Supplemental Material for Depth from Gradients in Dense Light Fields for Object Reconstruction

Kaan Yücer[1,2] Changil Kim[1] Alexander Sorkine-Hornung[2] Olga Sorkine-Hornung[1]

[1]ETH Zurich [2]Disney Research

1. Depth from gradient

In Section 3.1 of the paper, we discussed how depth can be computed using light field gradients. Here, we will describe the gradient computation on the light field patch $s_{i,j}(\mathbf{p},\mathbf{q})$ in more detail. Given a 2×5 patch, our aim is to compute the trajectory direction around $\mathbf{p}$, which is perpendicular to the gradient direction around the same pixel. Since the patch size is 2×5, the gradient computation is not well-defined, and using forward differences will lead to a 0.5 pixel shift of the computed values. However, this problem can easily be solved by employing a 5×5 patch, interpolated from $s_{i,j}(\mathbf{p},\mathbf{q})$: The first and the last rows of the new patch $s^*_{i,j}(\mathbf{p},\mathbf{q})$ are taken from the original patch, whereas the central 3 patches are interpolated linearly.

$$s^*_{i,j}(\mathbf{p},\mathbf{q}) = \begin{pmatrix} \mathbf{a} \\ 3/4\mathbf{a}+1/4\mathbf{b} \\ 1/2\mathbf{a}+1/2\mathbf{b} \\ 1/4\mathbf{a}+3/4\mathbf{b} \\ \mathbf{b} \end{pmatrix} \tag{1}$$

where $\mathbf{a}$ and $\mathbf{b}$ are the two rows of $s_{i,j}(\mathbf{p},\mathbf{q})$. Given this new patch, we apply a Sobel filter on $s^*_{i,j}(\mathbf{p},\mathbf{q})$ and compute the gradients $\nabla_x s^*_{i,j}(\mathbf{p},\mathbf{q})$ and $\nabla_y s^*_{i,j}(\mathbf{p},\mathbf{q})$. The linear interpolation between $\mathbf{a}$ and $\mathbf{b}$ keeps the relationship between the pixels in the x dimension the same, which is reflected in the computed gradients:

$$\nabla_x s_{i,j}(\mathbf{p},\mathbf{q}) = \nabla_x s^*_{i,j}(\mathbf{p},\mathbf{q}). \tag{2}$$

In the y dimension, we stretch the original patch by a factor of 4, such that the distance betwwen $\mathbf{p}$ and $\mathbf{q}$ increases from a single pixel to 4 pixels. This means that the gradients are also related by the same factor:

$$\nabla_y s_{i,j}(\mathbf{p},\mathbf{q}) = 4\cdot\nabla_y s^*_{i,j}(\mathbf{p},\mathbf{q}). \tag{3}$$

Note that we do not explicitly compute $s^*_{i,j}(\mathbf{p},\mathbf{q})$ or its gradients, but pre-compute the factors for computing the gradients of $s_{i,j}(\mathbf{p},\mathbf{q})$ directly.

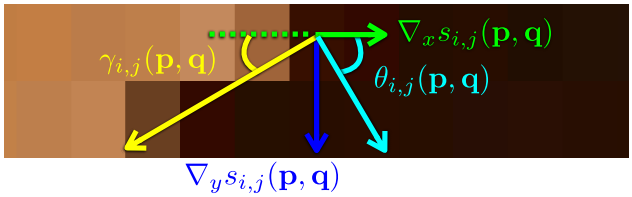

Figure 1: Given the light field patch $s_{i,j}(\mathbf{p},\mathbf{q})$, and the gradients $\nabla_x s_{i,j}(\mathbf{p},\mathbf{q})$ in green and $\nabla_y s_{i,j}(\mathbf{p},\mathbf{q})$ in blue, the gradient direction $\theta_{i,j}(\mathbf{p},\mathbf{q})$ and the trajectory direction $\gamma_{i,j}(\mathbf{p},\mathbf{q})$ can be computed using simple trigonometry. The gradient and trajectory directions are shown in cyan and yellow, respectively.

Given the two gradients, the gradient direction in the light field patch $s_{i,j}(\mathbf{p},\mathbf{q})$ is computed as:

$$\theta_{i,j}(\mathbf{p},\mathbf{q}) = \tan^{-1}(\nabla_y s_{i,j}(\mathbf{p},\mathbf{q})/\nabla_x s_{i,j}(\mathbf{p},\mathbf{q})). \tag{4}$$

Since the direction of the trajectory γ is perpendicuar to the gradient direction θ (change in color is minimal in the trajectory direction), we can compute it as:

$$\gamma_{i,j}(\mathbf{p},\mathbf{q}) = \tan^{-1}(-\nabla_x s_{i,j}(\mathbf{p},\mathbf{q})/\nabla_y s_{i,j}(\mathbf{p},\mathbf{q})). \tag{5}$$

See Figure 1 for a visualization of the gradient and trajectory directions.

Now that we know the gradient and trajectory directions, we can compute where $\mathbf{p}$ maps to in the second row of $s_{i,j}(\mathbf{p},\mathbf{q})$. Between the two rows, the motion along the y direction equals 1. Given that the light field patch is centered around $\mathbf{p}$ and $\mathbf{q}$, meaning that they have x coordinates 0, the motion along the x direction should equal p^s_j, *i.e.* the x coordinate of the point, where $\mathbf{p}$ maps to in the second row of the patch. The trajectory direction $\gamma_{i,j}(\mathbf{p},\mathbf{q})$ should remain the same, meaning:

$$\gamma_{i,j}(\mathbf{p},\mathbf{q}) = \tan^{-1}(1/p^s_j). \tag{6}$$

We can compute p^s_j simply by:

$$p^s_j = 1/\tan(\gamma_{i,j}(\mathbf{p},\mathbf{q})). \tag{7}$$

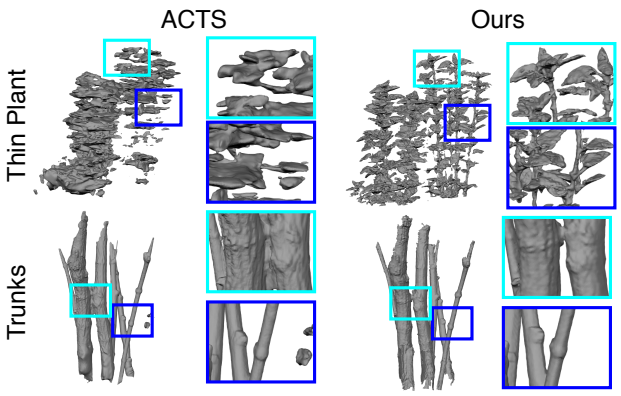

Figure 2: Comparison of our technique to the ACTS software [2], with close-ups on the reconstructed meshes. See Section 2 for a detailed discussion.

2. Comparison to ACTS

Here, we elaborate more on the comparison to the ACTS software [2], this time meshing the point clouds of ACTS. Since this software produces per-view depth maps without normal information, we first compute per-view normals using PCA over small patches centered at every pixel of the depth maps. We then merge the depth maps into a global point cloud, use a bounding box to filter only the foregroud points, and mesh them using Poisson surface reconstruction [1], see Figure 2.

Our reconstruction results are more faithful to the object, and can generate more details, especially in the THIN PLANT dataset, where most of the leaves are either removed or merged in the reconstructions of ACTS. Since the point clouds of ACTS can be noisy (see paper, Figure 6), the surface reconstruction step cannot generate the fine details. As for the TRUNKS dataset, ACTS combined with Poisson reconstruction generates similar results to ours. However, it also merges two tree trunks, and generates extra floating regions, due to the consistent noise in the point clouds (see paper, Figure 6).

References

[1] M. Kazhdan and H. Hoppe. Screened poisson surface reconstruction. *ACM Trans. Graph.*, 32(3), 2013. 2

[2] G. Zhang, J. Jia, T. Wong, and H. Bao. Consistent depth maps recovery from a video sequence. *PAMI*, 31(6), 2009. 2