Active-Passive SimStereo - Supplementary material

Laurent Jospin[1*] **Allen Antony**[1] **Lian Xu**[1] **Hamid Laga**[2] **Farid Boussaid**[1]
Mohammed Bennamoun[1]
[1]University of Western Australia [2]Murdoch University
{laurent.jospin,lian.xu,farid.boussaid,mohammed.bennamoun}@uwa.edu.au
H.Laga@murdoch.edu.au

1 Organisation of this document

This document provides additional information about the Active-Passive SimStereo dataset. Section 2 confirms and motivates the choice of the license for the dataset. The author statement in case of violation of rights is presented in Section 3. Section 4 lists the file formats used in the dataset, their purpose and how to use them. Section 5 evaluate traditional non-learning methods on our dataset. Section 6 evaluate methods fine-tuned on the Active-Passive SimStereo dataset on real images. Section 7 checks whether our dataset is large enough for its intended purposes. Section 8 presents the problematic of color management. Section 9 present the simulation toolkit we used to generate the dataset. Finally, Section 10 presents the plan for the long-term storage of the dataset.

2 License

The Active-Passive SimStereo dataset is published under a CC-BY 4.0 creative commons license (`https://creativecommons.org/licenses/by/4.0/`). This license is fit for research assets as it grants to the end-user the right to use, modify, and share the data provided that proper citation to the original work is provided [7].

The Active-Passive SimStereo dataset was generated using CC0 (public domain) assets from the Blendswap [2] website and commercial assets from the Blender Market [1] website. We also used some public models from the Standford 3D repository [16] (the Stanford Bunny and Dragon, mostly as easter eggs).

3 Author statement in case of violation of rights

The authors of the dataset bear all responsibility in case of a violation of right when the dataset was created.

4 Files and file formats

The dataset contains the active and passive image pairs (provided as .jpg and .exr images), as well as the ground truth disparities for each image (provided as .npy, .pfm and .exr files).

Jpg images: The jpg image format is a very popular, lossy image format, storing a colored image with 8 bits per channels. We use this format to provide the color managed images in display refered format (see Section 8). Jpg images can be loaded by any image application or library.

Exr images: The exr image format [13] is a high dynamic range image format popular in the visual effect industry. It uses a 16 bit or 32 bit floating-point value per channel. An exr image can

36th Conference on Neural Information Processing Systems (NeurIPS 2022) Track on Datasets and Benchmarks.

contain multiple layers, each of which is made of one or multiple channels. This means that a single file can contain both the left and right view for active and passive images, as well as the disparity in floating-point format. Working with exr images can be slightly more tedious than with traditional jpg or png images. Blender [8], as well as other image processing or visual effects software such as Photoshop, Krita, Nuke, and Natron, can be used to open and inspect exr images. While some image libraries can open exr images, some do not support the layer system. When programming in C++, we recommend using the official OpenExr library [6]. When using Python, we wrote our own tool to work with exr images, which is available on GitHub under an open-source license [3].

Npy files: The npy file format is a format used by the numpy library to save and load multidimensional arrays. It is one of the most convenient file format to pass floating point image data to python. For convenience, we do provide the disparity map in the npy format.

Pfm files. The pfm file format is an old format to transfer floating points images. Writing a parser for pfm files is straightforward. Also, many scientific image processing libraries can open pfm files. We provide the disparity map in the pfm format, for convenience when using lower level programming languages like C or C++.

5 Testing traditional non-learning methods on the Active-Passive SimStereo dataset

Traditional non-learning methods are known to generalize very well on active stereo [14]. Comparing their relative performances against deep learning methods is important to check whether or not deep learning methods can outperform traditional methods.

Evaluated methods include traditional local matching [18] and semi-global matching[10]. We have used a 9×9 correlation window and a ZNCC cost function. For the semi-global matching method, we used 8 directions for the cost aggregation, with the parameters $P1$ and $P2$ set to 0.001 and 0.01.

Table 1: Evaluation results of traditional non learning methods on passive and then active stereo images.

	Passive stereo images						Active stereo images					
Method	$\text{RMSE}_T \downarrow$	$\text{MAE}_T \downarrow$	$\text{BAD}_{0.5} \downarrow$	$\text{BAD}_1 \downarrow$	$\text{BAD}_2 \downarrow$	$\text{BAD}_4 \downarrow$	$\text{RMSE}_T \downarrow$	$\text{MAE}_T \downarrow$	$\text{BAD}_{0.5} \downarrow$	$\text{BAD}_1 \downarrow$	$\text{BAD}_2 \downarrow$	$\text{BAD}_4 \downarrow$
Local matching [18]	28.10px	11.79px	49%	43%	37%	32%	24.25px	7.03px	20%	17%	16%	14%
Semis-global matching [10]	23.03px	8.95px	45%	37%	31%	26%	22.48px	6.47px	20%	17%	15%	14%

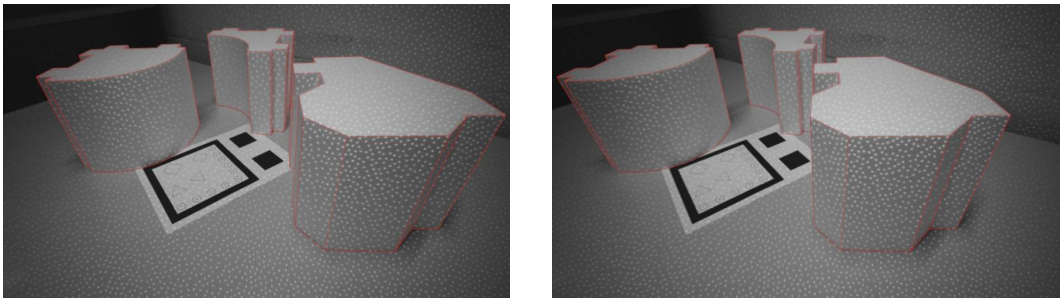
The results, reported in Table 1, show that traditional methods are unable to outperform a number of the deep learning methods that have been analyzed in the main paper (even before finetuning). For the RMSE and MAE metric, all deep learning methods outperform the traditional ones, but this can be misleading as traditional methods can have very high errors for a few single mismatched pixels. In contrast, deep learning methods tend to smooth out their solutions. For the BAD_N metrics, traditional methods are able to outperform a limited set of older deep learning methods (that have not been finetuned).

This shows that even though it is true that traditional methods perform well on active stereo, it does not mean that deep learning models are not useful for active stereo.

6 Testing the methods finetuned with the Active-Passive SimStereo dataset on real data

While our dataset is more aimed at evaluating the generalization abilities of deep learning models across passive and active stereo, it could also be used by the community to train active stereo models. One could thus evaluate whether our dataset can be used to improve the performances of deep learning models on real active stereo images.

To this end, we used the Shapes dataset [12], a dataset of active stereo images of geometric shapes cut in polystyrene. The CAD models of the shapes were then re-aligned with the images to generate high quality ground truth disparities; see Figure 1.



(a) (b)

Figure 1: Left (a) and right (b) frames of an Image pair from the Shapes dataset with the outlined CAD models aligned onto the real scene.

Finetuned methods were evaluated on the Shapes dataset, both with their original weights and with the weights obtained after finetuning. We also finetuned and evaluated ActiveStereoNet [22], which was originally trained on active stereo images, but in a self-supervised manner. Since ActiveStereoNet [22] and StereoNet [15] are related, we used the same loss as StereoNet to finetune ActiveStereoNet in a supervised manner. All other methods were trained, as described in the main paper, by using the same loss and hyperparameters as originally used by the method authors.

Table 2: Evaluation of Fine-tuned methods on the Shapes dataset.

	Shapes active stereo images					
Method	$\text{RMSE}_T \downarrow$	$\text{MAE}_T \downarrow$	$\text{BAD}_{0.5} \downarrow$	$\text{BAD}_1 \downarrow$	$\text{BAD}_2 \downarrow$	$\text{BAD}_4 \downarrow$
ACVNet [21] (original)	1.86px	0.91px	43%	20%	9%	4%
ACVNet [21] (active stereo fine-tuned)	1.58px	0.63px	34%	9%	3%	2%
Cascade-Stereo [9] (original)	2.55px	1.47px	62%	38%	18%	8%
Cascade-Stereo [9] (active stereo fine-tuned)	1.61px	0.69px	38%	13%	5%	2%
StereoNet [15] (original)	4.72px	2.07px	64%	40%	19%	8%
StereoNet [15] (active stereo fine-tuned)	4.37px	1.94px	70%	48%	20%	6%
ActiveStereoNet [22] (original self-supervised)	8.47px	2.84px	62%	39%	20%	9%
ActiveStereoNet [22] (active stereo fine-tuned)	2.04px	1.07px	58%	30%	11%	4%

The results, reported in Table 2, show clearly that finetuning on our synthetic dataset allowed the different models to improve their performances on real active stereo images, obtained with a RealSense camera. Even ActiveStereoNet [22], which was originally trained for active stereo but in a self supervised manner, see an improvement when trained in a supervised manner on our dataset. StereoNet [15] is the only exception, with the RMSE_T, MAE_T and BAD_4 metrics improving but the $\text{BAD}_{0.5}$, BAD_1 and BAD_2 metrics getting worse. This is due to the fact, as highlithed in the main paper, that the refinement module of StereoNet struggles with active stereo. This, in turn implies that the finetuning made the network focus more on large scale errors.

7 Impact of the dataset size on evaluating generalization ability

We are interested in checking if our Active-Passive SimStereo dataset is "sufficiently large". In practice, we say that a dataset is sufficiently large for the purpose of evaluating the generalization ability of deep learning stereo models if the uncertainty associated with the size of the dataset is negligible compared to the measure of interest. This can be measured by using bootstrapping [17], a method where a subset of a sample is selected at random to estimate the variability of a statistical estimator. This approach allows to get a confidence interval on our conclusions.

First, we want to check if the uncertainty due to the size of the dataset is negligible compared to the relative variations of performances between active and passive stereo. To this end, we randomly sample a subset $T_{50\%}$ of the test set T and recompute the metrics of interest. Here, we work with the MAE and BAD_2 metrics, but the code provided with this supplementary material can compute the values for any of the metrics we measured. We denote by $M_{50\%}$ the metric M computed on

Table 3: Expected MAE and BAD_2 variations between active and passive images, and within passive or active images

	MAE			BAD_2		
Method	$R(M_{\text{pas}}, M_{\text{act}})$	$\mathbb{E}[R(M, M_{50\%})]$ passive	$\mathbb{E}[R(M, M_{50\%})]$ active	$R(M_{\text{pas}}, M_{\text{act}})$	$\mathbb{E}[R(M, M_{50\%})]$ passive	$\mathbb{E}[R(M, M_{50\%})]$ active
AANet	50%	8%	5%	59%	5%	6%
ACVNet	67%	16%	12%	56%	8%	9%
AnyNet	36%	7%	6%	30%	2%	2%
CREStereo	54%	8%	9%	73%	10%	10%
CascadeStereo	58%	11%	8%	41%	6%	7%
Deep-Pruner (best)	70%	13%	10%	70%	6%	9%
Deep-Pruner (fast)	59%	10%	9%	59%	5%	8%
GANet	61%	7%	6%	67%	5%	7%
GwcNet	65%	7%	7%	65%	4%	7%
HighResStereo	56%	12%	8%	58%	6%	9%
Lac-GwcNet	68%	11%	14%	69%	5%	10%
MobileStereoNet2D	53%	6%	8%	61%	4%	7%
MobileStereoNet3D	63%	7%	7%	66%	3%	7%
PSMNet	45%	7%	3%	39%	3%	4%
RAFT-Stereo	58%	11%	8%	57%	6%	8%
RealTimeStereo	39%	6%	6%	36%	3%	3%
SMD-Nets	65%	7%	10%	63%	4%	7%
SRHNet	62%	10%	8%	68%	4%	7%
StereoNet	56%	6%	5%	54%	3%	5%
ActiveStereoNet	75%	6%	5%	61%	4%	4%

$T_{50\%}$ instead of T. Similarly, $BAD2_{50\%}$ refers to BAD_2 measured on $T_{50\%}$. We then compute the absolute relative variation between each metric M and $M_{50\%}$ defined as:

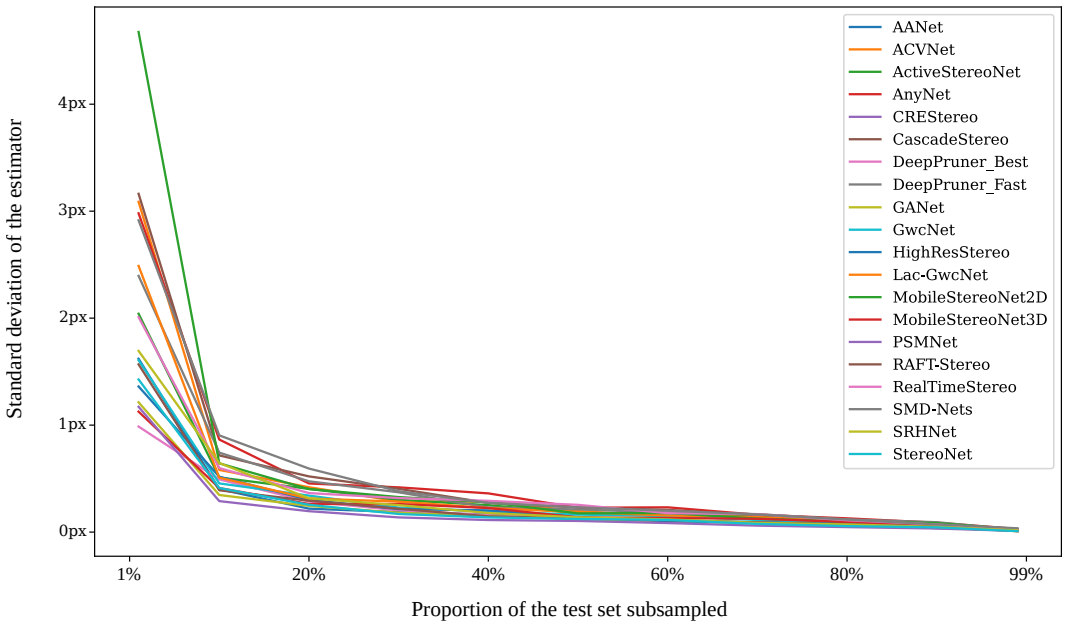
$$R(M, M_{50\%}) = \frac{|M - M_{50\%}|}{|M|}. \tag{1}$$

As $M_{50\%}$ is a random variable, we need an estimator for $R(M, M_{50\%})$. To this end, we sample N different random $T_{50\%}$ and use these to estimate the expected value $\mathbb{E}[R(M, M_{50\%})]$ of $R(M, M_{50\%})$. For our experiments, we choose N = 50. $\mathbb{E}[R(M, M_{50\%})]$ gives an estimate of the uncertainty of the value of $M_{50\%}$ for a dataset that is half the size of our test set. Using the central limit theorem, the uncertainty for the full test set of the value of M can be estimated by scaling down $\mathbb{E}[R(M, M_{50\%})]$ by a factor of $1/\sqrt{2}$. We finally compare $\mathbb{E}[R(M, M_{50\%})]$ to $R(M_{\text{pas}}, M_{\text{act}})$, the absolute relative variation between the passive and active version of M. The results are reported in Table 3.

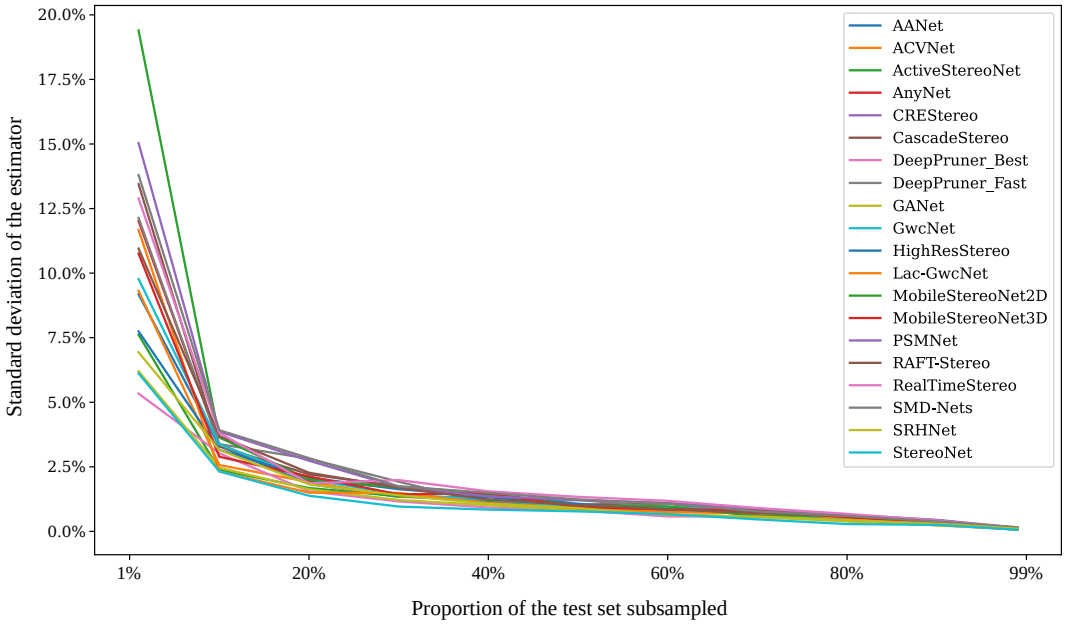
The amplitude of $R(M_{\text{pas}}, M_{\text{act}})$ is much larger than the amplitude of $\mathbb{E}[R(M, M_{50\%})]$ when M is either MAE or BAD_2. As such, our dataset can be seen as sufficiently large to enable the evaluation of the generalization ability, from passive to active stereo, of deep learning models.

To determine whether the absolute values of the performance metrics are reliable, we analysed the standard deviation $\sqrt{Var[M_{n\%}]}$ of the estimator $M_{n\%}$, where n is a variable parameter. We want to check whether or not $\sqrt{Var[M_{n\%}]}$ gets negligible compared to the metric M and if it tends towards 0 as n grows. We thus computed the value of $\sqrt{Var[M_{n\%}]}$ for M being both MAE and BAD_2 and n equal to 1%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90% and 99%. The results, reported in Figure 2, show clearly that the variance of the estimator stabilizes already towards 0 when $n = 50\%$ well below the variance of the $n = 1\%$ estimate. This shows that the dataset is large enough to get stable estimates of the performances of the different methods on the metrics used for evaluation.

Another point to consider is whether or not the results remains coherent when the test set is resampled. If the errors of all models are positively correlated, then selecting a test set that is designed to be more or less challenging should impact all methods in a similar fashion. It turns out that the metrics for each image, and consequently the differences between these metrics, are highly correlated (see e.g., Table 4, which reports the Pearson correlation coefficients of the difference between the passive and active MAE scores). A notable exception is CREStereo, which, while still clearly positively correlated with all methods, is significantly less correlated with each method than the other methods are correlated among themselves. This is probably due to the high performance of CREStereo on fine details, that other methods struggle to reconstruct. Still, Table 4 shows that resampling the test set would affect the performances of all methods in a similar fashion.



(a) MAE



(b) BAD_2

Figure 2: Standard deviation of the MAE and BAD_2 metrics as a function of the proportion of images subsampled at random in the test set (obtained with the active version of the test set).

Table 4: Pearson correlation coefficients of the differences between passive and active MAE scores on all images in T for all methods

Method	AANet	ACVNet	ActiveStereoNet	AnyNet	CREStereo	CascadeStereo	DeepPruner_Best	DeepPruner_Fast	GANet	GwcNet	HighResStereo	Lac-GwcNet	MobileStereoNet2D	MobileStereoNet3D	PSMNet	RAFT-Stereo	RealTimeStereo	SMD-Nets	SRHNet	StereoNet
AANet	100%	70%	82%	74%	42%	74%	95%	82%	86%	92%	96%	96%	96%	78%	97%	93%	66%	81%	92%	83%
ACVNet	70%	100%	74%	84%	42%	92%	78%	91%	90%	80%	72%	76%	75%	83%	73%	66%	63%	82%	73%	67%
ActiveStereoNet	82%	74%	100%	77%	50%	75%	81%	80%	82%	83%	80%	83%	87%	76%	82%	72%	77%	86%	80%	76%
AnyNet	74%	84%	77%	100%	38%	91%	77%	91%	90%	82%	74%	79%	81%	87%	77%	60%	75%	85%	76%	75%
CREStereo	42%	42%	50%	38%	100%	36%	44%	38%	39%	52%	47%	47%	44%	33%	44%	40%	58%	40%	47%	49%
CascadeStereo	74%	92%	75%	91%	36%	100%	80%	92%	91%	81%	74%	81%	80%	82%	75%	66%	64%	85%	76%	77%
DeepPruner_Best	95%	78%	81%	77%	44%	80%	100%	87%	89%	95%	97%	99%	96%	80%	94%	91%	64%	85%	92%	85%
DeepPruner_Fast	82%	91%	80%	91%	38%	92%	87%	100%	95%	87%	84%	87%	87%	94%	84%	76%	66%	90%	78%	75%
GANet	86%	90%	82%	90%	39%	91%	89%	95%	100%	91%	86%	89%	91%	89%	89%	81%	62%	88%	85%	78%
GwcNet	92%	80%	83%	82%	52%	81%	95%	87%	91%	100%	94%	96%	95%	82%	94%	84%	70%	88%	91%	84%
HighResStereo	96%	72%	80%	74%	47%	74%	97%	84%	86%	94%	100%	97%	95%	78%	96%	93%	64%	82%	90%	82%
Lac-GwcNet	96%	76%	83%	79%	47%	81%	99%	87%	89%	96%	97%	100%	97%	82%	95%	89%	67%	86%	93%	87%
MobileStereoNet2D	96%	75%	87%	81%	44%	80%	96%	87%	91%	95%	95%	97%	100%	83%	96%	89%	67%	90%	90%	85%
MobileStereoNet3D	78%	83%	76%	87%	33%	82%	80%	94%	89%	82%	78%	82%	83%	100%	81%	72%	64%	88%	74%	69%
PSMNet	97%	73%	82%	77%	44%	75%	94%	84%	89%	94%	96%	95%	96%	81%	100%	92%	66%	82%	91%	79%
RAFT-Stereo	93%	66%	72%	60%	40%	66%	91%	76%	81%	84%	93%	89%	89%	72%	92%	100%	50%	73%	82%	73%
RealTimeStereo	66%	63%	77%	75%	58%	64%	64%	66%	62%	70%	64%	67%	67%	64%	66%	50%	100%	71%	69%	62%
SMD-Nets	81%	82%	86%	85%	40%	85%	85%	90%	88%	88%	82%	86%	90%	88%	82%	73%	71%	100%	81%	80%
SRHNet	92%	73%	80%	76%	47%	76%	92%	78%	85%	91%	90%	93%	90%	74%	91%	82%	69%	81%	100%	83%
StereoNet	83%	67%	76%	75%	49%	77%	85%	75%	78%	84%	82%	87%	85%	69%	79%	73%	62%	80%	83%	100%

8 The question of color management

An important point which is often overlooked in computer vision datasets is the question of color management. Color management, in computer science, generally refers to transformations across color representation models (e.g., RGB, HSV, HSL, and XYZ). It can also refer to the collection of processes aiming at preserving the appearance of images across different devices (e.g., linear RGB vs. perceptual RGB) [19].

Modern pipelines for computer-generated imagery, including the one used for our simulations, are constructed around a scene-referred color space; see Fig. 3a. In other words, the numerical value that encodes a color is directly proportional to the light intensity in the scene. However, the image formats used for exchange and storage generally use a display-referred color space; see Fig. 3b). In other words, the numerical value encoding a color is proportional to the command that will be sent to the display device, either a screen or a projector. As the dynamic range available on such display devices is far lower than in the acquired scenes, images need to undergo a process called tone mapping, which applies a non-linear transformation to map the scene-referred color space to the display-referred color space [19].

We chose to keep a scene-referred color space for our dataset as it preserves more information. The display-referred images are also available with the published dataset. For the evaluation, we recommend choosing the color space that matches the intended application of the method. A method aimed at processing the raw signal from a camera has to be evaluated on the scene-referred images

(a) (b)

Figure 3: The same image in (a) scene referred colors (shown as if display referred) and (b) in display referred colors.

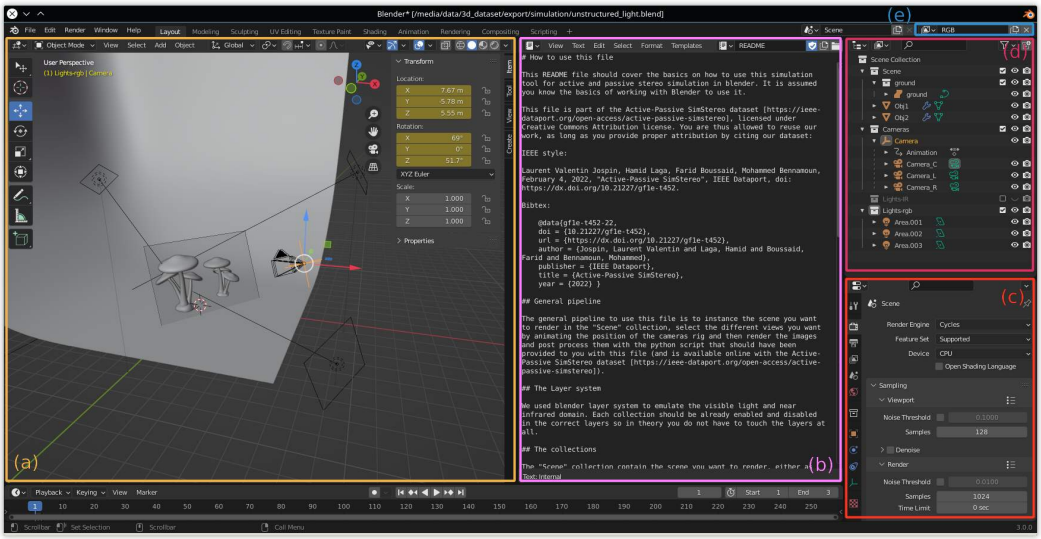


Figure 4: The interface of our simulation setup in Blender.

while methods taking usual image files as input need to be tested on display-referred images. When benchmarking existing methods on our dataset, we evaluated both color spaces and kept the best performing one for each of the methods we tested.

We rely on OpenColorIO [20] to perform the different color transformations between scene and display-refereed colors. OpenColorIO is an open source library available online [5]. The library requires configuration files encoding the different color transformations. We used the configuration files provided with the open source software Blender [8] (on version 2.93, but the files provided with the newer Blender 3.0 and 3.1 are very similar and are thus usable as well). Our post-processing code is also publicly available along with the dataset.

9 Creating your own simulated images

We provided our base simulation Blender file on IEEE dataport to let interested readers create their own simulated active-passive stereo images. This section introduces the pipeline in the file. Here, we assume a certain level of familiarity with the Blender software.

The assets need to be imported into the main 3D workspace (Fig. 4a) where the scene can be composed. Next to the 3D workspace, you can find the text editor with an internal README file opened (Fig. 4b), presenting the main pipeline.

There are a few parameters that need to be checked before rendering in the properties editor (Fig. 4c). Make sure you:

1. are rendering with the Cycles rendering engine,
2. have multi-view rendering enabled and set the rendering folder and filename template to sensible values,
3. render to Multilayer EXR images, and tick to have a single file with multiple views instead of one file per view.

Those parameters are pre-selected for you, but future versions of Blender might replace some of the default options. You might also want to tweak your rendering setting. The generated active pattern can interfere with the denoising systems in Cycles, so make sure to turn them off.

Rendering with a Ray-tracing engine can be quite slow. Table 5 reports average rendering times measured on our hardware configuration (NVIDIA GeForce RTX 2080 Ti GPU and Intel i9-10900X CPU) for a subset of frames. Rendering times may vary depending on the complexity of the scene at hand. The latest version of the Cycles rendering engine, Cycles X, support either optix or cuda

Table 5: Typical rendering time for 640×480 simulated images on an NVIDIA GeForce RTX 2080 Ti GPU and Intel i9-10900X CPU

	Cycles X (optix)	Cycles X (cuda)	Cycles X (cpu)
Passive frame	0m50s +/- 09s	2m48s +/- 0m26s	9m37s +/- 0m49s
Active frame	1m19s +/- 19s	5m25s +/- 0m51s	16m17s +/- 1m17s
Total (2x passive + 2x active)	4m18s +/- 56s	16m26s +/- 2m34s	51m48s +/- 4m12s

as back-end on the GPU. We recommend you use GPU rendering. If your GPU supports it, we recommend you use optix.

When you import your assets to create your scene, make sure to place them in the Scene collection in the outliner (Fig. 4d) as other collections might be turned on or off in certain rendering layers. If you want to edit your light for passive stereo, make sure to jump to the "Simulated NIR" view layer (Fig. 4e). If you want to go back to the visible light layer, return to the "RGB" view layer.

If you want to move or animate the camera, move the "Camera" empty object and not the three children "Camera_L", "Camera_C" and "Camera_R". This will keep the relative positions of the different cameras consistent.

Once you are happy with your composition, you can render a single image or an image sequence with a moving cameras. You then just need to post-process them with the "post_process_exr.py" Python script provided aside the simulation file.

We will also release additional data packs in the future, as we simulate more images for additional applications. At the time of writing, a first data pack with 7500 additional images from 5 scenes has been released.

10 Long-term storage

We are using IEEE dataport [4] for the long-term storage of the dataset. To alleviate the issue of the required subscription to access the data, the Active-Passive SimStereo dataset [11] has been published in open access. The only requirement to download the files is to get an IEEE account, which is free.

This should ensure that the Active-Passive SimStereo dataset and the associated tools will stay available in the foreseeable future.

References

[1] Blender market. `https://blendermarket.com/`, . Accessed: 2022-05-31.

[2] Blendswap. `https://www.blendswap.com/`, . Accessed: 2022-05-31.

[3] Exr tools. `https://github.com/french-paragon/exr-tools`. Accessed: 2022-05-31.

[4] IEEE dataport. `https://ieee-dataport.org`. Accessed: 2022-05-31.

[5] Opencolorio. `https://opencolorio.org/`. Accessed: 2022-05-31.

[6] Openexr. `https://www.openexr.com/`. Accessed: 2022-05-31.

[7] Michael W Carroll. Creative commons and the new intermediaries. *Mich. St. L. Rev.*, page 45, 2006.

[8] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. URL `http://www.blender.org`.

[9] Xiaodong Gu, Zhiwen Fan, Siyu Zhu, Zuozhuo Dai, Feitong Tan, and Ping Tan. Cascade cost volume for high-resolution multi-view stereo and stereo matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[10] H. Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 807–814 vol. 2, 2005.

[11] Laurent Valentin Jospin, Hamid Laga, Farid Boussaid, and Mohammed Bennamoun. Active-passive simstereo, 2022. URL `https://dx.doi.org/10.21227/gf1e-t452`.

[12] Laurent Valentin Jospin, Hamid Laga, Farid Boussaid, and Mohammed Bennamoun. Bayesian learning for disparity map refinement for semi-dense active stereo vision. unpublished, N.D.

[13] Florian Kainz, Rod Bogart, and Piotr Stanczyk. Technical introduction to openexr. *Industrial light and magic*, 21, 2009.

[14] Leonid Keselman, John Iselin Woodfill, Anders Grunnet-Jepsen, and Achintya Bhowmik. Intel realsense stereoscopic depth cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[15] 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 *Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany*, pages 8–14, 2018.

[16] Marc Levoy, J Gerth, B Curless, and K Pull. The stanford 3d scanning repository. *URL http://www-graphics. stanford. edu/data/3dscanrep*, 5(10), 2005.

[17] Ian Osband. Risk versus uncertainty in deep learning: Bayes, bootstrap and the dangers of dropout. In *NIPS workshop on bayesian deep learning*, volume 192, 2016.

[18] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1):7–42, 2002.

[19] Jeremy Selan. Cinematic color: From your monitor to the big screen. In *ACM SIGGRAPH 2012 Courses*. Association for Computing Machinery, 2012. ISBN 9781450316781.

[20] Doug Walker, Carol Payne, Patrick Hodoul, and Michael Dolan. Color management with opencolorio v2. In *ACM SIGGRAPH 2021 Courses*, SIGGRAPH '21, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383615.

[21] Gangwei Xu, Junda Cheng, Peng Guo, and Xin Yang. Attention concatenation volume for accurate and efficient stereo matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12981–12990, 2022.

[22] Yinda Zhang, Sameh Khamis, Christoph Rhemann, Julien Valentin, Adarsh Kowdle, Vladimir Tankovich, Michael Schoenberg, Shahram Izadi, Thomas Funkhouser, and Sean Fanello. Activestereonet: End-to-end self-supervised learning for active stereo systems. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] In section 3 we discuss the main difference of both real and simulated stereo datasets, explain why we choose to go with a simulated dataset, explain why we choose to work with physical based rendering to mitigate those issues. In the same section we also discuss why we choose to limit the resolution of our dataset and why our dataset has the size of a benchmark dataset instead of a training dataset.

(c) Did you discuss any potential negative societal impacts of your work? [No] The main applications of active stereo vision we are aware of are outlined in our introduction. While those applications can have a negative societal impact, our work proposes a benchmark to evaluate the methods aiming at that goal. Discussing the negative impacts of those applications thus feels to be out of the scope of this work.

(d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments (e.g. for benchmarks)...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We benchmarked only methods for which the code is publicly available. We do not control the repository of the models we benchmarked but in the supplementary material we gave links to each github repository, including the specific commit we used for benchmarking.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We do provide a test/train split for our dataset.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We do report estimates of the variability of predictions with our benchmark by using bootstrapping, results are discussed in section 5 of the supplementary material.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A] Our benchmark do not account for timing, so this is not relevant.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [No] Most of our assets are bought for commercial use or published under CC0, so citations are not required. We did provide a citation for the Stanford 3D models, for which it was required, in the supplementary material.
   (b) Did you mention the license of the assets? [Yes] Yes, in section 2 of the supplementary material.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Our dataset is publicly available at `https://dx.doi.org/10.21227/gf1e-t452`.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] It is assumed to be obvious from the license of the assets, which are discussed in the supplementary material: for the commercial assets this is part of the selling contract. For CC0 assets the authors choose to publish their assets under CC0.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] We do not discuss those points in the paper or supplementary material, but we made sure that our simulated scenes do not look offensive (e.g., no obscene or culturally sensitive content). For example we adhered to the guidelines given in the Stanford 3D repository for artifacts of religious or cultural significance [16].
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]