementation and training details for *MADNet* and *MAD*

We resume implementation details regarding how we have initially trained *MADNet* on synthetic data, how we split the network into independent portions and how we compute the self-supervised loss to train them online.

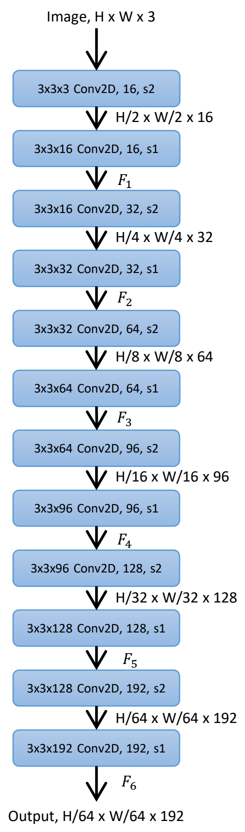

Figure 1. Detailed structure of our convolutional feature extractor. For each convolutional layer, we report the kernel dimensions and the stride as s followed by the stride step.

Pre-Training: Regarding the initial training of *MADNet*, we perform 1200000 training iterations on the FlyingThings3D dataset using Adam Optimizer and a learning rate of 0.0001, halved after 400k steps and further every 200k until convergence. As loss function, we compute the sum of per-pixel absolute errors between disparity maps estimated at each resolution and downsampled groundtruth labels. The final loss is a weighted sum of the contributions from the different resolutions. The weights used are respec-

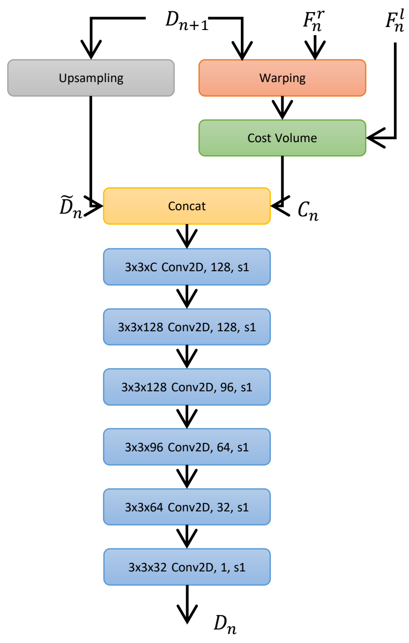

Figure 2. Detailed structure of our stereo estimation network at resolution n. We denote with $\mathcal{F}_n$ the convolutional features extracted at resolution n, superscript l for those obtained from the left frame and r for those referring to the right one. For the upsampling block, we use standard bilinear upsampling while for the cost volume creation we use the correlation layer introduced in DispNetC [4] with max displacement 2.

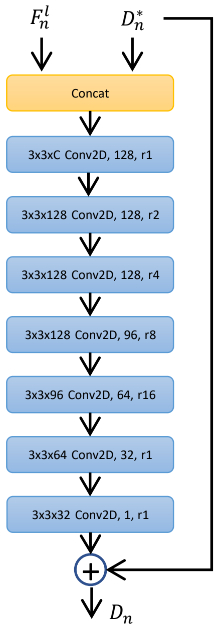

Figure 3. Detailed structure of our residual refinement network. The network takes as input an initial disparity estimation $\mathcal{D}_n^*$ and the corresponding left convolutional features at the same resolution $\mathcal{F}_n^l$, then uses atrous convolutions to elaborate them (we report the rate used as r followed by the rate value) and finally outputs a residual pixel-wise correction.

tively 0.005, 0.01, 0.02, 0.08, 0.32 from $\mathcal{D}_2$ to $\mathcal{D}_6$ following [5]. For the additional fine tuning on the KITTI 2012 and 2015 training sets used in section 4.2 of the submitted paper, we performed 500K optimization steps by computing the loss only on the full-resolution disparity map and using 0.001 as weight. All the other hyperparameters are kept as in the synthetic training.

Adaptation Concerning *MAD*, we have grouped layers (either from the feature extractor or the disparity estimators) according to the resolution at which they operate, *i.e.* ($\mathcal{F}_i$,$\mathcal{D}_i$) composes a module $\mathcal{M}_i$. We made this decision because in our architecture layers at the same resolution are directly connected through skip connections which may allow approximate backpropagation by flowing the gradients only through the connection without traversing the low-resolution layer in between. For example, considering the structure of the disparity estimation module depicted in Fig. 2, we would backprop only through $\mathcal{F}_n^l$ and $\mathcal{F}_N^r$ and not through $\mathcal{D}_{n+1}$. By following this strategy, we obtain 5 independently trainable portions of *MADNet* by grouping layers that produce $[\mathcal{D}_k, \mathcal{F}_k]$ for each one of the resolutions from 6 to 3. For the higher resolution portions (1-2) we collapsed together layers working at the half and quarter resolution (*i.e.*, $\mathcal{F}_1$, $\mathcal{F}_2$, $\mathcal{D}_2$ and the refinement module). Each independent portion of *MADNet* can be trained independently since each one can produce a disparity estimation amenable for loss computation.

We use as loss function for the adaptation the photometric consistency between the left RGB frame of the stereo couple and the right one reprojected according to the predicted disparity. Following [2], we perform the image reprojection using a fully differential bilinear sampler, then compare the two images using a linear combination of SSIM [6] computed on 3×3 patches and L_1 distance. The contribution of the two component are respectively weighted 0.85 and 0.15. We did not use the left-right consistency check proposed in [2] as it would need to elaborate each stereo pair twice, drastically reducing the frame rate of the system without improving the performance considerably. Finally, we run some experiments adding the smoothness term proposed in [2], without getting noticeable improvement. Therefore, we decided to omit it to keep our formulation simpler. For all our tests, every loss function is computed at full resolution by upsampling the small-scale predictions using bilinear sampling. All the code used for our experiments is available [1].

[1] https://github.com/CVLAB-Unibo/Real-time-self-adaptive-deep-stereo

Algorithm 1 Online Adaptation with *MAD*

```
1:  Require: $\mathcal{N} = [n_1, \dots, n_p]$
2:  $\mathcal{H} = [h_1, \dots, h_p] \leftarrow 0$
3:  while not stop do
4:      $x \leftarrow readFrames()$
5:      $[y, y_1, \dots, y_p] \leftarrow forward(\mathcal{N}, x)$
6:      $\mathcal{L}_t \leftarrow loss(x, y)$
7:      $\theta \leftarrow sample(argsoftmax(\mathcal{H}))$
8:      $\mathcal{L}_t^\theta \leftarrow loss(x, y_\theta)$
9:      $updateWeights(\mathcal{L}_t^\theta, n_\theta)$
10:     if $firstFrame$ then
11:         $\mathcal{L}_{t-2} \leftarrow \mathcal{L}_t, \mathcal{L}_{t-1} \leftarrow \mathcal{L}_t, \theta_{t-1} \leftarrow \theta$
12:     end if
13:     $\mathcal{L}_{exp} \leftarrow 2 \cdot \mathcal{L}_{t-1} - \mathcal{L}_{t-2}$
14:     $\gamma \leftarrow \mathcal{L}_{exp} - \mathcal{L}_t$
15:     $\mathcal{H} \leftarrow 0.99 \cdot \mathcal{H}$
16:     $\mathcal{H}[\theta_{t-1}] \leftarrow \mathcal{H}[\theta_{t-1}] + 0.01 \cdot \gamma$
17:     $\theta_{t-1} \leftarrow \theta_t, \mathcal{L}_{t-2} \leftarrow \mathcal{L}_{t-1}, \mathcal{L}_{t-1} \leftarrow \mathcal{L}_t$
18: end while
```

3. Detailed algorithm for one online adaptation step using *MAD*

Alg. 1 provides detailed pseudocode for online adaptation with *MAD* using our proposed sampling heuristics.

We start by creating a histogram $\mathcal{H}$ with p bins, *i.e.* one per module, all initialized at 0. For each stereo pair we perform a forward pass to get the disparity predictions (line 5) and measure the performance of the model by computing the loss $\mathcal{L}_t$ according to the full resolution disparity y and, potentially, the input frames x, *e.g.*, reprojection error between left and right frames [2] (line 6). Then, we pick the portion to train $\theta \in [1, \dots, p]$ by sampling from the probability distribution obtained as $softmax(\mathcal{H})$ (line 7). Once selected, we compute one optimization step for layers of $\mathcal{M}_\theta$ with respect to the loss L_t^θ computed on the lower scale prediction y_θ (line 8-9). We have now partially adapted the network to the current environment. Next, we update $\mathcal{H}$ increasing the probability of being sampled for the $\mathcal{M}_i$ that have proven effective. To do so, we can compute a noisy expected value for $\mathcal{L}_t$ by linear interpolation of the losses at the previous two time step: $L_{exp} = 2 \cdot \mathcal{L}_{t-1} - \mathcal{L}_{t-2}$ (line 13). By comparing it with the measured $\mathcal{L}_t$ we can assess the impact of the network portion sampled at the previous step (θ_{t-1}) as $\gamma = L_{exp} - \mathcal{L}_t$, and then increase or decrease its sampling probability accordingly (*i.e.*, if the adaptation was effective $\mathcal{L}_{exp} > \mathcal{L}_t$, thus $\gamma > 0$). We found out that adding a temporal decay to $\mathcal{H}$ helps increasing the stability of the system, so the final update rule for each step is: $\mathcal{H} = 0.99 \cdot \mathcal{H}, \mathcal{H}[\sigma_{\tau-1}] += 0.01 \cdot \gamma$ (lines 15 and 16).

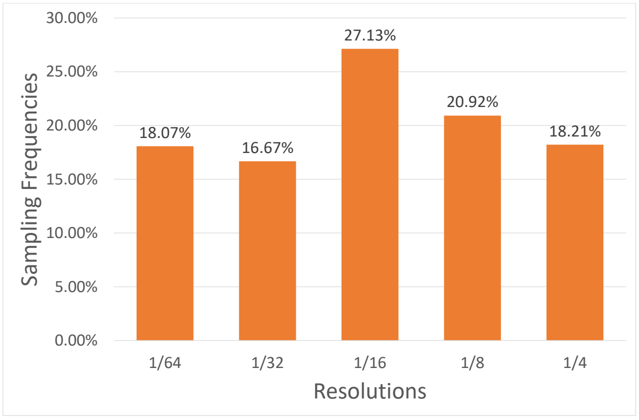

Figure 4. Sampling frequencies for the different independent portions of *MADNet* using *MAD* for fast online adaptation.

4. *MAD* sampling policy

We are interested in investigating which portions of the network are trained more by *MAD* according to the reward-punishment mechanism we designed. To get some insights we ran *MADNet* performing adaptation with *MAD* on the full KITTI raw dataset 5 times and kept track of the number of steps on which each portion has been sampled for training. In Fig. 4 we report on the y-axis the average sampling frequencies for each portion, whereby on the x-axis we identify each of the portions by the different scale at which it operates. Surprisingly the most sampled portion (*i.e.*, the one that according to our heuristic will grant the greater improvement once adapted) is not the last that produces the final predictions (*i.e.*, $\frac{1}{4}$), but the middle portion of the network (*i.e.*, $\frac{1}{16}$). As pointed out by [3], a good coarse disparity map can be easily up-sampled and refined to an accurate full resolution output. This is further confirmed by our analysis, showing how *MAD* favors the fine-tuning of lower resolution modules(*i.e.*, $\frac{1}{16}$). This behaviour might be closely linked to the architecture of *MADNet* that starts from a low resolution disparity estimation and iteratively refine it. If the low resolution disparity has major mistakes the upper modules are not able to solve them properly, thus training more the lower resolution modules might be preferable.

5. Qualitative comparison between fast networks

Fig. 5 provides additional qualitative comparisons between the output of three fast stereo architecture: DispNetC [4], StereoNet [3] and our *MADNet*. We wish to highlight how *MADNet* better maintains thin structures compared to StereoNet.

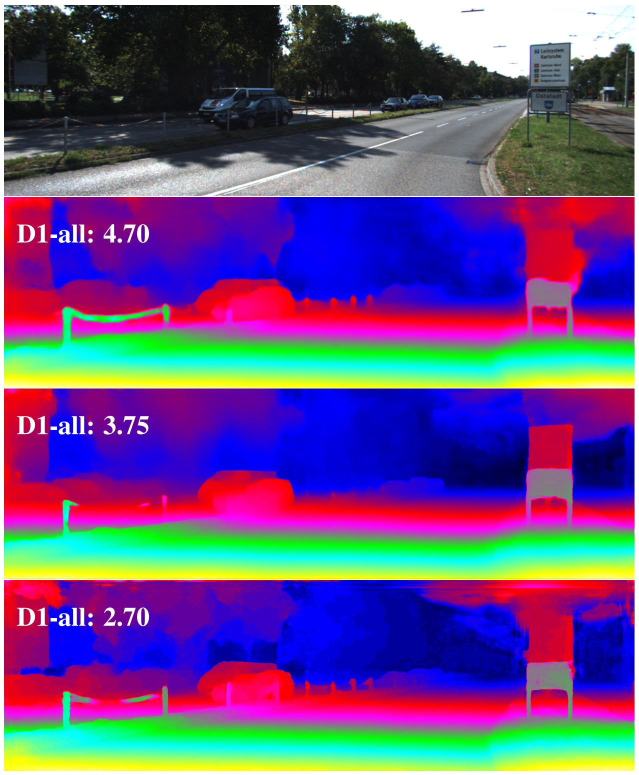

Figure 5. Qualitative comparison between disparity maps from different architectures. From top to bottom, reference image from KITTI 2015 online benchmark and disparity map by DispNetC [4], StereoNet [3] and *MADNet*.

6. Qualitative Results on Online Adaptation

As a further supplementary material, we refer the reader to a video showing the effectiveness of our online adaptation formulation in the two different configurations (Full and *MAD*), available at https://www.youtube.com/watch?v=7SjyzDxmCY4. The colormap used encodes with bright color points close to the camera (*i.e.*, high disparity) and with dark colors points far from the camera (*i.e.*, low disparity). We visualize in the upper right corner the predictions performing only inference at 40FPS, in the lower left performing fast adaptation using *MAD* at 25 FPS and in the lower right performing adaptation of the whole network using full back-prop at 15 FPS. For better visualization, we have synchronized the output of the three networks, so the video is not showing real execution times. To give a quantitative measurement of the improvement, we have superimposed over each image in the top left corner the D1-ALL and EPE metrics. For the two adapting networks we also report, between brackets, the gain compared to the same network without adaptation.

For the first half of the video we have select a video sequence from the KITTI raw dataset belonging to the *Residential* environment. The video clearly shows how online adaptation of the whole network by full backprop can solve most of the mistakes in as few as ~ 150 frames (*i.e.*, $10s$ of execution time at $15FPS$). Fast adaptation using *MAD*, instead, requires slightly more frames to achieve comparable improvements (*i.e.*, about ~ 400 frames or $16s$ at $25FPS$), but then can still benefit from the higher frame rate. Finally, we can see how going towards the end of the sequence the gap between the adaptation of the whole network by full backprop and *MAD* gets smaller and smaller up to converge to comparable performance in the final ~ 500 frames.

The second half of the video concerns performance achieved in an indoor scenario. For this qualitative evaluation we select a sequence from the Wean Hell dataset [1]. We can see how both adaptation strategies can drastically improve the network that does not perform adaptation. As in the outdoor scenario, the full adaptation requires less frame to achieve good performance while *MAD* needs slightly more frames. By the end of the video, both networks produce similar smooth predictions.

Finally, to better highlight some differences between the two adaptation methods we report on Fig. 6 and Fig. 7 some selected frames from the videos. On the left most column we show for each row the index of the frame in the sequence considered. For each example we show the disparity predicted by three different configuration of *MADNet*: without online adaptation, with online adaptation by full back-prop and with our computationally efficient *MAD*.

Fig. 6 shows predictions obtained on a sequence from the KITTI dataset. With as few as 100 frame *Full Adaptation* is able to resolve most of the mistakes in the predicted disparity, while *MAD* at the same iteration is only able to slightly decrease the magnitude of the mistake. Around the 500^{th} iteration (row 2), *MAD* start to improve drastically, with the predicted disparity showing way less mistakes. The same trends is visible in the following rows with *MAD* rapidly closing the performance gap with respect to *Full Adaptation*. By frame 2000 (*i.e.*, after 2000 step of online adaptation) both adaptation techniques converge to similar predictions

Fig. 7 shows predictions obtained on a sequence from the indoor Wean Hall dataset [1]. Even in this scenario both the adaptation mode are able to drastically increase the quality of the predicted disparity maps with relatively few considered steps. In particular by the 500^{th} frame the *Full Adaptation* has already adapted the model to the current environment as shown by the absence of macro mistakes in the predicted disparities. *MAD*, instead, needs more iterations, but once again can converge to performance comparable to full adaptation after around 1500 frames (rows 4 and 5 in Fig. 7). The frames reported in the last two rows show challenging scenes where the re-projection loss that we use for adaptation may fail to produce useful gradients due to the big reflections of the neon lights on the floor that will be

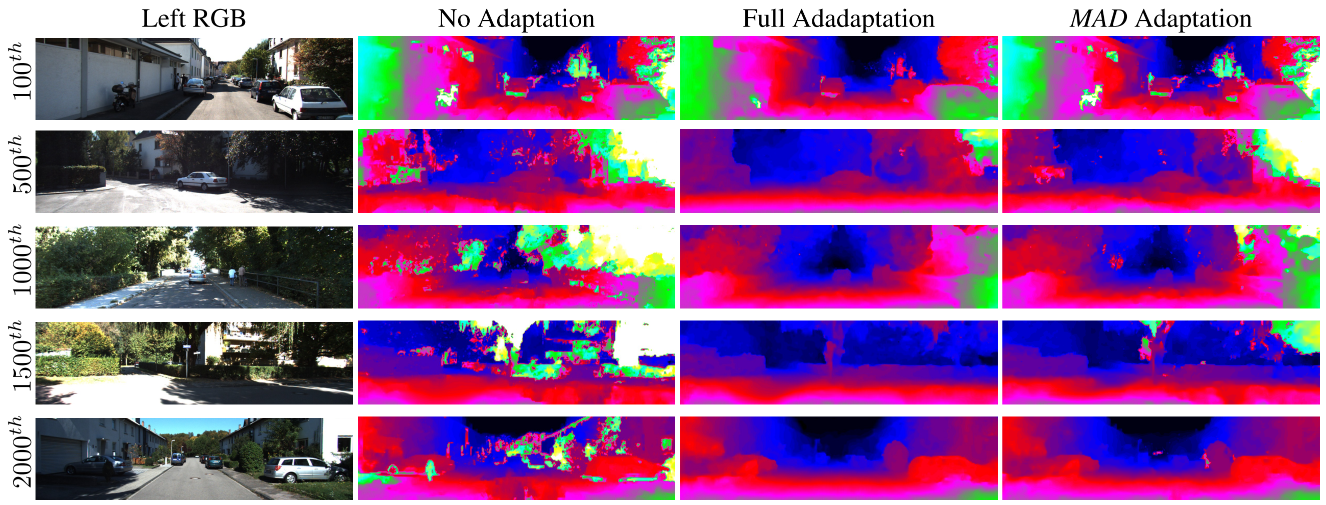

Figure 6. Comparison between predicted disparities obtained by *MADNet* with different adaptation modalities. The leftmost side of the table report the number of elaborated step in the sequence.

viewed differently by the left and right camera. We can see how all the three models fail to produce good prediction in this challenging situation, we plan to address this kind of challenging situation in the future by relying on multiple unsupervised losses.

References

[1] Hatem Alismail, Brett Browning, and M Bernardine Dias. Evaluating pose estimation methods for stereo visual odometry on robots. In *the 11th International Conference on Intelligent Autonomous Systems (IAS-11)*, 2011. 4

[2] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, volume 2, page 7, 2017. 2, 3

[3] Sameh Khamis, Sean Fanello, Christoph Rhemann, Adarsh Kowdle, Julien Valentin, and Shahram Izadi. Stereonet: Guided hierarchical refinement for real-time edge-aware depth prediction. In *15th European Conference on Computer Vision (ECCV 2018)*, 2018. 3, 4

[4] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 1, 2, 3, 4

[5] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2

[6] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *Trans. Img. Proc.*, 13(4):600–612, Apr. 2004. 2

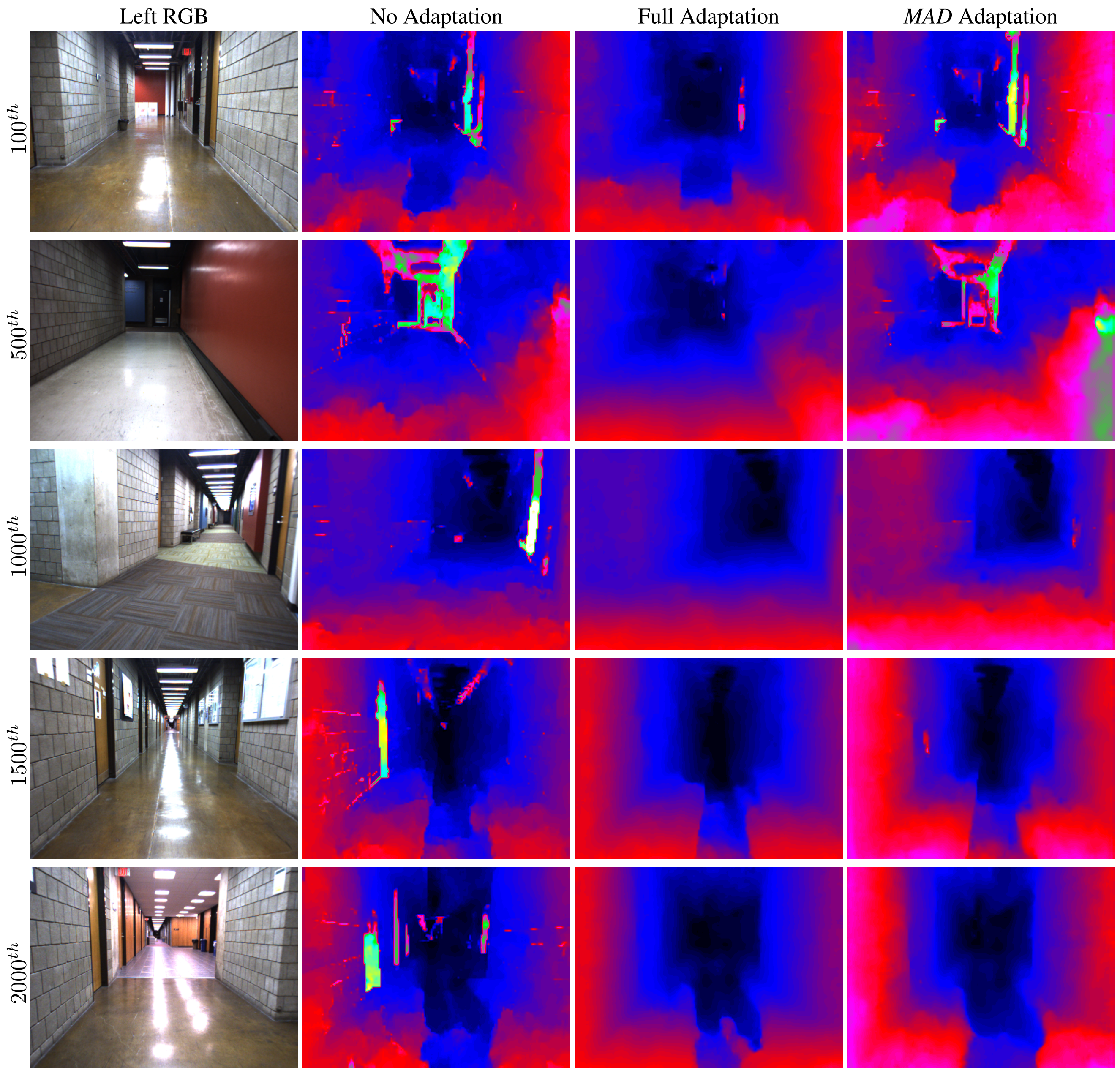

Figure 7. Comparison between predicted disparities obtained by *MADNet* with different adaptation techniques. The leftmost side of the table report the number of elaborated step in the sequence.