Addressing Low-Shot MVS by Detecting and Completing Planar Surfaces

Supplementary Material

1. Depth and Normal Map Updates

Our approach aims to improve upon the depth and normal maps generated by ACMP [5] using plane extent inferences as shown in Algorithm 1. The first step in the algorithm is to create a depth map for only the detected planes in a given image, which we'll refer to as a **planes depth map**. For each plane extent inference, we determine the depths by back-projecting the inferred pixels into the image coordinate space using ray-plane intersections. If a pixel is inferred as part of multiple planes, we assign it the depth corresponding to the plane for which it has the highest confidence. As a result, this process yields a depth map where each pixel has a depth corresponding to one of the detected planes (or no depth at all). Similarly, we also generate a **planes normal map** where each pixel contains the normal corresponding to the plane for which it has the highest confidence (if any). Furthermore, we also store the highest confidence value for each pixel which yields a **planes confidence map**.

Next, the planes depth map and the base depth map (generated by ACMP) are merged using the planes confidence and counter maps. Our output is a merged depth map and a depth source map that indicates the source of each depth in the merged depth map (base depth map or planes depth map). To determine the depth at each pixel, we need to ascertain which depth map source (the base depth map or the planes depth map) is more likely to be accurate. We use the planes confidence map and the counter map for this task because the planes confidence map represents the confidence in depth predictions in the plane depth map, and the counter map acts as a proxy for confidences in depth prediction of the base depth map; the larger the count, the more confident we are in the estimated depths of the base MVS method. To facilitate comparison between the planes confidence map and the counter map, we convert the counter map to a base confidence map based on a straightforward mapping from counts to confidences. The confidence maps (planes confidence map and base confidence map) are then compared and a higher value (at a pixel) determines the source of the depth, thus yielding a merged depth map and a depth source map.

Algorithm 1: Map Update

Inputs : Base map (depth or normal) $\boldsymbol{M}$, Counter map $\boldsymbol{N}$, Camera Intrinsics $\boldsymbol{K}$, Plane extent inferences I_1 ... I_n, Plane parameters P_1 ... P_n

$W, H \leftarrow Width_M, Height_M$

$x_{\mathrm{W}\times\mathrm{H}}, y_{\mathrm{W}\times\mathrm{H}}$ = Grid(W, H) ▷ 2D grid of every pixel

▷ Initialize **planes depth** and **planes confidence** maps

$D_P, C_P \leftarrow Zeros(W, H)$

▷ Determine world-space vectors via back-projection

$X_{\mathrm{W}\times\mathrm{H}}, Y_{\mathrm{W}\times\mathrm{H}}$ = Back-project($x_{\mathrm{W}\times\mathrm{H}}, y_{\mathrm{W}\times\mathrm{H}}, \boldsymbol{K}$)

foreach *detected plane* ***pp*** **do**

  ▷ Calculate plane depths using plane-vector intersections

  $Z_{\mathrm{W}\times\mathrm{H}} = P_{pp} \cap (X_{\mathrm{W}\times\mathrm{H}}, Y_{\mathrm{W}\times\mathrm{H}})$

  ▷ Pixels where inference confidence is greater than existing planes confidence map

  $x_r, y_r \leftarrow \arg_{x,y}(I_{pp} > C_P)$

  $D_P(x_r, y_r) = Z(x_r, y_r)$ ▷ Update planes depth map

  $C_P(x_r, y_r) = I_{pp}(x_r, y_r)$ ▷ Update planes conf. map

end

▷ Initialize **merged depth** and **depth source** maps

$D_M, D_S \leftarrow Zeros(W, H)$

▷ Convert counter map to **base confidence** map (example shown)

$C_O \leftarrow Mapping(N)$

$$C_O = \begin{cases} 0.50 \textbf{ if } count == 0 \\ 0.90 \textbf{ if } count == 1 \\ 0.99 \textbf{ if } count == 2 \\ 1.00 \textbf{ otherwise} \end{cases}$$

▷ Compare base and planes confidence maps and record source

$D_S \leftarrow \max_{Base\leftarrow 0; Planes\leftarrow 1}(C_O, C_P)$

▷ D_M contains depths from source corresponding to the higher confidence value (at a pixel level)

$D_M \leftarrow M$ `WHERE` $D_S == 0$

$D_M \leftarrow D_P$ `WHERE` $D_S == 1$

Outputs: Merged depth map D_M

Depth source map D_S

2. Image Sampling for Low-Shot MVS

To simulate the low-shot scenario for the ETH3D dataset [4], we generate smaller subsets of scenes using tracks generated as part of structure from motion (SfM). A "track" is generated by connecting a set of feature matches across image pairs and the track length indicates the number of images that observe the track. The tracks graph is a bipartite graph where the two sets are represent images and tracks. An edge between the sets indicates observation, i.e. an image observing a track. To generate smaller subsets, we recursively remove the most redundant image as indicated by the tracks graph until the desired low-shot scenario is achieved. An image is considered the most redundant if it observes the least number of short tracks, where the shortest track is observed by two images. Removal of such images has the least impact on tracks, as tracks with at least two observations are still generated, which indicates at least two images view the location of the track.

For the Tanks and Temples [2] test set, we generate a smaller subset by sampling the video at interval of 1 image every 10 seconds.

3. Segmentation Network Training Details

We use the DeepLabv3 architecture with a ResNet50 backbone pre-trained on the PASCAL VOC [1] classes from the MS COCO [3] dataset. We use the stochastic gradient descent w/ momentum optimizer with an initial learning rate

	30%				60%				90%				100%			
	I	P	T(A)	T(O)	I	P	T(A)	T(O)	I	P	T(A)	T(O)	I	P	T(A)	T(O)
ETH3D training set	11	23	90	42	21	53	234	85	32	85	409	153	35	94	463	160

Table 1. Average run time and overhead added by our approach for 13 scenes in the ETH3D training set. The results are grouped using different low-shot scenarios (30%, 60%, 90%, and 100% indicate the percentage of images we used from the original scene). The heading *"I"* indicates the average number of images per scene; *"P"* indicates the average number of detected planes per scene; *"T(A)"* indicates the average run time (in seconds) per scene using ACMP; and *"T(O)"* indicates the average overhead (in seconds) added by our approach.

of 0.001, a momentum value of 0.9, and a weight decay of 0.0001. We use a step-based decay scheduler where we systematically drop the learning rate to 95% of its previous value after every epoch of training.

4. Runtime Performance

To address low-shot MVS, our system detects planes, generates features for each detected plane, and infers its extent in each image. Table 1 shows the average runtime of ACMP and the overhead added by our approach for the scenes in the ETH3D[4] training set. Our experiments were run on an Intel Xeon E5-1650 V4 3.6GHz 6-Core machine with 64GB DDR4-2133 RAM and an NVIDIA Titan X 12GB GPU.

References

[1] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, 2015. 1

[2] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4), 2017. 1

[3] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2014. cite arxiv:1405.0312Comment: 1) updated annotation pipeline description and figures; 2) added new section describing datasets splits; 3) updated author list. 1

[4] Thomas Schöps, Johannes L. Schönberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, Marc Pollefeys, and Andreas Geiger. A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2

[5] Qingshan Xu and Wenbing Tao. Planar prior assisted patchmatch multi-view stereo. *CoRR*, abs/1912.11744, 2019. 1