

# High Fidelity Aggregated Planar Prior Assisted PatchMatch Multi-View Stereo

**Table 1: Quantitative results of runtime for generating a depth map. Our method consumes significantly less runtime while achieving state-of-the-art performance.**

	Time (s)	$F_1$ (test set)	$F_1$ (training set)
ACMM	20.3	80.78	78.86
ACMP	20.7	81.51	79.79
ACMMP	32.5	85.89	83.42
APD-MVS	304.7	<u>87.44</u>	<b>86.84</b>
ours	35.8	-	-
ours + SAM	44.2	<b>87.58</b>	<u>86.15</u>

## A BOUNDARY RECONSTRUCTION EFFECT

To demonstrate the performance of our method on object boundaries, we only fuse the depth of object boundaries on the ETH3D training set and compared it with other methods. table 4 shows the  $F_1$  scores for each scenario. Due to the significant performance gap between learning-based methods and PatchMatch-based methods on ETH3D, we mainly compare the performance with other PM-based methods here. Compared to the baseline, we have significantly improved our performance in scenarios such as *kicker*, *office*, *meadows*, and *playground*, while in some scenarios such as *delivery\_area* and *relief*, our performance is basically consistent with it. This is because we improve the performance of boundaries (or depth discontinuous areas) through boundary planar models, but some scenes have fewer object boundaries than others, so our boundary planar prior has less improvement in such scenes.

In fig. 1, we can see the qualitative results of the boundary, indicating that our method has significantly more accurate boundaries.

## B OBJECT RECONSTRUCTION EFFECT

We only fuse the depths in textureless objects to show how well our method performs there. We conducted experiments on ETH3D training set and the results are shown in table 5. Thanks to our object credibility detection and object plane fitting algorithm, our performance is superior to the baseline in all scenarios. Due to the varying degrees of impact of weak texture issues on different scenes, our algorithm has varying degrees of performance improvement on the scene. In scenes with large areas of weakly textured objects, such as the grasslands of *meadow* and *playground*, walls of *kicker*, etc., our algorithm significantly improves the quality of these scenes.

Weakly textured object qualitative results are also shown in fig. 2. In these cases, our method yields more complete depth.

## C RUNTIME & MEMORY

In table 1, we compare the runtime to generate a depth map and overall performance with other methods on ETH3D. APD-MVS can only reach state-of-the-art at the original resolution ( $6, 221 \times 4, 146$ ), but our method can achieve state-of-the-art at the downsampled resolution ( $3, 200 \times 2, 133$ ), which is also one of the reasons why

**Table 2: Memory consumption at different resolutions. We set the resolution of ETH3D ( $6, 221 \times 4, 146$ ) as 100%. The memory consumption of learning-based methods (marked with \*) is much greater than that of PM-based methods. We use a resolution of 50% to generate depth maps.**

Method	GPU Mem. (GB)		
	Res. (8.04%)	Res. (50.0%)	Res. (100%)
CasMVSNet*	7.8	-	-
GBi-Net*	3.6	20.7	-
PatchMatchNet*	3.5	18.6	-
IterMVS-LS*	2.5	11.2	22.0
APD-MVS	1.4	3.7	6.6
ACMMP	1.4	4.5	7.9
ours	1.5	5.2	9.4

**Table 3: Runtime at different stages. We conduct experiments on the last layer of the algorithm pyramid structure and analyze the time proportion of each stage. Our newly added module has almost no additional time consumption (mark with \*). Sparse Correspondences Generation is the process of obtaining the initial depth map through the original PM.**

Stage	Time (s)	Ratio (%)
Sparse Correspondences Generation	5.59	37.1
Initial Plane Construction	1.04	6.9
Boundary Plane Construction*	0.69	4.6
Object Plane Construction*	2.20	14.6
Planar Prior Assistance	5.54	36.8
Total	15.06	100

our method is faster. The depth maps of other methods are also downsampled, but they have a performance gap compared to ours. In addition, we introduce segmentation from SAM into pipelines and the runtime of this part can be further reduced as it develops.

As shown in table 3, we analyze the runtime of the algorithm at each stage. Compared to the baseline, we have added two additional steps, namely Boundary Plane Construction, and Object Plane Construction. The runtime of two parts can be almost negligible.

The table 2 shows the comparison of memory consumption with other methods, and it can be seen that the consumption of deep learning-based methods is significantly larger than traditional methods. The memory consumption for SAM to generate an Id Map is a constant independent of the number of views. SAM requires 4.7GB of memory on ETH3D, which is lower than the memory consumption of 5.2GB for generating depth maps. So SAM only increases time consumption without increasing memory consumption.

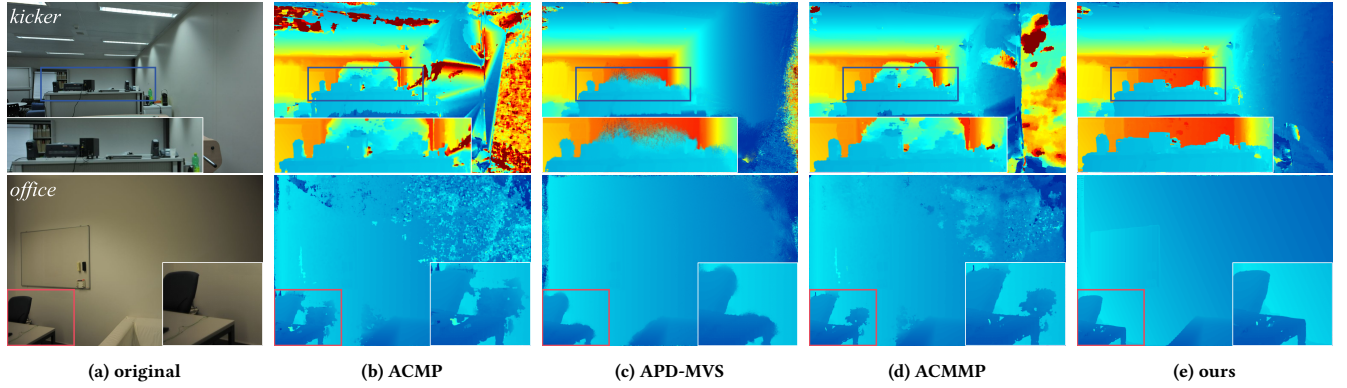
All parameter settings mentioned in our method are as follows:  $\{\alpha_{nb}, \epsilon, \eta, \gamma_0, \lambda, \tau, \pi, \rho, w_p, \beta, \gamma, \xi\} = \{20, 200, 0.0005, 0.3, 0.6, 0.75, 0.8, 0.1, 1.1, 0.18, 150, 0.9\}$ . We will also open up the complete code as soon as possible.

**Table 4:  $F_1$  score at object boundaries on ETH3D training set. The scenes marked with \* are scenes with more object boundaries (or depth discontinuities), and our method improves more in these complex scenes. Our algorithm's performance improvement is directly proportional to the number of object boundaries in the scene.**

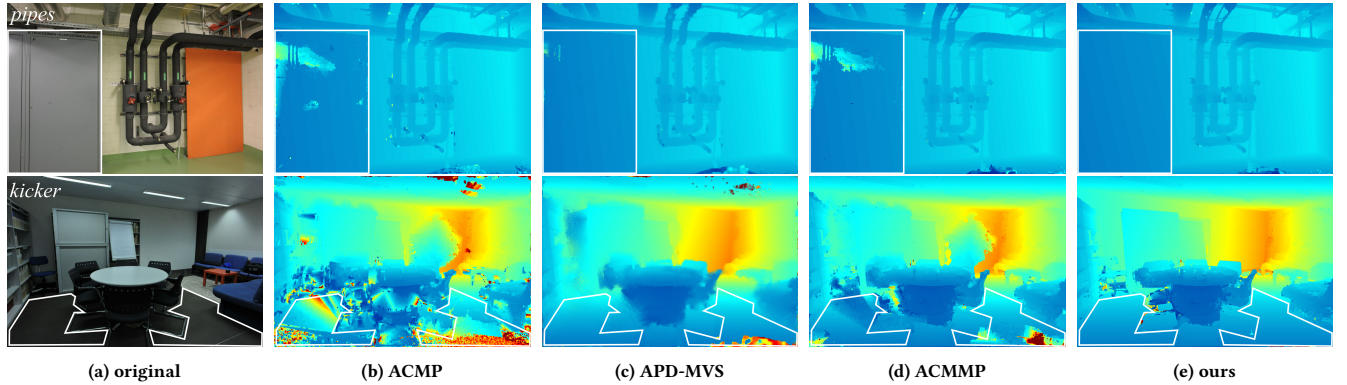
error	method	mean	indoor							outdoor					
			deli.	kick.*	offi.*	pipes*	relief	relief.	terrai.	court.	elec.	facade	mead.*	play.*	terrace
2cm	ACMP	26.87	24.27	33.99	33.87	28.06	26.57	23.71	51.08	13.24	19.35	27.04	15.90	36.35	15.89
	APD-MVS	25.80	23.62	42.15	39.79	32.00	25.17	22.74	41.10	12.06	15.24	22.17	15.18	29.48	14.75
	ACMMP	44.03	<b>48.97</b>	60.43	53.52	30.09	37.30	41.37	56.39	30.35	40.99	40.75	37.90	54.89	<b>39.43</b>
	ours	<b>44.98</b>	48.89	<b>64.26</b>	<b>55.32</b>	<b>31.30</b>	<b>37.38</b>	<b>41.42</b>	<b>57.07</b>	<b>30.47</b>	<b>41.31</b>	<b>41.03</b>	<b>39.91</b>	<b>57.13</b>	39.29
10cm	ACMP	49.51	49.67	64.45	66.53	49.99	48.22	41.73	76.77	24.49	37.86	50.04	33.21	66.78	33.86
	APD-MVS	51.23	49.74	73.58	74.32	<b>57.81</b>	48.50	42.83	74.34	24.92	35.65	50.11	35.31	65.76	33.17
	ACMMP	66.51	71.92	85.21	84.62	51.41	53.10	58.44	80.27	46.83	59.81	66.95	62.99	85.43	<b>57.65</b>
	ours	<b>67.65</b>	<b>72.13</b>	<b>88.74</b>	<b>86.27</b>	52.93	<b>53.24</b>	<b>58.55</b>	<b>80.98</b>	<b>46.94</b>	<b>60.50</b>	<b>67.36</b>	<b>66.83</b>	<b>87.50</b>	57.44

**Table 5:  $F_1$  score in the textureless object on the training set of ETH3D dataset. In the scene marked with \*, there are large areas of weakly textured objects, such as the walls of *kicker* and *office*, etc. Our method fits the object plane well for these weakly textured objects, significantly improving performance. Our performance is better than the baseline in all scenarios.**

error	method	mean	indoor							outdoor					
			deli.	kick.*	offi.*	pipes*	relief	relief.	terrai.	court.	elec.*	facade*	mead.*	play.*	terrace
2cm	ACMMP	68.01	73.62	69.02	62.91	61.73	73.47	69.14	55.34	82.96	82.71	64.41	62.41	61.57	64.87
	ours	<b>70.78</b>	<b>74.07</b>	<b>75.20</b>	<b>66.99</b>	<b>64.40</b>	<b>74.34</b>	<b>69.66</b>	<b>55.89</b>	<b>84.04</b>	<b>84.11</b>	<b>67.32</b>	<b>71.92</b>	<b>65.50</b>	<b>66.75</b>
10cm	ACMMP	81.04	81.31	82.66	81.05	70.21	85.00	82.24	64.49	91.79	89.52	88.20	78.67	86.23	72.11
	ours	<b>82.31</b>	<b>81.52</b>	<b>86.28</b>	<b>82.31</b>	<b>71.89</b>	<b>85.20</b>	<b>82.58</b>	<b>64.57</b>	<b>91.88</b>	<b>89.84</b>	<b>88.81</b>	<b>84.72</b>	<b>88.07</b>	<b>72.36</b>



**Figure 1: Boundary qualitative results. In *kicker*, it is difficult for other methods to accurately distinguish the depth information of the foreground and background at the object boundary, while our object boundary is very sharp. In *office*, we have a clear boundary at the edge of the chair, and we have successfully reconstructed the whiteboard on the wall.**



**Figure 2: Qualitative results of weakly textured objects. In *pipes*, our depth on the cabinet is very smooth, while other methods still have many depth errors on such weakly textured objects. In *kicker*, we have a more complete ground depth.**