Supplementary Material

1 Normalized Cross Correlation

For a given reference image I_r, there are m source images I_s that are most relevant to its content according to feature matching. Their calibrated camera parameters P are $P = K\,[R \mid t]$. The NCC corresponding to each pixel $p = [u, v]^T$ is defined as:

$$p = Hp', H = K_s \left(R_s R_r^T - \frac{R_s \left(R_s^T t_s - R_r^T t_r \right) n^T}{d} \right) K_r^{-1}, \tag{1}$$

$$NCC\left(\mu, \mu'\right) = \frac{\mathrm{Cov}\left(I_r\left(S(\mu)\right), I_s\left(S(\mu')\right)\right)}{\sqrt{\mathrm{Var}\left(I_r\left(S(\mu)\right)\right)\mathrm{Var}\left(I_s\left(S(\mu')\right)\right)}}, \tag{2}$$

where H is the plane-induced Homography matrix, p' is the corresponding pixel for p in I_s, n and d is the normal $N(p)$ and distance $Dist(p)$ of the plane parameters in local coordinate $Hypo(p) = [N(p), Dist(p)]$, S is the pixels in the patch which take p as center, Cov and Var represent the covariance and the variance. The final NCC cost is defined as the mean of top k largest NCC:

$$E_{NCC} = \frac{1}{k}\left(\sum_k 1 - NCC_k \right). \tag{3}$$

2 PatchMatch Operation

PatchMatch is based on the assumption of local photometric consistency, which means that adjacent pixels tend to have similar depths or disparities. PatchMatch uses an iterative optimization approach. During initialization, an initial depth and normal are randomly assigned to each pixel in the image as its corresponding spatial plane parameters. Then, at each iteration step, PatchMatch will check the spatial plane parameters of adjacent pixels within the neighborhood of the current pixel. If the matching result of an adjacent pixel can provide a better match for the current pixel, then the better parameters of the adjacent pixel will be propagated to the current pixel. Generally, the Normalized Cross Correlation cost is used as the criterion to evaluate the quality of the parameters.

After the propagation step, to avoid getting stuck in local optimal solutions, PatchMatch performs a random search. For each pixel, it randomly adds some noise within the neighborhood of its current parameters and tries them out. If the new matching result is better than the current one, the parameters of the pixel are updated. PatchMatch continuously loops through the above-mentioned process until convergence.

The advantage of PatchMatch is that it has high computational efficiency and can obtain relatively accurate depth in a short period of time. It is somewhat robust to illumination changes and image noise. Its disadvantage is that it relies on the assumption of local consistency, and inaccurate matching results may occur at object boundaries and in occluded areas. In addition, the selection of initial values and the setting of the number of iterations will affect the quality of the final result.

The code of our Prior Branch is developed based on CUDA on top of [3], which ensures computational efficiency.

3 Limitation and Future Work

GSAPro significantly improves the accuracy of edge detail reconstruction through the Semantic Awareness Module. However, in a visual comparison with methods such as GaussianSurfel [2] and PGSR [1], it is evident that its sharpness is inferior to those methods that compute the photometric consistency loss between the rendered image and the auxiliary images in each iteration. In the future, a coarse-to-fine reconstruction strategy can be employed. The coarse-level reconstruction can be guided by the denoised depth, while the fine-level reconstruction can be constrained by using only color or incorporating multiview photometric consistency based on the coarse-level results.

4 Semantic Aware Module Network Architecture

We present the detailed information of all modules in the multi-resolution residual UNet of our Semantic Aware Module. Please refer to Table 1.

5 BlendedMVG Evaluation

In Table 2, we present a comparison of the reconstruction accuracy between our algorithm and the SOTA algorithms on the BlendedMVG dataset. In Table 3, we show a comparison of the rendering results.

6 DTU Evaluation

In Table 4, we present a comparison of the reconstruction accuracy between our algorithm and the SOTA algorithms on the BlendedMVS dataset. In Table 5, we show a comparison of the rendering results.

References

[1] Danpeng Chen, Hai Li, Weicai Ye, Yifan Wang, Weijian Xie, Shangjin Zhai, Nan Wang, Haomin Liu, Hujun Bao, and Guofeng Zhang. 2024. PGSR: Planar-based Gaussian Splatting for Efficient and High-Fidelity Surface Reconstruction. arXiv preprint arXiv:2406.06521 (2024).

[2] Pinxuan Dai, Jiamin Xu, Wenxiang Xie, Xinguo Liu, Huamin Wang, and Weiwei Xu. 2024. High-quality surface reconstruction using gaussian surfels. In ACM SIGGRAPH 2024 Conference Papers. 1–11.

[3] Qingshan Xu, Weihang Kong, Wenbing Tao, and Marc Pollefeys. 2022. Multi-scale geometric consistency guided and planar prior assisted multi-view stereo. IEEE Transactions on Pattern Analysis and Machine Intelligence 45, 4 (2022), 4945–4963.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009

Conference'17, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Table 1: Model layer information of the Semantic Aware Module

Module Name	Layer Type	Channels (Input, Output) & Kernel Size & Stride
init_conv.0	Conv2d	(6, 16); (5, 5); (2, 2)
init_conv.1	BatchNorm2d	(16, 16); -
init_conv.2	ReLU	(16, 16); -
unet.enc_blocks.2d2_0.0.conv1	Conv2d	(16, 32); (3, 3); (1, 1)
unet.enc_blocks.2d2_0.0.bn1	BatchNorm2d	(32, 32); -
unet.enc_blocks.2d2_0.0.relu	ReLU	(32, 32); -
unet.enc_blocks.2d2_0.0.conv2	Conv2d	(32, 32); (3, 3); (1, 1)
unet.enc_blocks.2d2_0.0.bn2	BatchNorm2d	(32, 32); -
unet.enc_blocks.2d2_0.0.downsample.0	Conv2d	(16, 32); (1, 1); (1, 1)
unet.enc_blocks.2d2_0.0.downsample.1	BatchNorm2d	(32, 32); -
unet.enc_blocks.2d2_0.1.conv1	Conv2d	(32, 32); (3, 3); (1, 1)
unet.enc_blocks.2d2_0.1.bn1	BatchNorm2d	(32, 32); -
unet.enc_blocks.2d2_0.1.relu	ReLU	(32, 32); -
unet.enc_blocks.2d2_0.1.conv2	Conv2d	(32, 32); (3, 3); (1, 1)
unet.enc_blocks.2d2_0.1.bn2	BatchNorm2d	(32, 32); -
unet.enc_blocks.2d4_1.0.conv1	Conv2d	(32, 64); (3, 3); (2, 2)
unet.enc_blocks.2d4_1.0.bn1	BatchNorm2d	(64, 64); -
unet.enc_blocks.2d4_1.0.relu	ReLU	(64, 64); -
unet.enc_blocks.2d4_1.0.conv2	Conv2d	(64, 64); (3, 3); (1, 1)
unet.enc_blocks.2d4_1.0.bn2	BatchNorm2d	(64, 64); -
unet.enc_blocks.2d4_1.0.downsample.0	Conv2d	(32, 64); (1, 1); (2, 2)
unet.enc_blocks.2d4_1.0.downsample.1	BatchNorm2d	(64, 64); -
unet.enc_blocks.2d4_1.1.conv1	Conv2d	(64, 64); (3, 3); (1, 1)
unet.enc_blocks.2d4_1.1.bn1	BatchNorm2d	(64, 64); -
unet.enc_blocks.2d4_1.1.relu	ReLU	(64, 64); -
unet.enc_blocks.2d4_1.1.conv2	Conv2d	(64, 64); (3, 3); (1, 1)
unet.enc_blocks.2d4_1.1.bn2	BatchNorm2d	(64, 64); -
unet.enc_blocks.2d8_2.0.conv1	Conv2d	(64, 128); (3, 3); (2, 2)
unet.enc_blocks.2d8_2.0.bn1	BatchNorm2d	(128, 128); -
unet.enc_blocks.2d8_2.0.relu	ReLU	(128, 128); -
unet.enc_blocks.2d8_2.0.conv2	Conv2d	(128, 128); (3, 3); (1, 1)
unet.enc_blocks.2d8_2.0.bn2	BatchNorm2d	(128, 128); -
unet.enc_blocks.2d8_2.0.downsample.0	Conv2d	(64, 128); (1, 1); (2, 2)
unet.enc_blocks.2d8_2.0.downsample.1	BatchNorm2d	(128, 128); -
unet.enc_blocks.2d8_2.1.conv1	Conv2d	(128, 128); (3, 3); (1, 1)
unet.enc_blocks.2d8_2.1.bn1	BatchNorm2d	(128, 128); -
unet.enc_blocks.2d8_2.1.relu	ReLU	(128, 128); -
unet.enc_blocks.2d8_2.1.conv2	Conv2d	(128, 128); (3, 3); (1, 1)
unet.enc_blocks.2d8_2.1.bn2	BatchNorm2d	(128, 128); -
unet.dec_blocks.2d16_3.0	ConvTranspose2d	(128, 64); (3, 3); (2, 2)
unet.dec_blocks.2d16_3.1	Conv2d	(128, 64); (3, 3); (1, 1)
unet.dec_blocks.2d16_3.2.0.conv1	Conv2d	(64, 64); (3, 3); (1, 1)
unet.dec_blocks.2d16_3.2.0.bn1	BatchNorm2d	(64, 64); -
unet.dec_blocks.2d16_3.2.0.relu	ReLU	(64, 64); -
unet.dec_blocks.2d16_3.2.0.conv2	Conv2d	(64, 64); (3, 3); (1, 1)
unet.dec_blocks.2d16_3.2.0.bn2	BatchNorm2d	(64, 64); -
unet.dec_blocks.2d8_4.0	ConvTranspose2d	(64, 32); (3, 3); (2, 2)
unet.dec_blocks.2d8_4.1	Conv2d	(64, 32); (3, 3); (1, 1)
unet.dec_blocks.2d8_4.2.0.conv1	Conv2d	(32, 32); (3, 3); (1, 1)
unet.dec_blocks.2d8_4.2.0.bn1	BatchNorm2d	(32, 32); -
unet.dec_blocks.2d8_4.2.0.relu	ReLU	(32, 32); -
unet.dec_blocks.2d8_4.2.0.conv2	Conv2d	(32, 32); (3, 3); (1, 1)
unet.dec_blocks.2d8_4.2.0.bn2	BatchNorm2d	(32, 32); -
final_conv_1	Conv2d	(128, 32); (3, 3); (1, 1)
final_conv_2	Conv2d	(64, 32); (3, 3); (1, 1)
final_conv_3	Conv2d	(32, 32); (3, 3); (1, 1)
pred_head.dec_blocks.0	ConvTranspose2d	(32, 32); (3, 3); (2, 2)
pred_head.dec_blocks.1.0.conv1	Conv2d	(32, 32); (3, 3); (1, 1)
pred_head.dec_blocks.1.0.bn1	BatchNorm2d	(32, 32); -
pred_head.dec_blocks.1.0.relu	ReLU	(32, 32); -
pred_head.dec_blocks.1.0.conv2	Conv2d	(32, 32); (3, 3); (1, 1)
pred_head.dec_blocks.1.0.bn2	BatchNorm2d	(32, 32); -
pred_head.dec_blocks.1.1.conv1	Conv2d	(32, 32); (3, 3); (1, 1)
pred_head.dec_blocks.1.1.bn1	BatchNorm2d	(32, 32); -
pred_head.dec_blocks.1.1.relu	ReLU	(32, 32); -
pred_head.dec_blocks.1.1.conv2	Conv2d	(32, 32); (3, 3); (1, 1)
pred_head.dec_blocks.1.1.bn2	BatchNorm2d	(32, 32); -
pred_head.dec_blocks.1.2.conv1	Conv2d	(32, 32); (3, 3); (1, 1)
pred_head.dec_blocks.1.2.bn1	BatchNorm2d	(32, 32); -
pred_head.dec_blocks.1.2.relu	ReLU	(32, 32); -
pred_head.dec_blocks.1.2.conv2	Conv2d	(32, 32); (3, 3); (1, 1)
pred_head.dec_blocks.1.2.bn2	BatchNorm2d	(32, 32); -
pred_head.dec_blocks.1.3.conv1	Conv2d	(32, 32); (3, 3); (1, 1)
pred_head.dec_blocks.1.3.bn1	BatchNorm2d	(32, 32); -
pred_head.dec_blocks.1.3.relu	ReLU	(32, 32); -
pred_head.dec_blocks.1.3.conv2	Conv2d	(32, 32); (3, 3); (1, 1)
pred_head.dec_blocks.1.3.bn2	BatchNorm2d	(32, 32); -
pred_head.conv1.conv	Conv2d	(32, 32); (3, 3); (1, 1)
pred_head.conv1.bn	BatchNorm2d	(32, 32); -
pred_head.conv2.conv	Conv2d	(32, 32); (3, 3); (1, 1)
pred_head.conv2.bn	BatchNorm2d	(32, 32); -
pred_head.res	Conv2d	(32, 1); (3, 3); (1, 1)

Table 2: Detailed reconstruction results of various algorithms on Blended (All images).

	GaussianSurfel		GaussianPro		2DGS		PGSR		RadeGS		Ours	
	F-score	Overall	F-score	Overall	F-score	Overall	F-score	Overall	F-score	Overall	F-score	Overall
58eaf1513353456af3a1682a	90.578	0.091	81.043	0.118	83.346	0.113	89.801	0.094	65.831	0.163	95.514	0.075
5a4a38dad38c8a075495b5d2	84.601	0.082	82.064	0.095	93.332	0.070	98.284	0.039	96.525	0.050	98.652	0.030
5aa515e613d42d091d29d300	92.430	0.066	76.046	0.116	76.497	0.115	83.262	0.101	83.478	0.100	91.618	0.076
5af02e904c8216544b4ab5a2	89.090	0.081	73.464	0.133	90.962	0.078	95.575	0.073	65.142	0.154	98.389	0.072
5b08286b2775267d5b0634ba	83.846	0.113	62.722	0.166	66.281	0.151	51.867	0.196	51.441	0.201	71.029	0.136
5b558a928bbfb62204e77ba2	78.619	0.127	63.056	0.168	54.440	0.193	67.360	0.156	53.747	0.195	77.889	0.127
5b62647143840965efc0dbde	73.964	0.142	47.756	0.203	43.565	0.218	84.144	0.111	57.421	0.189	93.417	0.085
5b69cc0cb44b61786eb959bf	85.079	0.103	73.719	0.132	78.766	0.117	82.842	0.112	64.247	0.163	92.982	0.079
5b6e716d67b396324c2d77cb	88.459	0.092	81.461	0.119	70.210	0.147	85.036	0.110	80.534	0.118	95.790	0.075
5b7a3890fc8fcf6781e2593a	88.969	0.089	69.106	0.141	62.574	0.161	72.110	0.133	58.817	0.177	73.940	0.128
5ba75d79d76ffa2c86cf2f05	83.806	0.107	64.813	0.160	65.026	0.156	67.026	0.153	65.718	0.158	96.883	0.067
5bb7a08aea1cfa39f1a947ab	85.551	0.099	71.293	0.145	78.393	0.117	62.875	0.161	66.059	0.157	97.425	0.088
5bbb6eb2ea1cfa39f1af7e0c	79.538	0.117	73.022	0.135	66.139	0.153	70.318	0.143	60.298	0.174	78.991	0.120
5bf18642c50e6f7f8bdbd492	94.121	0.074	69.914	0.149	70.190	0.141	77.563	0.124	53.948	0.186	93.761	0.081
5bf26cbbd43923194854b270	85.962	0.093	79.307	0.117	81.556	0.107	79.102	0.117	64.202	0.161	91.109	0.087
5bfc9d5aec61ca1dd69132a2	89.090	0.084	78.045	0.123	85.438	0.097	81.472	0.109	65.667	0.155	94.109	0.079
5bfe5ae0fe0ea555e6a969ca	88.046	0.091	78.652	0.119	82.853	0.104	79.739	0.114	62.676	0.167	93.071	0.082
Mean	85.985	0.097	72.087	0.138	73.504	0.132	78.140	0.120	65.632	0.157	90.269	0.088

Table 3: Detailed rendering results of various algorithms on Blended (All images).

	GaussianSurfel		GaussianPro		2DGS		PGSR		RadeGS		ours	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
5a4a38dad38c8a075495b5d2	22.951	0.719	27.188	0.833	26.643	0.804	27.197	0.828	25.767	0.835	26.262	0.806
5aa515e613d42d091d29d300	22.872	0.682	23.476	0.707	21.489	0.574	23.865	0.717	22.953	0.652	23.106	0.661
5af02e904c8216544b4ab5a2	26.278	0.751	28.910	0.831	28.091	0.795	30.702	0.882	25.322	0.683	27.869	0.788
5b08286b2775267d5b0634ba	22.115	0.753	30.167	0.902	30.271	0.881	31.801	0.927	26.566	0.776	29.239	0.880
5b6e716d67b396324c2d77cb	29.035	0.926	30.340	0.920	29.620	0.901	32.203	0.947	25.572	0.784	28.985	0.898
5b7a3890fc8fcf6781e2593a	24.699	0.733	22.633	0.679	22.664	0.599	24.062	0.746	20.950	0.506	22.257	0.627
5b69cc0cb44b61786eb959bf	26.767	0.830	29.561	0.900	29.561	0.892	30.022	0.905	25.080	0.741	29.006	0.893
5b558a928bbfb62204e77ba2	21.072	0.708	20.204	0.659	18.129	0.446	21.073	0.662	18.538	0.492	19.373	0.599
5b62647143840965efc0dbde	31.452	0.862	41.051	0.978	31.823	0.868	39.235	0.965	33.202	0.897	38.149	0.960
5ba75d79d76ffa2c86cf2f05	23.029	0.656	25.812	0.771	24.669	0.704	27.153	0.841	22.964	0.643	25.160	0.748
5bb7a08aea1cfa39f1a947ab	33.292	0.875	35.026	0.908	34.041	0.876	35.106	0.905	31.275	0.824	34.502	0.898
5bbb6eb2ea1cfa39f1af7e0c	22.027	0.655	24.016	0.765	22.830	0.695	23.966	0.761	19.699	0.515	22.838	0.711
5bf26cbbd43923194854b270	24.115	0.694	26.988	0.821	26.265	0.767	26.826	0.803	23.803	0.650	25.835	0.772
5bf18642c50e6f7f8bdbd492	27.697	0.838	27.042	0.801	26.717	0.777	28.926	0.868	23.767	0.685	26.059	0.751
5bfc9d5aec61ca1dd69132a2	23.423	0.690	26.196	0.808	25.445	0.754	26.044	0.791	22.990	0.621	25.166	0.756
5bfe5ae0fe0ea555e6a969ca	23.886	0.683	27.101	0.823	26.799	0.780	27.135	0.808	23.682	0.625	26.485	0.785
58eaf1513353456af3a1682a	26.107	0.843	29.141	0.906	28.779	0.902	31.260	0.950	24.212	0.721	27.917	0.888
Mean	25.342	0.759	27.932	0.824	26.696	0.766	28.622	0.842	24.491	0.685	26.953	0.789

Table 4: Detailed reconstruction results of various algorithms on DTU (All images).

	PGSR	GaussianSurfel	2DGS	GaussianPro	RadeGS	Ours
scan24	0.34	0.66	0.48	1.22	0.40	0.35
scan37	0.58	0.93	0.91	0.92	0.71	0.46
scan40	0.29	0.54	0.39	0.86	0.33	0.31
scan55	0.29	0.41	0.39	0.72	0.37	0.33
scan63	0.78	1.06	1.01	1.24	0.87	0.84
scan65	0.58	1.14	0.83	1.59	0.79	0.58
scan69	0.54	0.85	0.81	0.91	0.77	0.48
scan83	1.01	1.29	1.36	1.38	1.22	1.17
scan97	0.73	1.53	1.27	1.10	1.26	0.66
scan105	0.51	0.79	0.76	1.03	0.70	0.62
scan106	0.49	0.82	0.70	1.21	0.65	0.46
scan110	0.69	1.58	1.40	1.28	0.85	0.51
scan114	0.31	0.45	0.40	1.01	0.33	0.29
scan118	0.37	0.66	0.76	1.00	0.66	0.37
scan122	0.38	0.53	0.52	0.81	0.44	0.32
Mean	0.53	0.88	0.80	1.08	0.69	0.52

Table 5: Detailed rendering results of various algorithms on DTU (All images).

	GaussianSurfel		GaussianPro		2DGS		PGSR		RadeGS		Ours	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
scan24	27.208	0.920	33.685	0.947	31.883	0.919	33.052	0.965	33.084	0.933	31.512	0.915
scan37	24.470	0.891	29.483	0.941	30.315	0.933	28.845	0.959	31.175	0.941	29.567	0.927
scan40	25.055	0.902	32.115	0.935	32.998	0.925	31.823	0.957	30.228	0.864	32.560	0.921
scan55	28.535	0.846	33.829	0.903	33.473	0.896	34.429	0.946	33.298	0.923	29.721	0.882
scan63	26.062	0.930	35.089	0.958	35.898	0.950	35.197	0.967	37.028	0.952	34.131	0.947
scan65	29.026	0.843	34.216	0.903	33.862	0.895	33.286	0.905	34.936	0.909	33.539	0.900
scan69	28.720	0.862	33.224	0.913	31.382	0.899	32.515	0.938	32.388	0.911	31.368	0.897
scan83	23.636	0.840	31.725	0.902	33.636	0.886	32.955	0.925	31.907	0.890	32.422	0.883
scan97	25.434	0.862	31.157	0.907	31.106	0.887	31.531	0.926	30.260	0.892	31.059	0.887
scan105	23.090	0.843	34.558	0.915	34.297	0.896	34.553	0.935	34.378	0.901	33.730	0.893
scan106	33.491	0.906	37.332	0.936	36.408	0.925	36.941	0.952	36.417	0.933	36.178	0.925
scan110	30.445	0.875	35.328	0.925	34.035	0.908	35.896	0.948	34.742	0.920	34.600	0.916
scan114	30.401	0.888	33.734	0.923	32.742	0.910	33.177	0.945	32.492	0.919	32.490	0.912
scan118	34.037	0.900	38.166	0.931	37.021	0.918	38.223	0.949	38.164	0.929	37.065	0.920
scan122	32.341	0.893	37.999	0.930	37.141	0.918	37.933	0.946	38.249	0.929	37.272	0.921
Mean	28.130	0.880	34.109	0.925	33.746	0.911	32.654	0.916	33.916	0.916	33.148	0.910