Improving Multiplane Images with Deformable Layers

Alpha-Weighted Mesh Deformation for Single-View MPI-Based View Synthesis

Lachezar Topalov[1]

Supervisor: Petr Kellnhofer [1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 21, 2026

Name of the student: Lachezar Topalov
Final project course: CSE3000 Research Project
Thesis committee: Petr Kellnhofer, Joana de Pinho Gonçalves

An electronic version of this thesis is available at http://repository.tudelft.nl/.

Abstract

Multiplane Images (MPIs) are an efficient representation for single-view novel-view synthesis, but their fronto-parallel planes limit how well they can represent slanted or non-planar scene geometry. This paper investigates whether MPI-based single-view view synthesis can be improved by replacing flat MPI planes with alpha-weighted deformable mesh layers. The proposed method takes a predicted MPI as input, converts each RGBA layer into a triangular mesh, and displaces its vertices using an expected disparity estimate derived from the MPI alpha distribution. The goal is to add local geometric flexibility while preserving the layered visibility structure of the original MPI.

The method is evaluated on RealEstate10K and LLFF using PSNR, SSIM, and LPIPS, with additional runtime measurements and a controlled synthetic Blender scene. On predicted MPIs, the deformation produces only marginal changes and does not yield a meaningful improvement over the flat MPI baseline, while also increasing rendering cost. However, in the controlled RGB-D experiment, where the MPI is constructed from accurate depth and uses fewer layers, the deformable representation better follows slanted geometry and improves over the flat baseline. These results suggest that deformable MPI layers can be beneficial when the input layered geometry is reliable, but are not sufficient as a standalone post-processing step for learned single-view MPI predictions.

1 Introduction

Novel-view synthesis aims to generate images of a scene from camera viewpoints that were not directly observed. This problem is central to applications such as virtual and augmented reality, image-based rendering, and interactive 3D photography. It is especially challenging in the single-view setting, where a method must infer both scene geometry and the appearance of regions that are hidden in the input image. Multiplane Images (MPIs) provide an attractive representation for this setting because they combine a simple geometric structure with efficient rendering. An MPI represents a scene as an ordered stack of fronto-parallel RGBA planes placed at fixed depths. Each plane stores color and opacity, and novel views are synthesized by warping the planes to a target camera and alpha-compositing them from back to front. The representation was popularized for learned view synthesis in stereo magnification by Zhou et al. [9]. Tucker and Snavely later showed that MPIs can be predicted from a single RGB image using scale-invariant view-synthesis supervision [6].

Despite their effectiveness, classical MPIs impose a strong geometric assumption: each layer is a flat plane with constant depth. This makes rendering efficient and supports layered visibility reasoning, but it also limits how accurately MPIs can represent non-planar scene structure. Slanted, curved, and irregular surfaces must be approximated by distributing opacity across multiple fronto-parallel planes. This can lead to stair-step geometry, ghosting across adjacent layers, or inaccurate motion parallax when the virtual camera moves. Increasing the number of MPI planes may reduce these artifacts, but it increases memory and computation while still representing the scene as a stack of planar slabs rather than as surfaces aligned with the underlying geometry.

This limitation motivates the central research question of this project: *can MPI-based single-view view synthesis be improved by replacing fronto-parallel MPI planes with alpha-weighted deformable mesh layers?* The goal is not to discard the MPI representation, since its layered alpha structure is useful for modelling foreground-background separation and disoccluded content. Instead, this work investigates whether the flat geometry of each MPI layer can be made more flexible while preserving MPI-style alpha compositing.

The proposed approach converts each MPI plane into a triangular mesh layer. Vertices are initialized at the original plane depth and are then displaced using a constrained, alpha-weighted residual displacement field. Intuitively, regions that strongly belong to a particular layer are allowed to move toward the estimated local scene surface, while low-opacity regions remain close to their original plane depth. This design is intended to avoid the main failure mode of a naive depth-mesh conversion: collapsing the layered MPI representation into a single surface and losing the occlusion reasoning provided by separate foreground and background layers. By retaining MPI color and alpha layers while allowing local geometric deformation, the proposed representation aims to improve parallax on curved and slanted surfaces without sacrificing the layered structure that makes MPIs effective.

The method is evaluated on RealEstate10K [9] and LLFF [3] using PSNR, SSIM [7], and LPIPS [8], together with runtime measurements and a controlled synthetic Blender scene. The results show that deformable layers do not meaningfully improve predicted MPIs on the tested datasets, but can improve an RGB-D-derived MPI under cleaner geometric conditions. The main contribution of this work is therefore an alpha-weighted deformable MPI representation and an evaluation of its limitations as a post-processing method for single-view MPI prediction.

The remainder of the paper reviews related work, describes the deformable MPI method, evaluates it on learned and controlled MPI settings, and discusses its limitations and future directions.

2 Related Work

This section reviews work related to the two properties this project aims to combine: the layered visibility structure of MPIs and the geometric flexibility of depth- and mesh-based representations. The discussion first covers MPI-based view synthesis and the fronto-parallel plane assumption used by the baseline representation. It then considers layered depth and mesh-based methods, which motivate the use of more explicit geometry for parallax and disocclusion handling. Finally, it reviews deformable layered geometry, which is most closely related to replacing fixed MPI planes with locally deformable layers.

2.1 Multiplane Images for View Synthesis

Multiplane Images are a layered scene representation designed for efficient image-based rendering. Instead of reconstructing a complete 3D model, an MPI stores color and

opacity on a fixed set of fronto-parallel depth planes. Novel views can then be generated by applying homographies to each plane and compositing the results in depth order. Zhou et al. [9] introduced MPIs as a learned representation for stereo magnification, showing that a neural network can predict an MPI from a small-baseline stereo pair and synthesize nearby views. This formulation is attractive because it avoids the need for explicit mesh reconstruction while still supporting visibility reasoning through alpha compositing.

Mildenhall et al. [3] further demonstrated the practical relevance of MPIs in Local Light Field Fusion. Their method uses local MPI-based light fields to synthesize novel views from sparsely sampled input images, showing that MPI representations can support practical view synthesis beyond the original stereo magnification setting. This work is relevant because it reinforces the usefulness of MPIs as efficient layered rendering representations. At the same time, it retains the same basic geometric assumption as other MPI methods: scene content is represented on fronto-parallel depth planes.

While these methods use multiple input images or sparse captured views, Tucker and Snavely extended the MPI formulation to the more difficult single-view setting [6]. Their method predicts an MPI from one RGB image and uses scale-invariant view-synthesis supervision, making it possible to train on video data without requiring metric camera scale. This work is the primary baseline for the present project. It demonstrates that MPIs are effective for single-view view synthesis, but it retains the same geometric structure as earlier MPI methods: a fixed stack of fronto-parallel planes. As a result, the predicted representation can synthesize plausible novel views, but the geometry of curved or slanted surfaces is still approximated by opacity distributed across flat layers.

The present work builds directly on this MPI formulation. Rather than changing the image-prediction network itself, it investigates whether the predicted MPI representation can be made more geometrically expressive after prediction. The main difference from classical MPI rendering is that each layer is no longer treated as a rigid fronto-parallel plane. Instead, each MPI layer is converted into a triangular mesh whose vertices can be displaced according to layer ownership and estimated scene depth.

2.2 Layered Depth and Mesh-Based Representations

A related line of work represents scenes using depth-aware layers or meshes rather than fixed fronto-parallel planes. Layered Depth Images were introduced by Shade et al. [4] as an image-based rendering representation that stores multiple depth samples along each camera ray, allowing the representation to model surfaces hidden behind the foremost visible surface. More recently, Shih et al. [5] used this idea for 3D photography by constructing a layered depth representation from a single RGB-D image and inpainting missing color and depth information behind foreground objects. This approach can produce convincing parallax effects because the representation is explicitly tied to scene depth and pixel connectivity.

Such layered depth methods address a limitation of flat MPI geometry, but they differ from the setting considered here. They typically rely on an input depth map or an explicit RGB-D representation, whereas the method in this project starts from an MPI predicted from a single RGB image. In addition, a direct conversion from MPI to a single depth mesh risks discarding the useful layered alpha structure of the MPI. Foreground and background content that are separated across MPI layers may collapse into one visible surface, which can harm disocclusion handling. This motivates a representation that keeps the MPI layers but gives each layer more local geometric flexibility.

2.3 Deformable Layered Geometry

Layered mesh representations have also been used in broader view-synthesis systems. For example, Broxton et al. [1] use a layered mesh representation for immersive light field video, showing that layered meshes can support high-quality free-viewpoint rendering when sufficient multi-view capture data is available. Although this setting differs from the single-view problem considered in this project, it supports the broader idea that layered geometry can provide a useful compromise between planar image-based representations and full 3D reconstruction.

The closest conceptual connection to this project is mesh-based single-view view synthesis. Worldsheet, proposed by Hu et al. [2], represents a scene using a deformable 3D sheet that is fitted to the visible scene structure. Compared with a fixed stack of fronto-parallel planes, this type of representation can better align with slanted and curved surfaces. Worldsheet also studies extensions with multiple sheets, making it particularly relevant to the idea of combining mesh deformation with layered visibility.

The present work differs in its starting point and research goal. Worldsheet proposes a mesh-sheet representation for single-image view synthesis, whereas this project asks whether an existing MPI representation can be improved by deforming its layers. This distinction is important because MPIs already contain multiple alpha layers and potentially hallucinated background content. The challenge is therefore not only to fit geometry more accurately, but to do so without destroying the layer ordering and alpha compositing behaviour that make MPIs useful for occlusion reasoning.

Overall, prior work highlights a trade-off between the efficiency of MPI-style layered rendering and the geometric flexibility of depth- or mesh-based representations. Classical MPIs are efficient and preserve layered visibility, but approximate non-planar surfaces using fixed fronto-parallel planes. Depth meshes and deformable surfaces represent geometry more directly, but may lose the layered alpha structure needed for occlusion and disocclusion reasoning. This project explores an intermediate representation that preserves MPI-style compositing while allowing each layer to deform locally.

3 Method

This work modifies the geometric representation used after a single-view MPI has been predicted. The method does not retrain the MPI prediction network of Tucker and Snavely [6]. Instead, it takes the predicted MPI as input and converts its flat fronto-parallel layers into alpha-textured triangular mesh

layers. The purpose of this conversion is to use the additional geometric flexibility to better align high-confidence parts of each layer with the estimated local scene surface. At the same time, the method retains the layered color and opacity structure that allows MPIs to represent occlusions and disoccluded background content.

The proposed pipeline consists of the following steps:

1. Predict a flat MPI from a single input image using the pretrained single-view MPI model.
2. Compute the expected disparity map and layer ownership weights from the MPI alpha distribution.
3. Displace each layer's vertices toward the expected disparity according to the ownership weights and displacement constraints.
4. Convert each deformed layer into a triangular mesh with RGBA vertex values.
5. Render the layers from the target viewpoint and composite them to obtain the synthesized image.

The rest of this section describes these stages in order, from the input MPI representation to mesh construction, rendering, and compositing, along with implementation details.

3.1 Input MPI Representation

Let the predicted MPI consist of L ordered RGBA layers

$$\mathcal{M} = \{(C_i, \alpha_i, d_i)\}_{i=1}^{L}, \tag{1}$$

where $C_i \in [0,1]^{H\times W\times 3}$ is the RGB texture of layer i, $\alpha_i \in [0,1]^{H\times W\times 1}$ is its opacity, and d_i is the depth of the corresponding fronto-parallel plane. Layers are stored back-to-front, so $i = 1$ denotes the farthest layer and $i = L$ the nearest layer. The implementation uses $L = 32$ layers, following the single-view MPI model, with depths sampled uniformly in inverse depth between a back plane and a front plane.

The baseline MPI renderer treats every layer as a flat plane. Novel views are synthesized by warping each plane into the target camera and compositing the warped RGBA layers from back to front. The proposed method keeps the same layer ordering, color textures, and alpha maps, but replaces the constant-depth plane d_i with a spatially varying depth field $d_i'(u)$, where u denotes an image-grid location. This produces a deformed surface for each layer rather than a fronto-parallel plane.

3.2 Visible-Scene Disparity Estimate

The deformation is guided by the expected disparity implied by the original MPI. Since MPI planes are sampled in inverse depth, the method operates in disparity rather than depth. Let

$$\delta_i = \frac{1}{d_i} \tag{2}$$

be the disparity of layer i. At each pixel, the contribution of a layer to the final rendered reference image is determined by its opacity and by the visibility of that layer through all layers in front of it. The visibility of layer i is

$$V_i(u) = \prod_{j=i+1}^{L} (1 - \alpha_j(u)), \tag{3}$$

and the corresponding alpha-compositing weight is

$$w_i(u) = \alpha_i(u) V_i(u). \tag{4}$$

The expected visible-scene disparity is then computed as

$$\hat{\delta}(u) = \sum_{i=1}^{L} w_i(u)\delta_i. \tag{5}$$

This value summarizes where the original MPI places the visible surface along the camera ray. A naive approach would be to move every layer directly to this disparity. However, doing so would collapse the layered MPI into a single depth surface and would remove the occlusion structure that makes MPIs useful. The proposed method therefore uses $\hat{\delta}$ only as a local target and controls how strongly each individual layer is allowed to move toward it.

3.3 Alpha-Weighted Layer Deformation

Each layer starts at its original disparity δ_i and receives a bounded residual displacement toward the expected visible-scene disparity $\hat{\delta}(u)$. The idea behind this displacement is to let high-confidence regions of a layer better align with the estimated local scene surface, while preventing low-confidence or occluded regions from collapsing into the same depth surface. The residual is controlled by a confidence term that measures how strongly the layer owns a pixel. The ownership term $c_i(u)$ is based on the alpha-compositing weight rather than raw alpha alone:

$$c_i(u) = w_i(u)^{\gamma}, \tag{6}$$

where γ is the alpha-power parameter. Larger values of γ make the deformation more selective, so that only high-confidence regions are moved significantly. This is useful because low-alpha regions often represent uncertain or occluded content and should not be pulled aggressively toward the visible surface.

A depth closeness term is applied to reduce deformation for layers whose nominal disparity is far from the expected scene disparity. Let s_i denote the local spacing between neighboring MPI disparities. For interior layers this spacing is the average of the neighboring disparity gaps, while the first and last layers use the nearest available gap. The depth-closeness factor is

$$g_i(u) = \exp\left(-\frac{|\hat{\delta}(u) - \delta_i|}{\sigma s_i}\right), \tag{7}$$

where σ controls the falloff width in units of plane spacing. The final deformation weight combines the layer ownership term with the depth closeness term:

$$q_i(u) = c_i(u) g_i(u). \tag{8}$$

This prevents distant background layers from being pulled toward near foreground surfaces merely because they contain non-zero opacity.

The raw residual displacement is

$$\Delta_i(u) = q_i(u)\big(\hat{\delta}(u) - \delta_i\big). \tag{9}$$

The raw residual can be large when the expected disparity is far from the original layer disparity. Applying such a displacement directly could move a layer far outside the depth

interval it is supposed to represent, producing abrupt deformations and making the fixed back-to-front MPI compositing order inconsistent with the deformed geometry. The residual is therefore clipped to a maximum number of local plane spacings:

$$\Delta_i'(u) = \mathrm{clip}\left(\Delta_i(u), -Js_i, Js_i\right), \tag{10}$$

where J is the maximum plane-jump parameter. The deformed disparity before ordering constraints is therefore

$$\tilde{\delta}_i(u) = \delta_i + \Delta_i'(u). \tag{11}$$

Finally, layer ordering is enforced by clamping each deformed disparity to the disparity cell around its original plane. The boundary between two adjacent layers is placed at the midpoint between their original disparities. Let ℓ_i and h_i denote the lower and upper boundaries of this cell. The final deformed disparity is then

$$\delta_i'(u) = \mathrm{clip}\left(\tilde{\delta}_i(u), \ell_i, h_i\right). \tag{12}$$

This constraint prevents neighboring layers from crossing, which would make the back-to-front compositing order inconsistent with the geometry. The final deformed depth is obtained by converting back from disparity,

$$d_i'(u) = \frac{1}{\max(\delta_i'(u), \epsilon)}, \tag{13}$$

where ϵ is a small numerical constant.

3.4 Triangular Mesh Layer Construction

After the per-layer depth fields have been computed, each layer is converted into a regular triangular mesh. A mesh resolution $H_m \times W_m$ is selected independently of the MPI image resolution. If necessary, the deformed depth maps and RGBA textures are resized to this mesh resolution using bilinear interpolation. Each grid location becomes one mesh vertex. Let $\tilde{\mathbf{r}}(u)$ denote the unnormalized camera-space back-projection direction for image-grid location u, computed from the source intrinsics with its third component fixed to one. The corresponding source-camera-space vertex is

$$\mathbf{X}_i(u) = d_i'(u)\tilde{\mathbf{r}}(u). \tag{14}$$

Because $\tilde{\mathbf{r}}(u)$ is not normalized, $d_i'(u)$ is interpreted as camera z-depth rather than Euclidean distance along a unit ray. Thus, a constant depth field still produces a fronto-parallel MPI plane. Adjacent grid vertices form quadrilateral cells, and each cell is split into two triangles, producing the mesh primitives used for rasterization. The same triangle connectivity is used for every MPI layer. The RGB and alpha values of the original MPI layer are attached to the mesh vertices and are interpolated during rasterization. In this way, the proposed representation changes only the geometry of the MPI layers; it does not alter the predicted colors or alpha values.

3.5 Mesh Rendering and Compositing

To synthesize a target view, each mesh vertex is transformed from the reference camera to the target camera and projected using the target intrinsics. Each layer is rasterized as an RGBA triangle mesh. The layers are then composited from back to front using the standard over operation, matching the visibility model of the original MPI. The implementation renders layers independently rather than relying on a single global depth buffer over all layers. This is important because the MPI representation intentionally stores multiple surfaces along the same reference-camera ray, including background content behind foreground objects. A global depth-buffered render would keep only the nearest surface and would therefore discard part of the layered representation.

The independent layer rendering also ensures that the final image is produced using the intended depth-ordered MPI compositing rule. Although the deformable layers may move locally in depth, they are still rendered as MPI layers rather than as one unordered mesh scene. The layer-ordering constraint introduced above keeps each deformed layer within its original disparity interval, so the back-to-front compositing order remains consistent with the geometry. This does not fully eliminate all possible ordering artifacts: under large vertex displacements or coarse mesh resolutions, triangles within the same layer may still self-occlude after projection. In practice, this risk is reduced by bounding the displacement magnitude and by using the same fixed mesh connectivity for each layer.

3.6 Implementation Details

The implementation is written in Python and builds on the single-view MPI pipeline of Tucker and Snavely [6]. The pretrained MPI prediction model is used only to generate the input RGBA layers and plane depths from a single RGB image. The proposed deformation and rendering stages are implemented separately from this prediction model, so the MPI network itself is not retrained.

The deformation stage uses NumPy and PyTorch tensor operations to compute expected disparities, layer ownership weights, bounded residual displacements, and final deformed depth fields. Mesh construction and rendering are implemented in PyTorch using `nvdiffrast`. The renderer transforms mesh vertices using the source and target camera poses, projects them to the target camera, rasterizes the resulting triangles, and alpha-composites the rendered layers in MPI depth order.

The values of method parameters such as the ownership exponent, depth-closeness width, maximum plane jump, and mesh resolution are selected experimentally and reported in Section 4.

4 Experiments and Results

This section evaluates the proposed deformable MPI representation through experiments on RealEstate10K and LLFF, runtime measurements, and a controlled synthetic scene. The evaluation first tests whether alpha-weighted layer deformation improves single-view view synthesis quality when applied to predicted MPIs. The section then reports the computational cost of rendering deformable mesh layers compared with standard flat MPI planes. Finally, a synthetic Blender scene is used to examine the method under cleaner geometric conditions, where the MPI is constructed from rendered RGB-D data rather than predicted from a single image.

Table 1: RealEstate10K parameter-search results across different source-target strides.

Stride	Pairs	J	γ	σ	Flat PSNR	ΔPSNR ↑	ΔSSIM ↑	ΔLPIPS ↑
1	132	0.5	8.0	0.75	24.0656	-0.0001	-0.000007	-0.000001
5	652	0.5	8.0	0.5	20.7316	-0.0003	-0.000023	0.000002
10	132	0.5	8.0	0.5	18.5955	-0.0004	-0.000037	-0.000008
15	128	1.5	8.0	0.5	17.3916	-0.0004	-0.000049	0.000010

Table 2: LLFF parameter-search results across different source-target strides.

Stride	Pairs	J	γ	σ	Flat PSNR	ΔPSNR ↑	ΔSSIM ↑	ΔLPIPS ↑
1	128	0.5	0.5	5.0	11.5572	0.0098	0.001306	0.006047
8	124	16.0	1.5	3.0	11.5710	0.0003	0.000009	0.000529
12	115	16.0	0.5	3.0	11.7206	0.0093	0.000279	0.003549
18	97	0.5	0.75	2.0	11.7163	0.0007	0.000023	0.000674

4.1 Datasets and Pair Generation

Experiments are performed on processed subsets of RealEstate10K [9] and LLFF [3]. RealEstate10K is used as the main benchmark because it matches the type of video-based camera motion used by the original single-view MPI method. LLFF is included as an additional real-world benchmark with more challenging camera motion and scene geometry.

Both datasets are converted into a common pair-based layout. Each sample consists of a source image, a target image, source and target camera intrinsics, and source and target camera poses. The source image is used to predict an MPI, while the target image is used only as the reference image for evaluation. Source-target pairs are generated using a fixed frame stride: each source image is paired with an image that appears a fixed number of steps later in the same sequence. This produces pairs with known relative camera motion while keeping the evaluation procedure simple and reproducible.

For RealEstate10K, extracted frames are resized to 512×512. For LLFF, images are resized to 512×384, matching the aspect ratio of the processed LLFF images. These resolutions are chosen as a compromise between evaluation quality and computational cost. They are large enough to preserve the main scene structure and make local deformation effects visible, while keeping storage requirements, MPI prediction time, and rendering time manageable. The actual number of evaluated pairs varies between experiments because unavailable videos, failed frame extractions, and different stride settings affect how many valid pairs can be formed.

4.2 Evaluation Metrics

The rendered target views are compared against the ground-truth target images using PSNR, SSIM [7], and LPIPS [8]. The first two are traditional image-similarity metrics: PSNR measures pixel-level reconstruction quality, while SSIM measures structural similarity. In contrast, LPIPS measures perceptual distance using deep image features and is designed to better correlate with human perceptual judgments than purely pixel-wise metrics. For the raw metrics, higher PSNR and SSIM values indicate better quality, whereas lower LPIPS values indicate better perceptual similarity. To make improvements comparable across metrics, the reported changes are signed so that positive values always indicate an improvement over the flat MPI baseline. In particular, ΔLPIPS is computed as flat LPIPS minus deformable LPIPS.

4.3 Evaluation Pipeline

For the RealEstate10K and LLFF evaluations, the evaluation pipeline is split into two workers to handle dependency incompatibilities between the original TensorFlow-based MPI prediction code and the PyTorch/`nvdiffrast`-based renderer. First, the TensorFlow worker runs the pretrained single-view MPI model and exports the predicted MPI layers for each source image. The PyTorch/`nvdiffrast` worker then loads the exported MPI, renders both the flat and deformable representations from the target camera pose, and computes the evaluation metrics against the target image.

For each source-target pair, the flat baseline and the deformable representation use the same predicted MPI. The flat baseline keeps the original fronto-parallel MPI planes, while the deformable variant applies the alpha-weighted mesh deformation described in Section 3. For image-quality evaluation, both variants are rendered through the same dense mesh renderer: in the flat baseline, every vertex of layer i is assigned the original constant depth d_i, while in the deformable variant each vertex uses the deformed depth $d'_i(u)$. By keeping the rendering procedure fixed, the comparison isolates the effect of the proposed deformation rather than differences between renderers. All GPU-based experiments are run under WSL on an NVIDIA RTX 3000 Ada Generation Laptop GPU.

4.4 Parameter Search

The method contains several parameters that control how strongly each layer is deformed. These include the maximum plane jump J, the ownership exponent γ, and the depth-closeness width σ. To select the deformation parameters, a grid search is performed over J, γ, and σ. The MPI for each source image is exported once and then reused while evaluating different parameter combinations. This avoids retraining or regenerating the MPI for each parameter setting.

Table 3: Quantitative results for representative RealEstate10K and LLFF evaluations.

Dataset	Pairs	Flat PSNR	Flat SSIM	Flat LPIPS	ΔPSNR ↑	ΔSSIM ↑	ΔLPIPS ↑
RealEstate10K	652	20.7316	0.7294	0.1244	-0.0003	-0.000023	0.000002
LLFF	464	11.6346	0.2027	0.6050	0.0102	0.000207	0.003995

Table 4: Runtime comparison between flat MPI rendering and deformable mesh rendering.

Representation	Rendering method	Average FPS ↑
Flat MPI	Textured quads	13.8
Deformable MPI	Dense mesh layers	5.9

Table 5: Controlled Blender scene results using RGB-D MPI construction.

Flat PSNR	ΔPSNR ↑	ΔSSIM ↑	ΔLPIPS ↑
19.1034	0.0969	0.01788	0.00661

The search evaluates 880 parameter combinations. The maximum plane jump is selected from $J \in \{0.5, 1, 1.5, 2, 3, 5, 8, 16, 24, 32\}$, the ownership exponent from $\gamma \in \{0.5, 0.75, 1, 1.5, 2, 3, 5, 8\}$, and the depth-closeness width from $\sigma \in \{0.5, 0.75, 1, 1.5, 2, 3, 5, 8, 16, 24, 32\}$. The ranges are chosen to cover conservative, moderate, and aggressive deformation settings while keeping the grid search computationally feasible. The mesh resolution could also affect the representation, so it is fixed to match the processed image resolution: 512×512 for RealEstate10K and 512×384 for LLFF. This gives the deformation enough spatial detail to affect visible image regions while avoiding a higher-resolution mesh that would increase rendering cost without adding information beyond the processed input images. Visibility weighting, depth-closeness weighting, and layer-order enforcement are enabled in all tested configurations.

For each parameter combination, the results are aggregated over all evaluated pairs. The primary selection criterion is the average rank over the improvements in PSNR, SSIM, and LPIPS. This rank-based selection avoids choosing a setting that improves only one metric while substantially degrading the others.

Tables 1 and 2 report the best rank-averaged parameter setting for each evaluated stride. On RealEstate10K, the selected settings lead to changes close to zero across all tested strides. LLFF shows slightly more variation, with some stride settings producing small positive changes across the metrics. However, the absolute improvements remain limited in both datasets, indicating that the current deformation strategy has only a marginal effect under the evaluated full-image metrics.

4.5 Quantitative Results

Table 3 reports two representative quantitative evaluations using the selected deformation parameters. For RealEstate10K, the stride-5 evaluation is used because it contains the largest number of evaluated pairs, while the stride-specific results in Table 1 show only negligible differences between strides. This evaluation uses 652 pairs, a processed resolution of 512×512, and the selected parameters $J = 0.5$, $\gamma = 8.0$, and $\sigma = 0.5$. For LLFF, the table reports the aggregate result over all evaluated pairs, as the stride-specific results show more variation. The parameters $J = 1.0$, $\gamma = 0.5$, and $\sigma = 3.0$ are used because they produced the best overall rank-averaged result across all evaluated LLFF pairs. This evaluation uses 464 pairs at a processed resolution of 512×384.

The RealEstate10K result shows almost no difference between the flat and deformable representations. PSNR and SSIM decrease slightly, while LPIPS improves by a negligible amount. The LLFF aggregate result shows small positive changes across all three metrics, with the largest relative change appearing in LPIPS. However, the absolute improvements remain small compared with the overall metric values. Together with the stride-specific parameter-search results, these results indicate that the proposed deformation has only a marginal quantitative effect under the evaluated full-image metrics.

4.6 Runtime Evaluation

Runtime is evaluated separately from image quality using an interactive `nvdiffrast`-based renderer. The deformable representation is rendered as dense triangular mesh layers, since its spatially varying depth requires per-vertex geometry. In contrast, the flat MPI baseline is rendered using a lightweight textured-plane renderer, where each MPI layer is represented as a single textured quad. This makes the runtime comparison reflect the practical cost of using deformable layers instead of standard flat MPI planes.

The measurement is performed on the original Tucker and Snavely input image at an output resolution of 1024×576. Both representations use 32 MPI layers. For the deformable representation, the mesh resolution is set to 512×512 vertices per layer, with deformation parameters $J = 0.5$ and $\gamma = 8.0$. The average frame rate is recorded over one minute of running each renderer.

Table 4 shows that the deformable representation is slower than the flat MPI baseline. The flat textured-plane renderer reaches 13.8 FPS, while the deformable mesh renderer reaches 5.9 FPS. This corresponds to approximately a $2.3\times$ reduction in frame rate. The decrease is expected because the deformable representation requires dense mesh rasterization, whereas the flat baseline can be rendered using one textured quad per MPI layer.

4.7 Controlled Synthetic Scene

To better understand the behaviour of the proposed deformation under cleaner geometric conditions, an additional con-

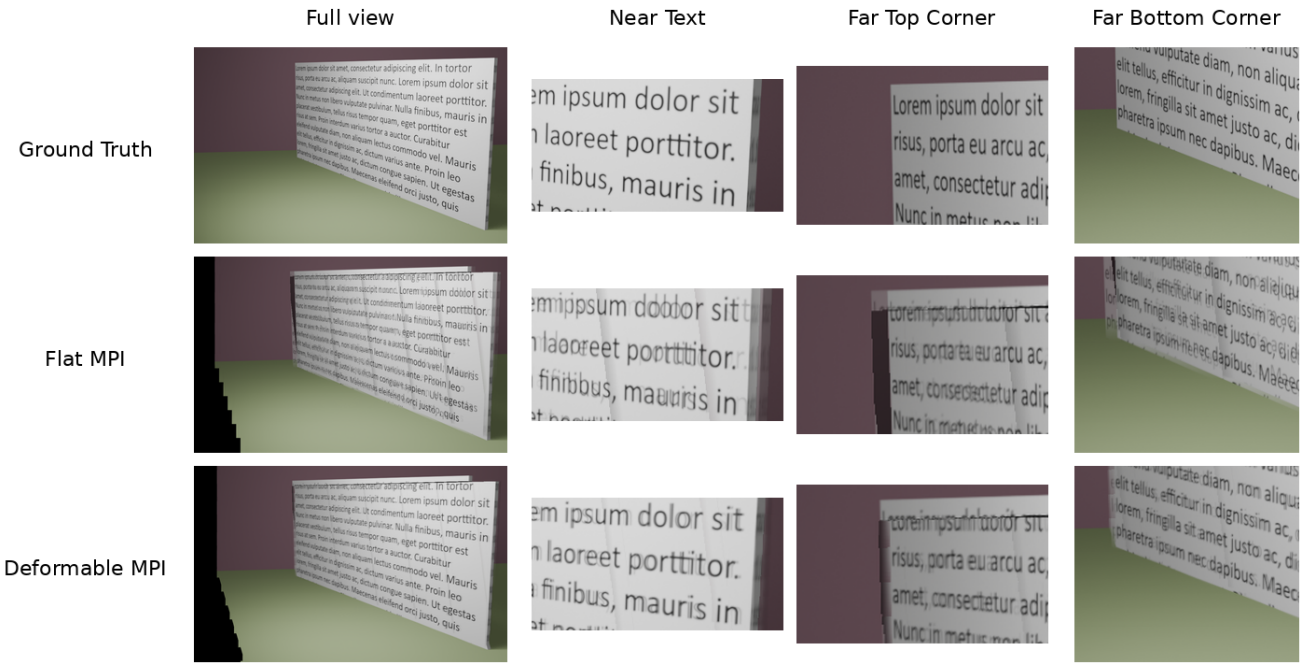

Figure 1: Qualitative comparison on the controlled Blender wall scene. Rows show the ground truth, flat MPI render, and deformable MPI render; crops highlight local differences around the text and wall boundaries. The shared top-edge displacement in both MPI renders comes from the RGB-D MPI construction rather than the proposed deformation.

trolled experiment is performed using a synthetic Blender scene. The scene contains a single slanted wall with a high-frequency text texture, rendered from a central source camera and laterally translated target cameras. Unlike the RealEstate10K and LLFF experiments, this experiment does not use the pretrained single-view MPI predictor because the simple synthetic scene is not representative of the natural-image data used by the model and produced unreliable MPI predictions. Instead, an oracle-style MPI is constructed from the rendered source RGB image and the Blender depth map using 8 inverse-depth planes at a resolution of 1024×640.The deformable variant uses $J = 1.0$, $\gamma = 0.01$, and $\sigma = 8.0$ to make the deformation more visible, allowing the experiment to isolate the effect of replacing flat planes with deformable layers under clean geometric conditions. The text texture is used because small geometric distortions are easier to observe when the surface contains readable structure.

Table 5 shows that, in this controlled setting, the deformable representation produces a meaningful improvement over the flat MPI baseline across all three metrics. The qualitative comparison in Figure 1 supports this result visually: the deformable render more closely matches the ground-truth target view in regions where the slanted surface causes visible differences between the two representations. Figure 2 further illustrates why this occurs, showing that the deformable layer bends toward the slanted wall geometry rather than remaining fronto-parallel. Importantly, this experiment is idealized because the MPI is constructed from accurate Blender depth rather than predicted from a single image.

4.8 Summary of Results

Overall, the learned-MPI experiments, runtime evaluation, and controlled Blender scene show that the deformation behaves as intended under clean geometric conditions, but does not provide a meaningful improvement as a post-processing step for predicted single-view MPIs.

5 Discussion

The results show that alpha-weighted deformation does not produce a meaningful improvement when applied as a post-processing step to predicted single-view MPIs. One explanation is that the effect of the deformation depends strongly on the quality of the input MPI. On RealEstate10K, the flat MPI baseline already performs relatively well, leaving limited room for a bounded geometric displacement to improve the result. On LLFF, the much lower baseline PSNR suggests that the predicted MPIs are less accurate for these more complex scenes. In this case, the deformation is limited because it can only move the existing MPI geometry; it cannot correct missing content, inaccurate colors, noisy alpha layers, or incorrect depth structure. Moreover, the expected disparity and ownership weights are computed from the predicted MPI itself, so the method does not introduce an independent depth or visibility estimate that could reliably correct these errors. This also helps explain why the controlled RGB-D experiment behaves differently: its input MPI is built from accurate depth, giving the deformation a cleaner geometric signal than the learned single-view predictions.

The number of MPI layers may also affect how visible the

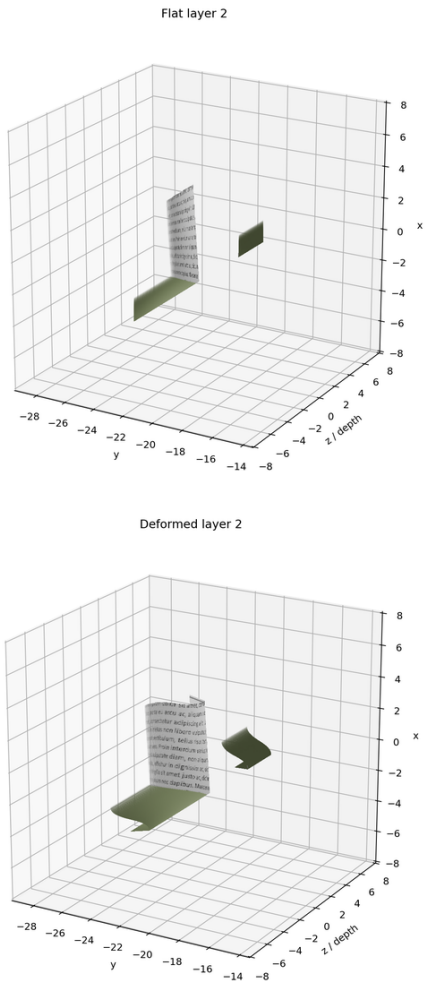

Figure 2: 3D geometry comparison for layer 2 in the controlled Blender scene. The flat MPI layer remains fronto-parallel, whereas the deformable layer bends to better follow the slanted wall geometry.

deformation becomes. The pretrained model produces 32 layers, which gives the flat MPI a relatively dense set of depth planes. As a result, the baseline can already approximate many depth changes through layer selection alone, reducing the additional benefit that vertex displacement can provide. A representation with fewer, more widely spaced layers might show a larger benefit from deformation, but testing this would require retraining the pretrained MPI prediction model. The controlled Blender experiment supports this interpretation: when the MPI is constructed from accurate RGB-D data and uses only 8 layers, the deformable representation can better follow the slanted wall geometry and produces a meaningful improvement over the flat baseline. However, this setting is idealized because it removes the single-view MPI prediction problem and therefore does not represent the full task addressed by the main experiments.

The full-image evaluation metrics may also understate small local effects of the deformation. Improvements around slanted surfaces, object boundaries, or disoccluded regions can be diluted by larger image regions that remain unchanged. However, the runtime results show that deformable layers introduce a clear computational cost, reducing the average frame rate to less than half of the flat MPI baseline. In its current form, the method therefore adds rendering complexity without producing a corresponding improvement on the main learned-MPI benchmarks.

Future work could integrate deformable geometry more directly into the MPI prediction process, allowing the network to learn layer geometry and appearance jointly rather than deforming a fixed predicted MPI afterward. It would also be useful to evaluate models trained with different numbers of MPI layers, since fewer layers may benefit more from geometric deformation. Finally, a more detailed ablation study could isolate the effects of visibility weighting, depth-closeness weighting, layer-order enforcement, and mesh resolution.

6 Responsible Research

This project uses existing datasets, a pretrained MPI prediction model, and a synthetic scene created specifically for controlled evaluation. The RealEstate10K and LLFF datasets are used only for research evaluation, and the paper reports aggregate quantitative results and selected qualitative examples rather than redistributing the original datasets. The experiments use processed subsets of the datasets, and no new real-world image data is collected for this project. This reduces additional privacy risks and avoids introducing new data-collection concerns. The controlled Blender scene is synthetic and therefore does not contain real-world personal or private content. The use of external datasets and pretrained models follows their released research-use conditions, and all external work is cited where it is used.

Reproducibility is an important concern for this project because the evaluation depends on multiple software environments and external data sources. The MPI prediction stage uses the original TensorFlow-based single-view MPI implementation, while the deformation and rendering stages use a separate PyTorch and `nvdiffrast`-based pipeline. This split is necessary because of dependency incompatibilities, but it also makes the experimental setup more complex to reproduce. To make the work more transparent, the paper reports the tested parameter ranges, selected parameter values, image resolutions, mesh resolutions, dataset strides, number of evaluated pairs, hardware, and rendering setup. The MPI for each source image is exported once and reused across parameter settings, which also makes the parameter search more consistent.

There are still reproducibility limitations. RealEstate10K is derived from online videos, so some videos may become unavailable or fail during processing. As a result, another run of the preprocessing pipeline may not produce exactly the same set of valid source-target pairs. GPU rendering can also introduce small numerical differences across hardware or driver versions. These issues do not invalidate the results, but they mean that exact numerical reproduction may require using the same processed pair lists, exported MPI files, parameter settings, and rendering environment.

A second responsible-research concern is the interpretation of novel-view synthesis outputs. Single-view view synthesis creates images of viewpoints that were not directly observed. As a result, the synthesized view may contain plausible but incorrect geometry, texture, or disoccluded background content. Such hallucinated details can appear visually convincing even when they do not correspond to the true scene. This is especially important for single-view MPI prediction, where the model must infer hidden content from only one input image.

For this reason, generated novel views should not be treated as reliable evidence of the true scene structure without additional validation. This is particularly relevant in applications such as inspection, mapping, reconstruction, or safety-critical visualization, where an incorrect synthesized detail could affect later decisions. The evaluation in this project therefore compares synthesized views against ground-truth target images using PSNR, SSIM, and LPIPS, and qualitative results are interpreted cautiously. The controlled Blender experiment is also explicitly described as idealized because it uses accurate Blender depth rather than a predicted single-view MPI. For this reason, the controlled synthetic experiment is interpreted separately from the learned-MPI evaluations. It demonstrates the behaviour of the deformation under clean geometric conditions, but does not by itself establish that the same improvement transfers to predicted single-view MPIs.

Artificial intelligence tools were used during this project as a supporting aid, but not as a replacement for the research, implementation, or interpretation work. AI assistance was used to help write and debug parts of the project code, improve grammar and spelling, and check topic-specific terminology related to MPI-based view synthesis, mesh deformation, and evaluation metrics. All AI-generated suggestions were reviewed critically before use, and code suggestions were tested, adapted, and integrated manually into the project pipeline. The experimental design, parameter choices, evaluation procedure, interpretation of results, and final conclusions remained the responsibility of the author and were based on the reported experiments rather than on AI-generated claims. AI tools were also not treated as scientific sources: technical claims and related-work statements were checked against the cited literature. This ensured that the final paper accurately reflects the work that was implemented and evaluated, while maintaining responsibility for the correctness, integrity, and transparency of the research.

7 Conclusion

This project investigated whether MPI-based single-view view synthesis can be improved by replacing fronto-parallel MPI planes with alpha-weighted deformable mesh layers. The proposed method keeps the color, opacity, and ordering structure of a predicted MPI, but converts each layer into a triangular mesh whose vertices are displaced using expected disparity and layer ownership weights. The goal is to add local geometric flexibility while preserving the layered visibility structure of MPIs.

The experiments show that, as a post-processing step for predicted single-view MPIs, the proposed deformation does not produce a meaningful improvement on the tested RealEstate10K and LLFF evaluations. RealEstate10K remains effectively unchanged, while LLFF shows only small positive changes under full-image PSNR, SSIM, and LPIPS. The method also introduces a clear runtime cost, indicating that the added rendering complexity is not justified by the measured improvements on the main learned-MPI benchmarks.

The controlled Blender experiment provides a more positive result: when the MPI is constructed from accurate RGB-D data and uses fewer layers, the deformable representation better follows the slanted wall geometry and improves over the flat baseline. This suggests that deformable MPI layers can be useful when the input layered geometry is reliable, but that applying deformation after a fixed learned MPI prediction is not sufficient to substantially improve single-view synthesis quality. Future work should therefore integrate deformable geometry more directly into the MPI prediction process, so that geometry and appearance can be learned jointly.

References

[1] Michael Broxton, John Flynn, Ryan Overbeck, Daniel Erickson, Peter Hedman, Matthew DuVall, Jason Dourgarian, Jay Busch, Matt Whalen, and Paul Debevec. Immersive light field video with a layered mesh representation. *ACM Transactions on Graphics*, 39(4), 2020.

[2] Ronghang Hu, Nikhila Ravi, Alexander C. Berg, and Deepak Pathak. Worldsheet: Wrapping the world in a 3d sheet for view synthesis from a single image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12528–12537, 2021.

[3] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Amit Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics*, 38(4), 2019.

[4] Jonathan Shade, Steven Gortler, Li-wei He, and Richard Szeliski. Layered depth images. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, pages 231–242. ACM, 1998.

[5] Meng-Li Shih, Shih-Yang Su, Johannes Kopf, and Jia-Bin Huang. 3d photography using context-aware layered depth inpainting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8028–8038, 2020.

[6] Richard Tucker and Noah Snavely. Single-view view synthesis with multiplane images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 551–560, 2020.

[7] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[8] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness

of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 586–595, 2018.

[9] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *ACM Transactions on Graphics*, 37(4):65:1–65:12, 2018.