The Chinese University of Hong Kong, Shenzhen

DDA2082: Independent Study II Report

Name: Xiang Fei

Student ID: 120090414

Email: xiangfei@link.cuhk.edu.cn

Mentor: Qilin Sun

Date: 2024.5

Table of Contents

1 Preface

The report shows my learning and exploration in the field of PatchMatch MVS during the course DDA2082 Independent Study II. Since the course emphasizes more than just scientific research, this report is not strictly organized as an academic paper. Before presenting the proposed method, I first introduce my understanding of the basic ideas of standard PatchMatch Stereo. If you are not familiar with PatchMatch Stereo, the first part of the report will help a lot. Then, the proposed MVS method with Broad Adaptive Checkerboard Sampling and Dynamic Multi-Hypothesis Joint View Selection is presented in the second part. Finally, the third part shows the results of the proposed method in ETH3D dataset.

I am very grateful to Professor Sun Qilin for his help in my study and research process, especially for giving me very useful suggestions on how to introduce my research.

2 Introduction to PatchMatch Stereo

2.1 Overview

PatchMatch Stereo (PMS) [Bleyer et al. 2011] is a binocular stereo matching algorithm article published at the British Machine Vision Conference (BMVC) in 2011. The method is very classic and the idea of slanted support windows breaks the shackles of traditional fixed window local matching thinking. What's even more valuable is that, like SGM, it has excellent data generalization capabilities and can achieve good results for most data.

Although PatchMatch Stereo here is a binocular stereo algorithm instead of multi-view stereo (MVS) algorithm, it is the basis of PatchMatch-based MVS and reveals the core idea of using PatchMatch to estimate disparity and depth, and then obtain the 3D reconstruction. Therefore, it is necessary to first introduce PatchMatch Stereo in this report.

2.2 Slanted Support Windows

Fronto-Parallel Windows. Before introducing Slanted support windows, it is very necessary to introduce another fixed window model: Fronto-parallel windows.

Fronto-parallel windows is a very classic window model, which refers to windows directly in front of the camera that are parallel to the image plane after epipolar correction, and are also perpendicular to the Z-axis of the camera coordinate system of each camera after correction. The characteristics of this window are as follows:

- The projection lengths of any line segments in the window on the left and right images (epipolar line image pairs) are equal.
- All spatial points in the window have the same depth. From D=bf/d, it can be seen

that the disparity of the projection point of the spatial point on the image is also the same.

These two characteristics are very friendly to rectangular window matching, so that all pixels in the window where the left and right images are centered on a point pair with the same name can correspond one-to-one with the same name, and all pixels in the window have the same unique disparity, so if If the texture conditions are good, there is even no need for cost aggregation, and good results can be obtained through simple local similarity algorithms such as the correlation coefficient method.

But the problem is that such a window is too ideal, and it is difficult to find such a scene in practical applications. More often, there may be several fronto-parallel windows in the scene, or there may be none. But fortunately, this window provides us with good research ideas. Many algorithms are further optimized based on this window model, such as SGM [Hirschmuller 2005] and AD-Census [Mei et al. 2011]. They calculate the initial cost value based on Fronto-parallel windows, and then obtain the cost Aggregation, optimizing the cost, and getting very good results.

At the same time, some researchers have begun to find other ways to find more reasonable window models. In 2011, the PatchMatch Stereo algorithm based on Slanted support windows came into being.

Slanted Support Windows.

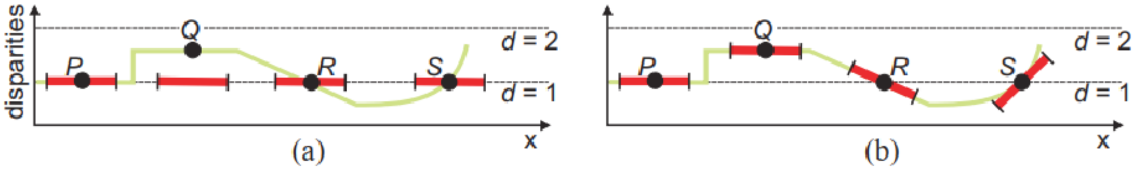


Figure 1: The illustration of Fronto-parallel windows and slanted support windows in [Bleyer et al. 2011], which shows the support regions (in 1D). The points of green surface shall be reconstructed. Support regions are show by red bars. (a) Fronto-parallel windows at integer disparities as used in standard methods. (b) Slanted support windows. The 3D plane is estimated at each point.

According to Fig.1 (a), we can see that P conforms to the assumption of Fronto-parallel windows, in which the local surface is Fronto-parallel and has the same disparity value. However, R and S do not conform. The surfaces are both inclined. R is an inclined plane and S is Inclined surface. According to Fig.1 (b), We can observe the changes in the R and S windows. The windows under the assumption of Fronto-parallel windows are parallel to the disparity dimension and do not fit the surface trend, while slanted support windows fit the surface very well. Slanted support windows should dynamically change with the orientation of the surface.

The core idea of PatchMatch Stereo is to find a dynamic disparity plane for all pixels. Consider pixel p with disparity d_p, the disparity plane of p is:

$$d_p = a_{f_p} p_x + b_{f_p} p_y + c_{f_p} \tag{1}$$

where a_{f_p}, b_{f_p}, c_{f_p} are parameters of the disparity plane. Therefore, the disparity estimation problem is converted into a plane estimation problem. Stereo matching is to find the parameters of the optimal plane for each pixel, that is, to find the plane with the smallest aggregation cost for each pixel:

$$f_p = \arg\min_{f \in F} m(p, f) \tag{2}$$

where F is a set of planes, $m(p, f)$ is the aggregation cost of p with disparity plane f:

$$m(p, f) = \sum_{q \in W_p} w(p, q) \cdot \rho(q, q - (a_{f_p} p_x + b_{f_p} p_y + c_{f_p})) \tag{3}$$

where W_p is a square window centered at p, $w(p, q)$ is an adaptive weight to solve edge-fattening problem. In PMS, $w(p, q)$ is obtained by computing the possibility that p and q has the same plane, determining with the differences of colors:

$$w(p, q) = e^{-\frac{\|I_p - I_q\|}{\gamma}} \tag{4}$$

where γ is a hyper-parameter, $\|I_p - I_q\|$ is the L1-distance in RGB space between p and q.

In the formula of $m(p, f)$, there is an important function ρ, which is to compute the dissimilarity of two pixels. For left image pixel q, its disparity is $d_q = a_f q_x + b_f q_y + c_f$, the corresponding pixel in the right image $q' = q - d_q$, the dissimilarity is:

$$\rho(q, q') = (1 - \alpha) \cdot min(\|I_q - I_{q'}\|, \tau_{col}) + \alpha \cdot min(\|\Delta I_q - \Delta I_{q'}\|, \tau_{grad}) \tag{5}$$

where τ_{col} and τ_{grad} are truncate parameters to achieve more robust computation in occlusion areas.

However, how to solve the optimization problem? Searching in the unbounded set F is not reasonable. Therefore, the next part will introduce the method to solve this problem.

2.3 Disparity Estimation Based on PatchMatch

The basic idea of PatchMatch is: in images, the disparity planes of all pixels in a pixel block of a certain size can be approximated as the same. This also constitutes the basic idea of PMS, that is, the image can be regarded as multiple pixel blocks, and each pixel block has an approximate disparity plane. The goal of the algorithm is to find all the disparity planes of the image.

The procedures are as follows:

1. **Random Initialization.** This is the first step for PMS to find a disparity plane, that is, to initialize a random disparity plane for each pixel. PMS hopes that through this step, at least 1 pixel can be randomly assigned to the correct plane.

For the initialization, PMS does not randomly assign values to the three plane parameters a_f, b_f, c_f. The disadvantage of this way that it cannot give a range constraint to the plane. Therefore, the method in PMS is that: Give each pixel a random disparity value z_0 within the disparity range, and then randomly assign an unit vector $\vec{n} = (n_x, n_y, n_z)$ as the normal. Then, a_f, b_f, c_f can be computed:

$$a_f = -\frac{n_x}{n_z} \tag{6}$$

$$b_f = -\frac{n_y}{n_z} \tag{7}$$

$$c_f = \frac{n_x x_0 + n_y y_0 + n_z z_0}{n_z} \tag{8}$$

2. **Disparity Propagation.** The basic idea is to propagate a small number of correct disparity planes among all random disparity planes to other pixels. Here I only introduced Spatial Propagation which is also used in PatchMatch MVS.

   Spatial Propagation. The idea behind spatial propagation is that spatially adjacent pixels are most likely to have similar disparity planes. Therefore, consider pixel p with plane f_p, check whether the disparity plane f_q of pixel q in its neighborhood is more suitable for p. That is, to check $m(p, f_q) < m(p, f_p)$. If so, then treat f_q as the new disparity plane of p. In even-numbered iterations, q is the left and upper pixels of p; in odd-numbered iterations, q is the right and lower pixels of p.

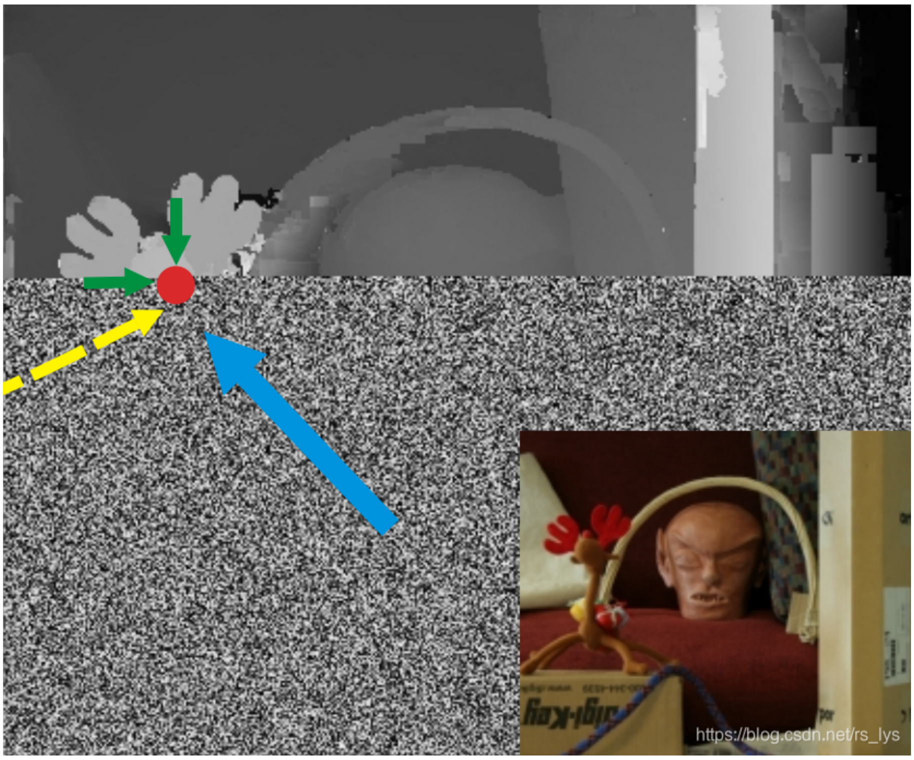


Figure 2: The upper part is the results of disparity propagation, the lower part is the results of random initialization.

3. **Plane Refinement.** The objective of plane refinement is to optimize the parameters of f_p and further reduce the aggregation cost $m(p, f_p)$. PMS sets two parameters $\Delta_{z_0}^{max}$

and Δ_n^{max}. Then, randomly choose a value Δ_{z_0} in $[-\Delta_{z_0}^{max}, \Delta_{z_0}^{max}]$ and set $z_0' = z_0 + \Delta_{z_0}$, and choose $\vec{\Delta_n}$ in $[-\Delta_n^{max}, \Delta_n^{max}]$, set $\vec{n'} = u(\vec{n} + \vec{\Delta_n})$, u means computing the unit vector. Therefore, we get a new plane $f_{p'}$, if $m(p, f_{p'}) < m(p, f_p)$, then treat $f_{p'}$ as the new plane of p.

The plane refinement is also perform iteratively. After every iteration, the parameters $\Delta_{z_0}^{max}$ and Δ_n^{max} will decrease.

Therefore, we can solve the plane optimization problem without the need of brute force enumeration. In fact, through the propagation method, in many cases we only need 3 iterations to obtain a good disparity map.

3 MVS with Broad Adaptive Checkerboard Sampling and Dynamic Multi-Hypothesis Joint View Selection

After understanding the basic idea of PatchMatch, I started to learn some state-of-the-art PatchMatch MVS algorithms, and aimed to make improvements based on these algorithms. After exploration, I had ideas for improving an PatchMatch MVS method called ACMH [Xu and Tao 2019], which is published at CVPR 2019. My improvements are focused on two parts: the pixel hypothesis sampling before disparity propagation, and the view selection. These parts are named as **Broad Checkerboard Sampling** and **Dynamic Multi-Hypothesis Joint View Selection**, which is introduced in the following sections. Other parts of the entire MVS algorithm such as random initialization, refinement, multi-scale geometric consistency, etc. are completely consistent with ACMH. For completeness, I will also introduce them in the following sections.

3.1 Overview

The task of the MVS algorithm can be described as: Given a set of input images $I = \{I_i | i = 1 \cdots N\}$ with known calibrated camera parameters $P = \{P_i | i = 1 \cdots N\}$, our goal is to estimate depth maps $D = \{D_i | i = 1 \cdots N\}$ for all images and fuse them into a 3D point cloud. More specifically, we just need to estimate the depth map of reference image I_{ref} sequentially selected from I with the guidance of source images $I_{src}(I - I_{ref})$.

3.2 Structured Region Information

Structured region information means that pixels within a relatively large region can be approximately be modeled by the same 3D plane.

3.2.1 Random Initialization

The initialization for ACMH is very similar to the previously introduced method. First, randomly generate a hypothesis including depth (or disparity) and normal to build a plane for each pixel in the reference image I_{ref}. For each hypothesis, a matching cost is computed from each of $N-1$ source images. And then, the top K best matching costs within the $N-1$ costs are aggregated to form the initial aggregation cost of the pixel.

3.2.2 Broad Checkerboard Sampling

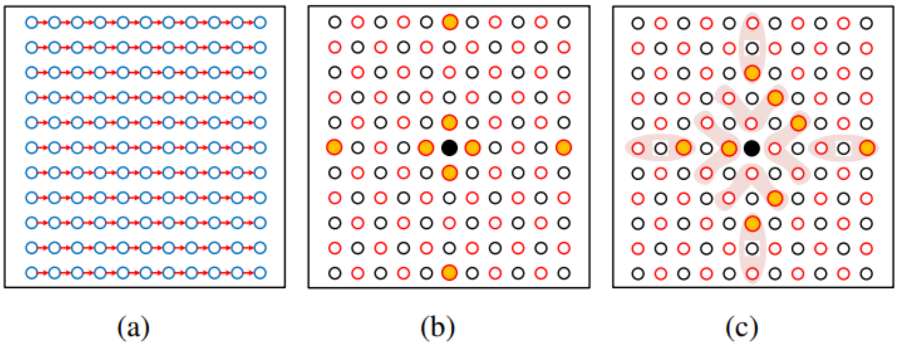


Figure 3: Propagation scheme showed in [Xu and Tao 2019]. (a) Sequential propagation. (b) Symmetric checkerboard propagation. (c) Adaptive checkerboard propagation. The light red areas in (c) show sampling regions. The solid yellow circles in (b) and (c) show the sampled points.

During the introduction of PatchMatch Stereo, we've talked about a propagation scheme called sequential propagation, which is also used in a well-known MVS method, COLMAP [Schönberger et al. 2016]. However, this method is inefficient and time-consuming. Therefore, [Galliani et al. 2015] proposed to partition the pixels of I_{ref} into red-black grids of a checkerboard. This pattern allows us to simultaneously update the hypotheses of black pixels using red pixels and vice versa. Besides, this method can make a full use of GPU, which can largely speed up the algorithm. However, in [Galliani et al. 2015], the pixel hypotheses used to update the considered pixel are eight fixed points, which limits the propagation of good planes. Therefore, ACMH proposed to sample eight points from four V-shaped areas and four long strip areas based on the aggregation matching costs. However, this method is also very handcrafted in the construction of the eight areas and also missed some pixels in the neighborhood of the considered pixel, which still limits the propagation of good planes.

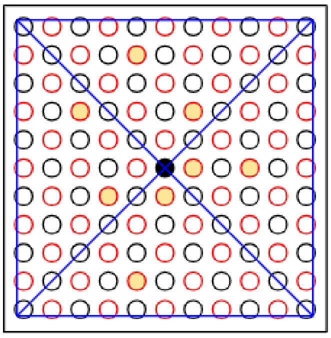


Figure 4: Propagation scheme of Broad Checkerboard Sampling. Blue line shows the window and the four areas. The solid yellow circles show the sampled points.

Therefore, I propose a new hypothesis sampling method called Broad Checkerboard Sampling. Specifically, for each considered black/red pixel i, I construct a squared window which is center at i, and divided the window into four areas. During the sampling, all the red/black pixels in the window are considers. Finally, in each area, two best hypotheses are selected for later propagation. This method enlarges the number of considered pixels during the sampling and helps a good plane of a local shared region to spread further. Besides, this method helps deal with low-texture areas. This is because during random initialization, it is very likely to obtain a good hypothesis in the low-texture areas. However, when using sequential propagation and the adaptive checkerboard propagation, the sampling and propagation is not thorough, and it is very difficult for the good hypothesis to propagate to other pixels. When using the broad checkerboard sampling and propagation method, the good hypothesis is more likely to be propagate to other pixels. In addition, when applying the proposed method, we can use less iterations to get better results.

3.2.3 Dynamic Multi-Hypothesis Joint View Selection

To obtain a robust multi-view matching cost for each pixel, ACMH further leverages these the obtained eight structured hypotheses to infer the weight of every neighboring views. Firstly, build a matching costs matrix for each pixel p:

$$\mathbf{M} = \begin{bmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,N-1} \\ m_{2,1} & m_{2,2} & \cdots & m_{2,N-1} \\ \vdots & \vdots & \ddots & \vdots \\ m_{8,1} & m_{8,2} & \cdots & m_{8,N-1} \end{bmatrix} \tag{9}$$

where $m_{i,j}$ is the matching cost for the i-th hypothesis h_i scored by the j-th view I_j.

After that, ACMH uses a voting scheme to infer the weights of the source views with the matching cost matrix. More specifically, ACMH sets a good matching cost bound and a bad matching cost bound of matching costs, and for each source view, consider the number of matching costs (totally 8 costs) that are good views and the number of bad views. Then, use this information to refer the weight of each view. However, the method in ACMH uses fixed bad matching bound. In fact, through propagation iterations, the matching costs are updated to be smaller, so the bad matching bound should also be updated to be smaller. Therefore, I proposed a view selection method called Dynamic Multi-Hypothesis Joint View Selection based on the method used in ACMH.

Dynamic Multi-Hypothesis Joint View Selection.

The good matching cost bound is defined as:

$$\tau_g(t) = \tau_{g0} \cdot e^{-\frac{t^2}{\alpha_g}} \tag{10}$$

where t means the t-th iteration, τ_{g0} is the initial good matching cost threshold and α_g is a constant.

The bad matching cost bound is defined as:

$$\tau_b(t) = \tau_{b0} \cdot e^{-\frac{t^2}{\alpha_b}} \tag{11}$$

where t means the t-th iteration, τ_{b0} is the initial bad matching cost threshold and α_b is a constant.

For a specific view I_j, there should exist n_1 matching costs satisfying: $m_{i,j} < \tau_g(t)$. Then, define good matching cost set $S_{good}(j)$. Also, there should be less than n_2 matching costs satisfying: $m_{i,j} > \tau_b(t)$. A view simultaneously satisfying the above conditions will be incorporated into the current view selection set S_t in the t-th iteration. S_t may contain some unstable views because of noise, viewing points, etc. To evaluate the importance of each selected view, the confidence of a matching cost is computed as follows:

$$C(m_{ij}) = e^{-\frac{m_{ij}^2}{2\beta^2}} \tag{12}$$

where β is a constant. This makes good views more discriminative. The weight of each selected view can be defined as:

$$w(I_j) = \frac{1}{|S_{good}(j)|} \sum_{m_{i,j} \in S_{good}(j)} C(m_{i,j}), I_j \in S_t \tag{13}$$

Suppose the most important view v_{t-1} in iteration $t-1$ shall continue to have influence on the view selection of current iteration t. Thus, we obtain:

$$w'\left(I_j\right) = \begin{cases} \left(\mathbb{I}\left(I_j = v_{t-1}\right) + 1\right) \cdot w\left(I_j\right), & \text{if } I_j \in \mathrm{S}_t \\ 0.2 \cdot \mathbb{I}\left(I_j = v_{t-1}\right), & \text{else.} \end{cases} \tag{14}$$

This modification can make the view selection more robust.

$$m_{photo}(p, h_i) = \frac{\sum_{j=1}^{N-1} w'(I_j) \cdot m_{i,j}}{\sum_{j=1}^{N-1} w'(I_j)} \tag{15}$$

The current best estimate for pixel p is updated by the hypothesis with the minimum multi-view aggregated cost.

3.2.4 Refinement

After each red-black iteration, a refinement step is applied to enrich the diversity of solution space, which is similar to the previous introduced method. That is, make some changes to depth and normal to form new hypotheses, and compare them with the matching costs of the original hypothesis. At the end, a median filter of size 5×5 is applied to our final depth maps. In addition, we can also add the Multi-Scale Geometric Consistency module in [Xu and Tao 2019] to get better results in low-texture areas.

3.3 Fusion

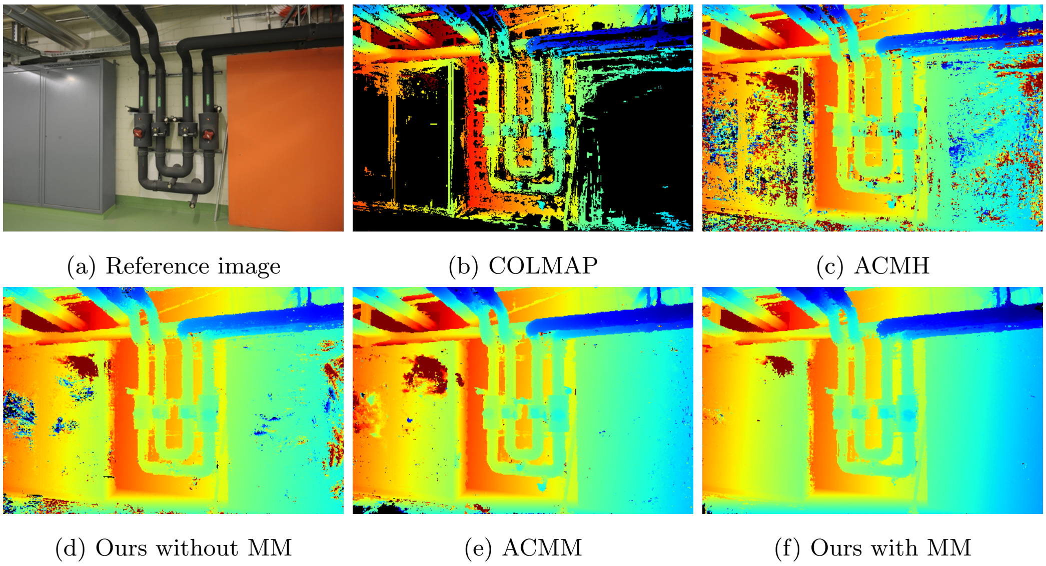
After obtaining all depth maps, we sequentially view each image as a reference image, convert its depth map into 3D points in world coordinates, and project them to its adjacent views to obtain corresponding matches. We define a consistent match that satisfies the relative depth difference $\epsilon \leq 0.01$, the angle between normals $\theta \leq 30^o$ and the reprojection error $\phi \leq 2$ as in [Schönberger et al. 2016]. If there exist more than 2 satisfied neighboring views, the depth estimate will be accept. Finally, the 3D points corresponding to these consistent depth estimates and the normal estimates are averaged into a unified 3D point.

4 Results

I tested the proposed method in a MVS dataset, ETH3D benchmark [Schops et al. 2017]. In the multi-hypothesis joint view selection scheme:

$\{\tau_{g0}, \tau_{b0}, \alpha_g, \alpha_b, \beta, n_1, n_2\} = \{0.8, 1.2, 90, 120, 0.3, 2, 3\}$

From both the depth maps and the reconstructed point clouds, we can see that the proposed method presents very obvious improvements in low texture areas and produces the best reconstruction results. At the same time, our approach does not sacrifice in details.



(a) Reference image

(b) COLMAP

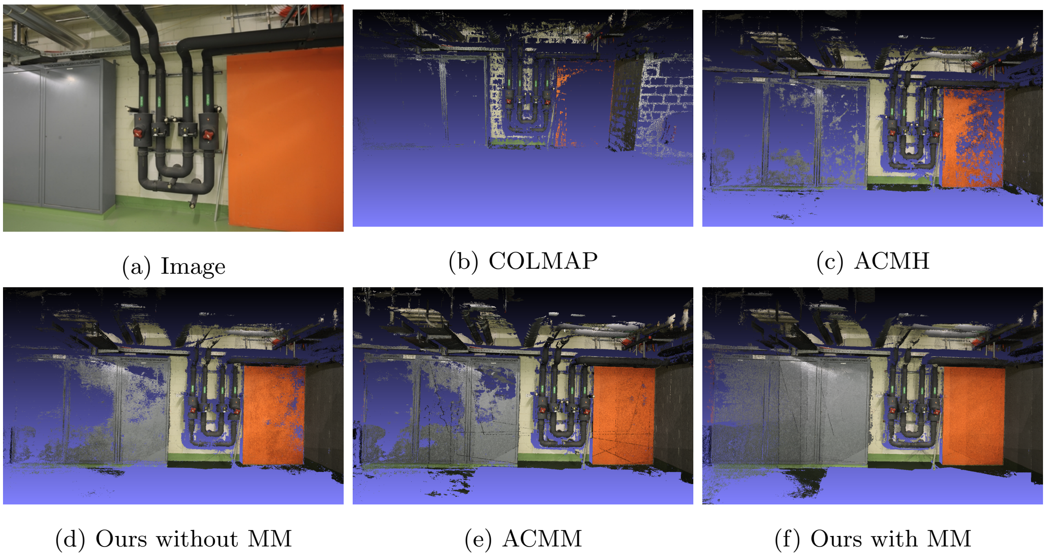
(c) ACMH

(d) Ours without MM

(e) ACMM

(f) Ours with MM

Figure 5: Depth map comparisons between different algorithms on ETH3D pipes dataset.



(a) Image

(b) COLMAP

(c) ACMH

(d) Ours without MM

(e) ACMM

(f) Ours with MM

Figure 6: Point cloud comparisons between different algorithms on ETH3D pipes dataset.

Bibliography

Bleyer, Michael, Christoph Rhemann and Carsten Rother (2011). ‘Patchmatch stereo-stereo matching with slanted support windows.’ In: *Bmvc.* Vol. 11, pp. 1–11.

Galliani, Silvano, Katrin Lasinger and Konrad Schindler (2015). ‘Massively parallel multiview stereopsis by surface normal diffusion’. In: *Proceedings of the IEEE international conference on computer vision*, pp. 873–881.

Hirschmuller, Heiko (2005). ‘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).* Vol. 2. IEEE, pp. 807–814.

Mei, Xing et al. (2011). ‘On building an accurate stereo matching system on graphics hardware’. In: *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops).* IEEE, pp. 467–474.

Schönberger, Johannes L et al. (2016). ‘Pixelwise view selection for unstructured multi-view stereo’. In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14*. Springer, pp. 501–518.

Schops, Thomas et al. (2017). ‘A multi-view stereo benchmark with high-resolution images and multi-camera videos’. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3260–3269.

Xu, Qingshan and Wenbing Tao (2019). ‘Multi-Scale Geometric Consistency Guided Multi-View Stereo’. In: *Computer Vision and Pattern Recognition (CVPR).*