Compacting Singleshot Multi-Plane Image via Scale Adjustment

Max Bergfelt
Lund University
Keio University

Viktor Larsson
Lund University

Hideo Saito*
Keio University

Shohei Mori†
Graz University of Technology
Keio University

Figure 1: Our algorithm optimizes the placement of individual layers in singleshot MPI. (a) While the original MPI layers lay uniformly in an arbitrarily scaled diopter space, (b) our algorithm adjusts the individual layer scales with a given depth information of dense or sparse reconstruction from an AR-tracked camera (dark red dots). After the optimization, multiple layers with an identical location and scale are merged into one layer, and thus, the MPI data is compressed. The right images demonstrate rendering from "ten frames away." Without scale adjustment (a), the rendering shows "stack-of-cards" artifacts. With our scale adjustment (b), no clear artifacts appear.

Abstract

A recent singleshot multiplane image (MPI) generation enables to copy an observed reality within a camera frame into other reality domains via view synthesis. While the scene scale is unknown due to the nature of singleshot MPI processing, camera tracking algorithms can estimate depth within the application world coordinate system. Given such depth information, we propose to adjust the scale of singleshot MPI to that of the currently observed scene. We find the individual scales of the MPI layers by minimizing the differences between the depth of MPI rendering and that of camera tracking. We eventually found that many layers fall within a close depth. Therefore, we merge such layers into one to compact the MPI representation. We compared our method with baselines using real and synthetic datasets with dense and sparse depth inputs. Our results demonstrate that our algorithm achieves higher scores in image metrics and reduces MPI data amount by up to 78%.

Index Terms: Computing methodologies—Computer graphics—Image manipulation—Image-based rendering; Human-centered computing—Human computer interaction (HCI)—Interaction paradigms—Mixed / augmented reality

1 Introduction

View synthesis techniques generate novel views by interpolating scene-registered photographs and classically known as image-based rendering (IBR) [34]. Such techniques can capture reality and reproduce the scene itself or scene content and therefore have been considered one of the key technologies for virtual and augmented reality (VR/AR) [26]. The major usages would include introducing real-world content into the virtual world (i.e., augmented virtuality [21]) and real-scene analysis for better lighting and object interactions in AR [22]. Literature has addressed an efficient and effective scene representation using scene proxies [4], camera focus plane proxies [15], view-dependent textures [8], and deep neural networks (DNN) [19, 20].

A modern multi-layer scene representation comprises multiple RGB+α layers to represent a scene with point ambiguity. The layers proxy can be a set of perpendicular planes in view frustum (i.e., multi-plane image; MPI) [11, 16, 19, 36] or concentric spheres (i.e., multi-sphere image; MSI) [1, 3] spaced uniformly in diopter space for efficient scene coverage. Overlaying all layers from back to front results in a complete opaque scene from a viewpoint. The volume representation is generated via multi-view plane-sweep volume analysis using DNN. Compared to neural radiance field (NeRF) [20, 30], multi-layer scene representation is explicit and usually requires fewer images satisfying the sampling theory to start with. Once generated, layers can be rendered in real time using a conventional rasterization rendering pipeline (e.g., graphics API with alpha blending capability: OpenGL). An analogy to singleshot depth estimation [18], the recent advances in DNN have enabled multi-layer image generation from a single photograph [14, 17, 31].

Singleshot MPI generators further add the benefit for AR since the user can immediately start scene rendering in the application by taking one photograph. However, one drawback of the approaches makes the usage AR practically challenging. Singleshot MPI is estimated up to scale. Therefore, the rendered scene does not match the real-world scale, or can be either too small or too large.

To address this issue, we propose to adjust the multi-layer scene scale to the scene depth observed via a camera tracking system (i.e., simultaneous localization and mapping; SLAM). Our algorithm can optimize individual layer scales so that MPI rendering appears to match up with the depth appearance of the SLAM. Note that multi-layer scene representation does not have explicit depth but soft depth [24, 29], and therefore, only depth rendering can be a reference for depth adjustment. We found that our individual layer scale optimization would lead to multiple layers that fall within a practically identical depth and scale. We further merge those layers into one to reduce the number of layers for more compact view synthesis. Compared to the state-of-the-art layer adjustment approaches [14, 23], only our algorithm can reduce the data amount.

*e-mail: hs@keio.jp
†e-mail: s.mori.jp@ieee.org

In summary, our contributions are the following three folds.

- We propose an algorithm for individual MPI layer scale alignment via analyzing a given depth map and MPI depth rendered
- We further propose to merge multiple layers that fall into a close depth level after the scale adjustment to reduce the data amount of MPI
- We provide comprehensive evaluations using real and synthetic data with dense and sparse depth information.

2 Related Work

We review multi-layer scene representation for view synthesis. We overview multi-layer scene representation from multi-view input and then singleshot MPI generation, which is our main focus. Finally, we discuss the model-scene scale fitting problem in AR.

2.1 Multi-layer Scene Representation

Multi-layer scene representation consists of a set of a fixed number of ordered D RGB+α texture mapped proxies in diopter space [19] (i.e., layers, $l_i \in \{\mathbf{c}_i, \alpha_i\}$ $(1 \leq i \leq D)$). The additional fourth channel can be either transparency [19] or density [17]. Depending on the proxy shape, multi-layer scene representation can be categorized either to MPI (i.e., multiple planes expanding from the image center to the space) or MSI (i.e., concentric spheres). Hereafter, without loss of generality, we use multi-layer scene representation and MPI interchangeably as MPI is our main focus. Rather than having explicit scene depths, MPI represents a scene point with multiple overlapping semi-transparent pixels. The more certain the point existence is, the more opaque the pixels are. Rendering such layers, l_i, from back ($i = 1$) to front ($i = D$) with over alpha composition results in a perspective image, $\mathscr{I} \in \{\mathbf{c}\}$,

$$\mathbf{c} = \sum_{i=1}^{D} \left(\mathbf{c}_i \alpha_i \prod_{j=i+1}^{D} (1 - \alpha_j) \right). \tag{1}$$

Here, we omit the layer warping using Homography from MPI view to the current target view for brevity. MPI rendering is, by nature, highly efficient on the conventional rasterization rendering scheme running on GPU.

While classic approaches use color or depth analysis from given multi-view images [24, 29], recent approaches can successfully infer MPI via differentiable renderer with DNN optimization framework [19, 36]. Multi-view images are first converted into plane-sweep volumes and stacked to be fed to a pre-trained network. The network "sees" the disparities in PSV data and outputs RGB+α layers. With given multi-view images, the scene scale is known as a baseline length between input views.

MPI is considered as a compact form of a light field [19]. Therefore, the plenoptic sampling theory [5] applies to MPI, and its spacial sampling rate is D^2 times more efficient in a 2D grid than that of a raw light field [19]. Given N multi-view MPIs for broader scene coverage and fewer invalid pixel disocclusions, we can render individual views and blend them to complete the current view rendering.

$$c = \frac{\sum_{k=1}^{N} \alpha_k c_k}{\sum_{k=1}^{N} \alpha_k}. \tag{2}$$

Contrary to recently emerged NeRF representation [20, 30], MPI is explicit (thus human-readable and editable), highly GPU-friendly even for low-end GPU, and thus suitable for mobile VR/AR usages. However, MPI is more memory consuming as a single photograph becomes D images with α components. To efficiently bound the scene with meaningful pixel values, one may find the nearest and furthest layer depths during an iterative optimization [23]. Our approach, instead, is an add-on for existing singleshot MPI generator [31] and is also able to reduce the number of layers for data compression.

2.2 Singleshot Multi-layer Scene Generation

Recent advances in DNN enabled MPI generation from a singleshot photograph [14,31]. Such networks are trained from a massive video dataset, in which one view is rendered using an inferred MPI warped from another view. Similarly to singleshot DNN depth estimators [18], only relative poses are available for the view warping, and therefore, the scene scale is normalized during the training. As such, MPI is estimated up to the scale.

Considering AR use cases, one may wish to estimate MPI immediately after photographing the scene to start the AR application. However, the scene scale obtained from a camera tracking system and that of MPI are different. Therefore, an algorithm to match up with them is necessary. Naïvely rendering MPI will otherwise lead to "stack-of-cards" artifact (Figure 1a).

To solve this issue, we use MPI depth rendering and a depth map (Figure 1b), which is often available in many AR applicationsAR-Foundation[1]. Nonetheless, the latest approach, AdaMPI [14], even infers depth-adjusted MPI layers to encourage the best pixel distributions over the layers using a DNN-inferred depth map [25]. With the state-of-the-art DNN depth estimator, depth scale seems available using dataset-specific fine-tuning (e.g., for KITTI [13] and NYUv2 datasets [28]). Thus, the generated MPI can be rescaled. Due to our add-on nature to our baseline singleshot MPI generator [31], our algorithm never reaches higher quality rendering than AdaMPI [14] within the commonly scaled space. However, we put an emphasis on studying our scale adjustment algorithm to investigate how well it reduces the number of layers while keeping the rendering quality. In other words, we expect future singleshot MPI generators that can achieve higher quality rendering without layer scale adjustment and adapting our algorithm to them.

2.3 Scene-Map Scale Adjustment for AR

3D scene reconstructions (e.g., point clouds or dense reconstruction) from structure from motion (SfM) algorithms [27] and SLAM systems [10] are arbitrarily scaled without a common metric given by inertia sensor, stereo system, or a runtime scale adjustment algorithm. For AR applications, an internal tracking system and preserved map information do not match without scale adjustment. A typical example is to register the AR camera to the saved map data created using a different system [33].

Similarly to the scene-map scale adjustment, our goal is to implement a scene-MPI scale adjustment algorithm. The current scene scale is given by undergoing SLAM in an AR application. Therefore, we use SLAM map projection within the MPI generated view. We match the depth frame and MPI depth rendering since MPI does not have explicit depth but a depth map rendering instead.

3 The Proposed Method

Our pipeline follows the five steps below (Figure 2):

1. Single-view MPI generation (e.g., [17, 31])
2. Scene depth calculation (SLAM [10] or SfM [27])
3. MPI depth rendering (Section 3.1)
4. Layer scaling (Section 3.2)
5. MPI rendering (Section Equation 1)

[1] https://unity.com/unity/features/arfoundation

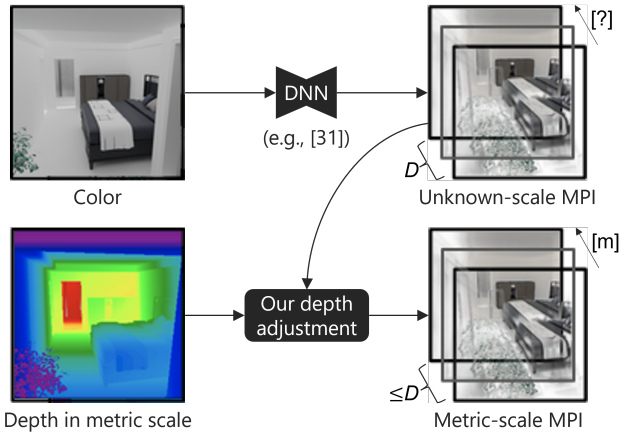

Figure 2: Pipeline. Our depth adjustment algorithm receives a scale-ambiguous singleshot MPI (e.g., from [31]) and a metric-scale depth map (e.g., from DepthLab [10]) and [25]) and outputs a metric-scale MPI with a reduced number of layers.

3.1 MPI Depth Rendering

We adjust individual MPI layer scales (i.e., depths) using MPI depth rendering, d_{MPI}, (e.g., [31]) and the corresponding external scene depth information using a SLAM tool, d_{SLAM}, (e.g., [10]). As demonstrated by Tuker et al. [31], rendering MPI with its depth pixels d_i or inverse depth d_i^{-1} generates a depth or disparity map, respectively, using the over operation in equation 1. For a depth map rendering, we can rewrite equation 1 as

$$d_{\text{MPI}} = \sum_{i=1}^{D} \left(d_i \alpha_i \prod_{j=i+1}^{D} (1-\alpha_j) \right). \tag{3}$$

Further rewriting the α value related terms as a constant value, c_i, we further simplify the equation as

$$d_{\text{MPI}} = \sum_{i=1}^{D} d_i c_i = \mathbf{d} \cdot \mathbf{c}. \tag{4}$$

3.2 Individual Layer Scaling

The strategy for individual layer depth calculation is to use SLAM depth information, $\mathbf{d}_{\text{SLAM}}$, as a set of reference depth values, and minimize the error between the rendered depths, $\mathbf{d}_{\text{MPI}}$, and the reference depths, $\mathbf{d}_{\text{SLAM}}$. To this end, we collect all N rendered depth pixels where the corresponding SLAM depth pixels exist and construct an $N \times D$ matrix of constants of $\mathbf{c}$, $\mathbf{A}$, to solve the following linear equation,

$$\mathbf{d}_{\text{MPI}} = \mathbf{A}\mathbf{d}. \tag{5}$$

Therefore, we solve the following for $\mathbf{d} = (d_1, d_2, ..., d_D)^\intercal$ of layer depths,

$$\min_{\mathbf{d}} |\mathbf{d}_{\text{SLAM}} - \mathbf{d}_{\text{MPI}}|. \tag{6}$$

Linear Constraints. Optimization of equation 3.2 without any reasonable constraints may result in negative depth values (i.e., layers existing back of the camera frustum) or shuffled-order layers. To avoid such inconvenient layers, we set the following two constraints for solving equation 3.2,

$$\min_{\mathbf{d}} |\mathbf{d}_{\text{SLAM}} - \mathbf{d}_{\text{MPI}}| \quad \text{s.t. } d_i \geq 0,\ d_i \geq d_{i+1}. \tag{7}$$

Namely, we force the solver not to take negative depth values (i.e., $d_i \geq 0$) and keep the order (i.e., $d_i \geq d_{i+1}$). The strict inequality is not used for the zero constraints to allow redundant layers to be removed from the rendering. Likewise, strict inequalities are not used for the ordering constraint either. As long as the rendering order of the layers is preserved, layers placed at the same depth do not cause a problem. It should also be noted that an upper limit constraint for the depths would be excessive. Since all rows of $\mathbf{A}$ sum to 1, any layer depth value is naturally constrained to the maximum value of $\mathbf{d}$, the highest value of the ground truth depth map.

Solver. For solving the constraint linear problem, we use an implementation of the alternating direction method of multipliers (ADMM) [2]. ADMM optimizes one variable at a time while keeping the others fixed and uses dual variables to enforce constraints. The implementation used proximal operators and a solver was constructed using the python package `proxop` [6].

3.3 Layer Merging

We found that the optimization often resulted in adjacent layer depths with the same depth values due to the design of the linear problem and in particular the ordering constraint. Each time an optimal solution lies close enough to the edge of this constraint, the optimal solution within the constrained set will lie right on the edge, placing the layers at the same depth.

We merge such layers into one layer to reduce the total number of layers in MPI. We, therefore, render a new layer in the same way as the adjacent layers would be rendered using equation 1. For two images, $\{C_1, \alpha_1\}$ and $\{C_2, \alpha_2\}$, the resulting new image is calculated by

$$\begin{aligned} \mathbf{c}_{1,2} &= \frac{\mathbf{c}_1 \alpha_1 (1-\alpha_2) + \mathbf{c}_2 \alpha_2}{\alpha_2} \\ \alpha_{1,2} &= \alpha_2 + \alpha_1 (1-\alpha_2). \end{aligned} \tag{8}$$

Performing the algorithm iteratively on all layers placed at the same depth in a front-to-back manner allows an arbitrary number of images to be merged. Since this merging step can be seen as a pre-rendering for the layers of the MPI that contribute with the same perspective effects, it will result in no loss of image quality or visual effects in the MPI. Any image rendered from the MPI will be the same regardless if the layers are merged in advanced, or rendered at the same position during run-time. The advantage of layer merging, however, is the MPI file size compression.

4 Evaluations

There are only a few attempts of singleshot MPI generator without layer scaling during the inference [17, 31], from which we selected the first implementation by Tuker et al. [31] as our core MPI generator that has been widely tested with the provided pre-trained weights. Our goal here is to evaluate visual quality when a rendering camera moves in a metric scale and compare our approach with baseline depth adjustment approaches. We also measure compression rates after our algorithm is applied to datasets.

4.1 Baseline Scaling Algorithms

We prepared two different baselines.

Uniform scaling To design scene-independent scaling, we scale MPI layers uniformly so that the layers lay within the fixed minimum and maximum depth range,

$$d_i' = \frac{1}{1/d_{\max} + \frac{(1/d_{\min} - 1/d_{\max})(i-1)}{D-1}}, \tag{9}$$

where d_i', $d_{\min}$, and $d_{\max}$ are the scaled layer depth, the minimum layer depth, and the maximum layer depth, respectively. We set

Table 1: Dataset summary.

Modality	Synthetic	Real
Core dataset	3D-FRONT (BlenderProc)	RealEstate10K
Scenes	17	38
Number of MPI	615 ($\bar{X} = 36.2(8.8)$)	667 ($\bar{X} = 17.6(3.8)$)
Frame interval	5 cm	5 frames
Depth data	Dense depth map	Sparse points

$d_{\text{min}} = 1$m and $d_{\text{max}} = 100$ m, which are the default values in the original implementation[2].

Min-max scaling Similarly to the Uniform scaling, the Min-max scaling takes the minimum and maximum depth observed in the scene. Namely, we set d_{min} and d_{max} in equation 4.1 to the minimum and maximum depth values obtained from a SLAM system, respectively.

4.2 Datasets

We used real and synthetic datasets. Table 1 summarizes our synthetic and real datasets.

Synthetic dataset. To have full control over camera parameters and color and depth images as ground truth of our algorithm, we created a synthetic dataset, although image quality is less promising and far from the trained feature domain for our core MPI generator [31].

We used BlenderProc2 [9] in combination with 3D furnished rooms with layouts and semantics (3D-FRONT) [12] to generate our synthetic image dataset. BlenderProc2 is a pipeline for the open source computer graphics software Blender [7]. The software allows the photorealistic rendering of Blender scenes for the sake of training convolutional neural networks. 3D-FRONT [12] provides various fully furnished high-quality textured indoor environments. We chose the assets because of their similarities to the real image RealEstate10K dataset [36], on which our core singleshot MPI generator [31] was trained. We rendered 17 different scenes with a camera moving horizontally to create motion stereo views (i.e., at 5 cm and 10 cm aside from the original viewpoint). We consequently generated 615 MPI in total.

To ensure that generated synthetic images would work well with the network model, the camera parameters were set up to mimic the training data for the pre-trained model, the YouTube videos in RealEstate10K [36]. Upon the camera intrinsic parameters provided in RealEstate10K dataset, most clips are in a 16:9 aspect ratio. From this fact, we end up with the typical field of view for the cameras as 90° horizontally and approximately 59° vertically. The synthetic images were generated at a 512×512 resolution and to make this consistent with the training data, both the horizontal and vertical field of view was set to the minimum 59°. In other words, we generated images to be identical to those in a 16:9 aspect ratio at a 90° horizontal and 59° vertical field of view that are then cropped to a $1:1$ aspect ratio. In addition, a multiple of 128 pixel resolution allows the model to be used without any padding or cropping. Finally, we generated a pair of color and depth images together with the intrinsic and extrinsic parameters.

Real dataset. We used a part of the RealEstate10K dataset [36], whose camera motions are random. To extract data for MPI generation, the text files in the dataset were processed and the corresponding frames were downloaded from the YouTube videos. While the authors of the dataset made their best effort to select sequences without blurry and distorted artifacts and as well as editing effects, some of the downloaded frames still contained large font descriptive texts and transition effects (e.g., fading out at the end of the sequence). We, therefore, manually sorted out such frames.

For the selected frames, we downloaded every fifth frame, resulting in between 10 and 25 images per sequence. The reason for this was to mimic the test data that [31] used to test their model, which consisted of images extracted from every 5 and 10 frames in the video sequences. We extracted 667 MPI over 38 scenes overall. For depth information at the frames, we project sparse point clouds to the frames.

[2]https://single-view-mpi.github.io/

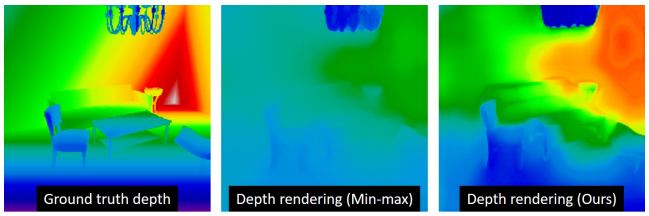

Figure 3: MPI depth rendering in comparison with the synthetic ground truth. Our depth optimization using dense depth points captures the better scene structure in the appearance (right) compared to that of the second-best scored Min-max scaling approach (middle).

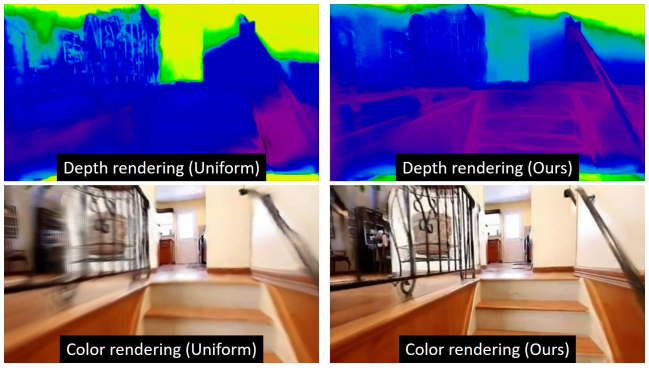

Figure 4: MPI depth and color rendering for qualitative comparisons in the real dataset. Our depth optimization using sparse depth points results in a higher quality with fewer "stack-of-cards" artifacts (top) both in depth (left) and color (right) MPI rendering when compared to the second-best scored Uniform approach (bottom).

4.3 Evaluation Metrics

Image quality metrics. We measured peak signal-to-noise ratio (PSNR), structural similarity index measure (SSIM) [32], and learned perceptual image patch similarity (LPIPS) [35] between the ground truth and rendered MPI.

Due to the perspective nature of Homography upon MPI rendering, blank pixels would appear around the image borders. To remove such unnecessary deterioration from the evaluations, we cropped 5% of the image at each side.

Layer reduction ratio. To measure how much our algorithm can save the amount of MPI data, we compute the number of MPI layers with our algorithm applied over that of the original.

4.4 Results

Rendering quality Table 2 and 3 summarize the results of the image quality metrics. All approaches show decreased scores with larger baselines both in our synthetic and real datasets. While our depth optimization approach achieved the best mean scores in both datasets, the second-best-scored approach is different depending on

Table 2: Synthetic scene results.

Algorithm	Baseline	PSNR (↑)	SSIM (↑)	LPIPS (↓)
Uniform	5 cm	26.1	0.877	0.103
	10 cm	23.4	0.833	0.164
Min-max	5 cm	27.1	0.890	0.095
	10 cm	24.2	0.848	0.144
Ours	5 cm	**28.9**	**0.919**	**0.085**
	10 cm	**25.9**	**0.880**	**0.115**

Table 3: Real scene results.

Algorithm	Baseline	PSNR (↑)	SSIM (↑)	LPIPS (↓)
Uniform	5 frames	22.6	0.727	0.162
	10 frames	20.0	0.649	0.237
Min-max	5 frames	20.6	0.664	0.209
	10 frames	18.2	0.585	0.293
Ours	5 frames	**23.7**	**0.763**	**0.133**
	10 frames	**21.1**	**0.687**	**0.188**

Table 4: Layer reduction summary.

Dataset		Number of layers		Reduction ratio (%)
		Total	Per view	
Synthetic	Original	19,648	32.0 (0.0)	0.0 (0.0)
	Ours	5,245	8.5 (2.2)	73.3 (6.8)
Real	Original	21,504	32.0 (0.0)	0.0 (0.0)
	Ours	4,300	6.4 (2.6)	78.7 (8.8)

the image modality. For the synthetic dataset, Min-max takes the second-best and, for the real dataset, Uniform did so.

Figure 3 and 4 present qualitative results of depth rendering in the synthetic dataset and depth and color rendering in the real dataset, respectively. With our algorithm, the depth values appear reasonably distributed, showing a clearer scene structure.

Layer reduction Table 4 shows a summary of the layer reduction results. Overall, more layer depths were placed at the same depths in the real dataset than the synthetic dataset. The variance was also greater. The maximum average size reduction for a single scene in the real dataset was 91.7% compared to 80.1% in the synthetic dataset. The minimum average reduction in a scene, however, was found in the synthetic dataset at 63.9% compared to the minimum of 65.6% in the real dataset. In the real dataset 3.9% of all MPI were reduced to two layers or less while the same number in the synthetic dataset was 0.16%. The same numbers for three layers or less were 12.4% and 1.0%, respectively. The minimum amount of layers in any MPI after reduction in the synthetic dataset was 15 and 14 in the real dataset.

5 Limitations and Future Work

Our approach accomplishes scale adjustments of individual MPI layers and successfully reduces the MPI data amount. However, we observe some limiting factors in rendering quality.

Inaccurate real scene depth points. Min-max scaling performs better than Uniform scaling in the synthetic dataset, while in the real dataset, Uniform scaling works better in image quality metrics. We speculate that inaccurate depth points in the real dataset made Min-max scaling unstable. On the other hand, Uniform scaling is ultimately independent of the scene depth.

Sparse points from a SLAM system depend on the scene details and can unevenly distribute over the field of view. Therefore, texture-rich areas are weighted more than texture-less areas in the optimization. However, scenes with evenly distributed SLAM points make MPI inference challenging in general, and therefore, our optimization tends to be fragile anyway. Figure 5 shows a typical example of such cases. Note that the network [31] is trained on real estate images, while the scene shows a wooden house in a forest.

Figure 5: Failure case. Although well-distributed depth points exist (left), our approach falsely presents blurry artifacts. In this example, ours shows a strong MPI quality-dependency (right). The Uniform scaling approach, which is ultimately stable over the frames, would be more suitable for such scenes.

Inaccurate MPI depth rendering. MPI depth rendering can be squashed or rough compared to the actual details in a depth map. In other words, different depths can appear in an area with similar values in MPI depth rendering, or similar depths can appear in different pixels in depth rendering, both of which make the layer placement ambiguous. Our insights, however, can be network-specific.

Optimization constraints. Although the optimization constraints are necessary for reasonable layer placements, they may also infer a limitation. Due to the constraint for layer order, one layer can stop the other layer, and therefore, one layer at a non-optimal layer can also force the other layers at non-optimal depths.

6 Conclusion

We present an algorithm to adjust the scales of individual MPI layers so that the appearance of MPI depth rendering matches the depth points from a tracking system. We further propose to merge multiple MPI layers at the same resultant depth to reduce the data amount. We evaluated our approach in the synthetic dataset with ground truth dense depth maps and the real dataset with sparse depth points. Our results demonstrate that our approach achieves the best performance in comparison with two baseline approaches of uniform scaling and min-max depth-based scaling. After reducing the layer numbers, the resultant MPI is 73.3 % and 78.7 % less data size in the synthetic and real datasets, respectively.

Acknowledgments

This work was supported by the Austrian Science Fund FWF (grant no. P33634).

References

[1] B. Attal, S. Ling, A. Gokaslan, C. Richardt, and J. Tompkin. Matryodshka: Real-time 6dof video view synthesis using multi-sphere images. In *Proc. European Conference on Computer Vision (ECCV)*, pp. 441–459. Springer, 2020.

[2] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3:1–122, 01 2011.

[3] M. Broxton, J. Flynn, R. Overbeck, D. Erickson, P. Hedman, M. Duvall, J. Dourgarian, J. Busch, M. Whalen, and P. Debevec. Immersive light field video with a layered mesh representation. *ACM Transactions on Graphics (TOG)*, 39(4):86–1, 2020.

[4] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen. Unstructured lumigraph rendering. In *Proc. Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pp. 425–432, 2001.

[5] J.-X. Chai, X. Tong, S.-C. Chan, and H.-Y. Shum. Plenoptic sampling. In *Proc. Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pp. 307–318, 2000.

[6] G. Chierchia, E. Chouzenoux, P. L. Combettes, and J.-C. Pesquet. *The proximity operator repository. user's guide, 2020*.

[7] B. O. Community. *Blender - A 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.

[8] P. Debevec, Y. Yu, and G. Borshukov. Efficient view-dependent image-based rendering with projective texture-mapping. In *Rendering Techniques: Proc. the Eurographics Workshop*, pp. 105–116. Springer, 1998.

[9] M. Denninger, M. Sundermeyer, D. Winkelbauer, Y. Zidan, D. Olefir, M. Elbadrawy, A. Lodhi, and H. Katam. Blenderproc. *arXiv preprint arXiv:1911.01911*, 2019.

[10] R. Du, E. Turner, M. Dzitsiuk, L. Prasso, I. Duarte, J. Dourgarian, J. Afonso, J. Pascoal, J. Gladstone, N. Cruces, S. Izadi, A. Kowdle, K. Tsotsos, and D. Kim. Depthlab: Real-time 3d interaction with depth maps for mobile augmented reality. In *Proc. ACM Symposium on User Interface Software and Technology (UIST)*, UIST '20, pp. 829–843. ACM, 2020.

[11] J. Flynn, M. Broxton, P. Debevec, M. DuVall, G. Fyffe, R. Overbeck, N. Snavely, and R. Tucker. Deepview: View synthesis with learned gradient descent. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2367–2376, 2019.

[12] H. Fu, B. Cai, L. Gao, L.-X. Zhang, J. Wang, C. Li, Q. Zeng, C. Sun, R. Jia, B. Zhao, et al. 3d-front: 3d furnished rooms with layouts and semantics. In *Proc. International Conference on Computer Vision (ICCV)*, pp. 10933–10942, 2021.

[13] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3354–3361. IEEE, 2012.

[14] Y. Han, R. Wang, and J. Yang. Single-view view synthesis in the wild with learned adaptive multiplane images. In *Proc. Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 2022.

[15] A. Isaksen, L. McMillan, and S. J. Gortler. Dynamically reparameterized light fields. In *Proc. Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pp. 297–306, 2000.

[16] R. Ishikawa, H. Saito, D. Kalkofen, and S. Mori. Multi-layer scene representation from composed focal stacks. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 2023.

[17] J. Li, Z. Feng, Q. She, H. Ding, C. Wang, and G. H. Lee. Mine: Towards continuous depth mpi with nerf for novel view synthesis. In *Proc. International Conference on Computer Vision (ICCV)*, 2021.

[18] Z. Li and N. Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2041–2050, 2018.

[19] B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, N. K. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 2019.

[20] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *Proc. European Conference on Computer Vision (ECCV)*, 2020.

[21] P. Milgram, H. Takemura, A. Utsumi, and F. Kishino. Augmented reality: A class of displays on the reality-virtuality continuum. In *Telemanipulator and telepresence technologies*, vol. 2351, pp. 282–292. SPIE, 1995.

[22] S. Mori, O. Erat, W. Broll, H. Saito, D. Schmalstieg, and D. Kalkofen. Inpaintfusion: Incremental rgb-d inpainting for 3d scenes. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 26(10):2994–3007, 2020.

[23] J. Navarro and N. Sabater. Deep view synthesis with compact and adaptive multiplane images. *Signal Processing: Image Communication*, 107:116763, 2022.

[24] E. Penner and L. Zhang. Soft 3d reconstruction for view synthesis. *ACM Transactions on Graphics (TOG)*, 36(6):1–11, 2017.

[25] R. Ranftl, A. Bochkovskiy, and V. Koltun. Vision transformers for dense prediction. In *Proc. International Conference on Computer Vision (ICCV)*, pp. 12159–12168, 2021.

[26] D. Schmalstieg and T. Höllerer. *Augmented Reality: Principles and Practice*. Addison-Wesley Professional, 2016.

[27] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[28] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. *Proc. European Conference on Computer Vision (ECCV)*, 7576:746–760, 2012.

[29] R. Szeliski and P. Golland. Stereo matching with transparency and matting. *International Journal of Computer Vision (IJCV)*, 32(1):45–61, 1999.

[30] A. Tewari, O. Fried, J. Thies, V. Sitzmann, S. Lombardi, Z. Xu, T. Simon, M. Nießner, E. Tretschk, L. Liu, B. Mildenhall, P. Srinivasan, R. Pandey, S. Orts-Escolano, S. Fanello, M. Guo, G. Wetzstein, J.-Y. Zhu, C. Theobalt, M. Agrawala, D. B. Goldman, and M. Zollhöfer. Advances in neural rendering. In *ACM SIGGRAPH 2021 Courses*, SIGGRAPH '21. Association for Computing Machinery, New York, NY, USA, 2021.

[31] R. Tucker and N. Snavely. Single-view view synthesis with multiplane images. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[32] Z. Wang, A. Bovik, and H. Sheikh. Structural similarity based image quality assessment. *Digital Video Image Quality and Perceptual Coding, Ser. Series in Signal Processing and Communications*, 11 2005.

[33] M. Yamaguchi, T. P. Truong, S. Mori, V. Nozick, H. Saito, S. Yachida, and H. Sato. Superimposing thermal-infrared data on 3d structure reconstructed by rgb visual odometry. *IEICE Trans. on Information and Systems*, 101(5):1296–1307, 2018.

[34] C. Zhang and T. Chen. A survey on image-based rendering—representation, sampling and compression. *Signal Processing: Image Communication*, 19(1):1–28, 2004.

[35] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[36] T. Zhou, R. Tucker, J. Flynn, G. Fyffe, and N. Snavely. Stereo magnification: Learning view synthesis using multiplane images. In *ACM Transactions on Graphics (TOG)*, July 2018.