GeoDiff: Geometry-Guided Diffusion for Metric Depth Estimation

Supplementary Material

In this supplementary material, we first provide additional derivations and insights in Section 7. We then present our method through pseudocode in Section 8, followed by implementation details in Section 9. Dataset specifications are described in Section 10, while limitations and future work are discussed in Section 11. Finally, additional experimental results are presented in Section 12.

7. Detailed method

7.1. Conditional Score Derivation

In this section, we provide the complete derivation of our conditional score. Applying Bayes' theorem, the score function of the conditional distribution can be expressed as:

$$\nabla_{\boldsymbol{z}_t} \log p(\boldsymbol{z}_t \mid \boldsymbol{y}_1, \boldsymbol{y}_2) = \nabla_{\boldsymbol{z}_t} \log p(\boldsymbol{z}_t \mid \boldsymbol{y}_1) + \nabla_{\boldsymbol{z}_t} \log p(\boldsymbol{y}_2 \mid \boldsymbol{z}_t, \boldsymbol{y}_1) \quad (18)$$

Under mild assumptions [7], and a decoder $\boldsymbol{D}$ that maps the latent back to the image space, we can approximate this score function using:

$$\begin{aligned} \nabla_{\boldsymbol{z}_t} \log p(\boldsymbol{y}_2 \mid \boldsymbol{z}_t, \boldsymbol{y}_1) &\approx \nabla_{\boldsymbol{z}_t} \log p(\boldsymbol{y}_2 \mid \hat{\boldsymbol{z}}_0, \boldsymbol{y}_1) \\ &\approx \nabla_{\boldsymbol{z}_t} \log p(\boldsymbol{y}_2 \mid \boldsymbol{D}(\hat{\boldsymbol{z}}_0), \boldsymbol{y}_1) \end{aligned} \quad (19)$$

where $\hat{\boldsymbol{z}}_0$ is estimated using Tweedie's formula [37]:

$$\hat{\boldsymbol{z}}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}} \left(\boldsymbol{z}_t + \sqrt{1-\bar{\alpha}_t} \boldsymbol{s}_\theta \left(\boldsymbol{z}_t, t, \boldsymbol{y}_1 \right) \right) \quad (20)$$

Leveraging the Gaussian noise model assumption in Equation 11, we get:

$$\nabla_{\boldsymbol{z}_t} \log p \left(\boldsymbol{y}_2 \mid \boldsymbol{z}_t, \boldsymbol{y}_1 \right) \simeq -\frac{1}{\sigma^2} \nabla_{\boldsymbol{z}_t} \left\| \boldsymbol{y}_2 - \boldsymbol{P}_{\boldsymbol{y}_1 \to \boldsymbol{y}_2} \left(\tilde{\boldsymbol{x}}_0, \boldsymbol{y}_1 \right) \right\|_2^2 \quad (21)$$

in which $\tilde{\boldsymbol{x}}_0$ is the metric depth calculated following the Equation 13.

Thus, the final conditional score function is:

$$\nabla_{\boldsymbol{z}_t} \log p \left(\boldsymbol{z}_t \mid \boldsymbol{y}_1, \boldsymbol{y}_2 \right) \approx \nabla_{\boldsymbol{z}_t} \log p(\boldsymbol{z}_t \mid \boldsymbol{y}_1) - \lambda \nabla_{\boldsymbol{z}_t} \| \boldsymbol{y}_2 - \boldsymbol{P}_{\boldsymbol{y}_1 \to \boldsymbol{y}_2} (\tilde{\boldsymbol{x}}_0, \boldsymbol{y}_1) \|_2^2 \quad (22)$$

7.2. Differentiable warping

As discussed in Section 3.3 and Section 4.2, we use differentiable warping to render novel view given predicted depth. Then, our method leverages given RGB input image to calculate photometric loss as guidance (see Equation 14) for diffusion process. There are two design choices for warping operator, which are forward warping operator $\boldsymbol{P}_{\boldsymbol{y}_1 \to \boldsymbol{y}_2} (\boldsymbol{x}_1, \boldsymbol{y}_1)$, which projects $\boldsymbol{y}_1$ onto $\boldsymbol{y}_2$; and the backward warping operator $\boldsymbol{P}_{\boldsymbol{y}_2 \to \boldsymbol{y}_1} (\boldsymbol{x}_1, \boldsymbol{y}_2)$. If one opts to use forward warping as renderer, $\mathcal{L}_{geo}$ in Equation 14 has the following form:

$$\begin{aligned} \mathcal{L}_{geo}^{forward} &= \eta(1 - \text{SSIM} \left(\boldsymbol{y}_2, \boldsymbol{P}_{\boldsymbol{y}_1 \to \boldsymbol{y}_2} \left(\tilde{\boldsymbol{x}}_0, \boldsymbol{y}_1 \right) \right))/2 \\ &+ (1-\eta) \| \boldsymbol{y}_2 - \boldsymbol{P}_{\boldsymbol{y}_1 \to \boldsymbol{y}_2} \left(\tilde{\boldsymbol{x}}_0, \boldsymbol{y}_1 \right) \|_1 \end{aligned} \quad (23)$$

otherwise, the backward warping could also be used with the form:

$$\begin{aligned} \mathcal{L}_{geo}^{backward} &= \eta(1 - \text{SSIM} \left(\boldsymbol{y}_1, \boldsymbol{P}_{\boldsymbol{y}_2 \to \boldsymbol{y}_1} \left(\tilde{\boldsymbol{x}}_0, \boldsymbol{y}_2 \right) \right))/2 \\ &+ (1-\eta) \| \boldsymbol{y}_1 - \boldsymbol{P}_{\boldsymbol{y}_2 \to \boldsymbol{y}_1} \left(\tilde{\boldsymbol{x}}_0, \boldsymbol{y}_2 \right) \|_1 \end{aligned} \quad (24)$$

Now, we discuss the design choice of the two options. Given a source image ($\boldsymbol{y}_1$), target image ($\boldsymbol{y}_2$), intrinsic camera matrices K_1, K_2, and camera-to-world extrinsic matrices E_1, E_2 for the source and target views respectively, we establish a generalized framework for our method. We explicitly represent both camera extrinsics to handle arbitrary camera configurations rather than just using a single relative transformation $T_{\mathbf{1} \to \mathbf{2}}$ as in Equation 9. In the specific case of calibrated stereo pairs captured simultaneously by a binocular rig, the transformation simplifies to a pure translation. However, for arbitrarily captured image pairs, the complete extrinsic matrices are necessary to accurately transform points between the two coordinate systems.

Forward warping maps pixels from a source image ($\boldsymbol{y}_1$) to positions in target image ($\boldsymbol{y}_2$). The target coordinate is formulated as:

$$\boldsymbol{c}_2 \sim K_2 E_2^{-1} E_1 \boldsymbol{x}_1(\boldsymbol{c}_1) K_1^{-1} \boldsymbol{c}_1 \quad (25)$$

where $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ denote the homogeneous pixel coordinates in $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$, respectively, and $\boldsymbol{x}_1(\boldsymbol{c}_1)$ represents the depth at pixel $\boldsymbol{c}_1$ in $\boldsymbol{y}_1$. After getting the corresponding pixel coordinates, we can "splat" each source pixel to its corresponding location in target view. However, there are a few implementation challenges. A fundamental issue is that some target pixels might not receive any values, creating holes in the warped image. These voids occur due to disocclusions (regions visible in the target view but occluded in the source view) and sampling disparities (discrete source pixels mapping to non-integer target coordinates with gaps between them). Addressing these holes requires complex post-processing techniques such as depth-aware inpainting or multi-scale filtering. The non-integer mapping of source

pixels to target coordinates further introduces discretization errors and potential aliasing, requiring appropriate interpolation strategies. Depth map inaccuracies are particularly problematic at discontinuities, where slight errors can significantly distort the warped result, making it even more difficult to apply to our framework. From a computational perspective, the unpredictable memory access patterns inherent in forward warping present optimization difficulties, particularly for parallel processing implementations.

Backward warping pulls back pixels from target image ($\boldsymbol{y}_2$) to source image ($\boldsymbol{y}_1$). It is worth noting that our backward warping is different from previous works [15, 25], where they perform backward warping from source to target given target depth. One the other hand, we warp from target back to source using source depth. Specifically, our backward warping is formulated as:

$$\boldsymbol{c}_2 \sim K_2 E_2 E_1^{-1} \boldsymbol{x}_1(\boldsymbol{c}_1) K_1^{-1} \boldsymbol{c}_1 \tag{26}$$

After computing the corresponding pixel coordinates, we sample pixel colors from the target image at these new coordinates. Since these coordinates are generally non-integer, we employ bilinear interpolation for color sampling. This approach inherently avoids the hole artifacts characteristic of forward warping methods. For this reason, we adopt backward warping as our rendering technique throughout this work.

Discussion. We explored several alternative techniques that ultimately proved suboptimal. Initial experiments with point cloud rasterization from Pytorch3D [35] revealed high sensitivity to point diameter and opacity parameters, resulting in rendering artifacts including holes and visible disk-like structures. Similarly, we attempted to initialize the point cloud as 3D Gaussians (3DGS) to leverage recent differentiable Gaussian rasterization techniques [24]. However, the 3DGS renderer introduces an excessive number of parameters to optimize, which proved inefficient during the limited sampling steps of our diffusion process.

Left-right consistency. While left-right consistency checks are commonly employed in stereo methods [15], we deliberately omit this approach in our framework. Unlike learning-based methods that can infer depth of both left and right from a single image, our optimization-based technique would require running the depth prediction process twice—once for each view—effectively doubling the computational cost. Therefore, in this work, we demonstrate our method's efficacy by optimizing the photometric loss using only a single reference view, achieving a favorable balance between accuracy and computational efficiency.

7.3. Regularization

As described in Section 4, we stabilize the optimization process by introducing a global scale g_s and applying L_2 regularization to the scale $\hat{s}$ and shift $\hat{t}$ parameters. To illustrate the design of these parameters, we present a toy example. Given a predicted relative depth map $\tilde{\boldsymbol{x}}_0^{rel}$ (normalized to the range $[0, 1]$), we incrementally increase the global scale g_s to compute the scaled depth $\tilde{\boldsymbol{x}}_0^{scale}$ such that $\tilde{\boldsymbol{x}}_0^{scale} = g_s \cdot \tilde{\boldsymbol{x}}_0^{rel}$. For each scale, we evaluate the Absolute Relative (AbsRel) metric (where lower values are better) using the scaled depth $\tilde{\boldsymbol{x}}_0^{scale}$ and the ground truth depth. Additionally, using the left image, right image, and the predicted scaled depth $\tilde{\boldsymbol{x}}_0^{scale}$, we compute the reprojection loss $\mathcal{L}_{geo}$, as defined in Equation 14, for each scale.

As shown in Figure 10, increasing the depth scale improves the resemblance of the re-rendered image to the left image. This occurs because closer depths result in larger disparities between the source and target viewpoints, requiring more significant transformations to align the images. Such large transformations often cause distortions, stretching, or undersampling in areas lacking sufficient source information, degrading the quality of the re-rendered image. In contrast, at greater depths, disparities between the viewpoints are smaller, leading to less dramatic transformations. These smaller adjustments maintain spatial coherence more effectively and reduce interpolation artifacts, producing sharper and more accurate re-rendered images.

Figures 7 and 8 demonstrate this pattern as the global scale g_s increases from 1 to 50. However, further increasing g_s, while improving the re-rendered image quality and enhancing $\mathcal{L}_{geo}$, leads to worse AbsRel metrics. This indicates that the depth scale $\tilde{\boldsymbol{x}}_0^{scale}$ deviates from the ground truth depth. This behavior underscores the strong geometric guidance provided by $\mathcal{L}_{geo}$ for the diffusion model during sampling.

To balance these considerations, we pre-select g_s based on the geometric loss. Specifically, we search for g_s within a predefined depth range and define the optimal scale as $g_s^* := \arg\min_{g_s} \mathcal{L}_{geo}$. Our approach can be viewed as a variant of the traditional Plane Sweep Volume technique [9], commonly used in stereo vision. Unlike conventional methods, our approach leverages the predicted relative depth to identify the correct depth scale, which is then applied uniformly to all pixels.

Finally, we apply L_2 regularization to the scale and shift parameters of $\tilde{\boldsymbol{x}}_0^{rel}$ to counteract the tendency of the optimization process to inflate these parameters, which can lead to incorrect metric depth predictions.

8. Algorithm

We provide detail pseudo algorithm for our method at Algorithm 1. To avoid confusion, note that while learnable scale

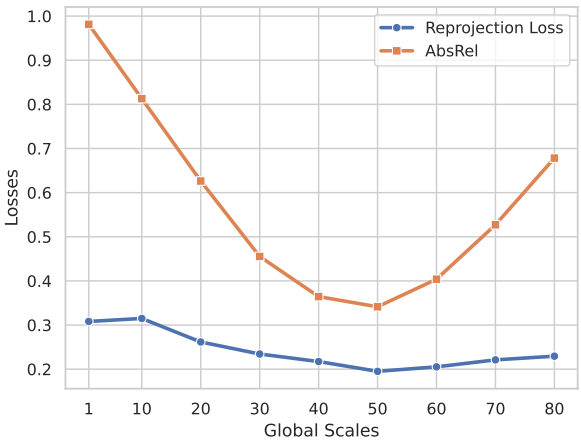

Figure 7. Global scale up to 80

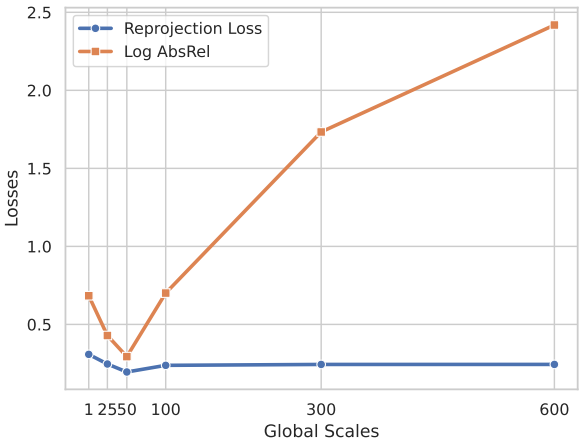

Figure 8. Global scale up to 600

Figure 9. We gradually increase the global scale g_s and observe a strong correlation between the reprojection loss and the AbsRel metric. For this example, the AbsRel reaches its minimum at a global scale of 50. However, beyond 50, the AbsRel significantly increases, while the reprojection loss shows little change, deviating from the pattern.

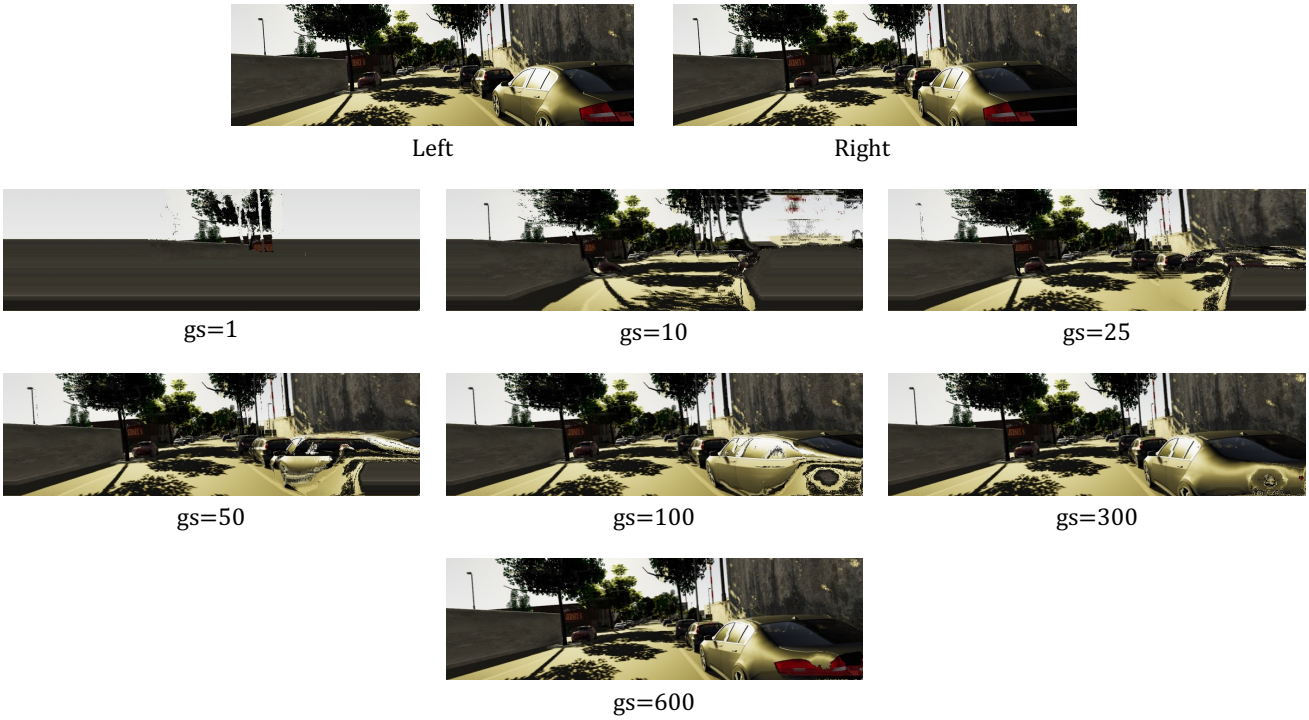

Figure 10. We gradually increase the global scale g_s and re-render the left image using the right image and $\tilde{\boldsymbol{x}}_0^{scale}$ (see Section 7.3). While greater depths result in higher-quality re-rendered images, this does not necessarily correspond to more accurate predicted depths.

and shift are denoted as $(\hat{s}, \hat{t})$, score function and time step of diffusion are denoted as $(\boldsymbol{s}, t)$, respectively.

9. Implementation details

Geometric optimization steps. Our method employs a test-time optimization approach. While multiple gradient updates could theoretically be performed during sampling

Algorithm 1 Geometric-Guided Diffusion for Metric Depth Estimation

Require: Stereo images $\boldsymbol{y}_1, \boldsymbol{y}_2$, camera intrinsics and extrinsics, pretrained diffusion model $\boldsymbol{s}_\theta(\boldsymbol{z}_t, t, \boldsymbol{y}_1)$
  Initialize learnable scale $\hat{s}$ and shift $\hat{t}$
  Initialize random noise $\boldsymbol{z}_T \sim \mathcal{N}(0, I)$.
  for $t = T-1$ to 0 **do**
    $\hat{\boldsymbol{s}}_{t+1} = \boldsymbol{s}_\theta(\boldsymbol{z}_t, t, \boldsymbol{y}_1)$ ▷ Compute the score
    $\hat{\boldsymbol{z}}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(\boldsymbol{z}_t + \sqrt{1-\bar{\alpha}_t}\boldsymbol{s}_\theta\left(\boldsymbol{z}_t, t, \boldsymbol{y}_1\right)\right)$ ▷ Compute relative depth using Tweedie's formula
    $\tilde{\boldsymbol{x}}_0 = \text{softplus}(\hat{s}) \cdot \boldsymbol{D}(\boldsymbol{z}_0) + \text{softplus}(\hat{t})$ ▷ Convert relative depth to metric scale
    Compute $\mathcal{L}_{geo}$ following Eq. 14
    $\hat{s} \leftarrow \hat{s} - \lambda_{\hat{s}} \nabla_{\hat{s}} \mathcal{L}_{geo}$ ▷ Gradient update for $\hat{s}$
    $\hat{t} \leftarrow \hat{t} - \lambda_{\hat{t}} \nabla_{\hat{t}} \mathcal{L}_{geo}$ ▷ Gradient update for $\hat{t}$
    $\boldsymbol{z}_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\hat{\boldsymbol{z}}_0 + \sqrt{1-\bar{\alpha}_{t-1}}\boldsymbol{s}_\theta(\boldsymbol{z}_t, t, \boldsymbol{y}_1) - \lambda \nabla_{\boldsymbol{z}_t} \mathcal{L}_{geo}$ ▷ Perform DDIM step with geometric guidance
  end for
  Output: Estimated metric depth map $\tilde{\boldsymbol{x}}_0$

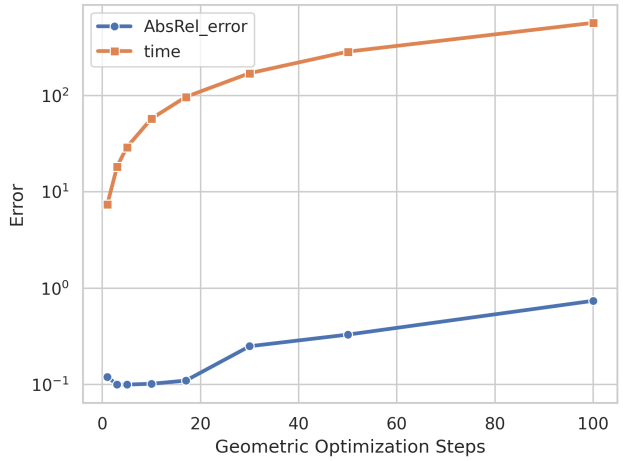

Figure 11. We increase the number of times to update inside one sampling step. We observe that it not only does not improve result, but also very time consuming to run.

to minimize the geometric loss, our experiments in Figure 11 demonstrate that only a limited number of gradient steps are beneficial. Consequently, we implement a single gradient update per sampling step in this work. Empirical observations indicate that increasing the number of gradient updates not only fails to improve performance but also significantly increases computational time.

Inference time. Inference time for a single sample using our approach is approximately 7 seconds on an RTX A6000 GPU with images of 768 pixels in dimension. This measurement excludes data preparation time, which varies across datasets.

Run time comparison. Our method requires test-time optimization but maintains computational efficiency comparable to the baseline Marigold [22], adding only seconds per image to processing time. This efficiency stems from our implementation of single-step gradient updates with minimal learnable parameters. Additionally, our depth warping-based rendering technique is both fast and fully differentiable. Consequently, despite achieving superior results, our approach does not significantly increase computational overhead compared to Marigold.

10. Dataset details

Training data. Our proposed method is a test time optimization-based, so we do not require any training sample. For details about training dataset of our baseline method, we refer reader to Marigold [22].

Evaluation data. We evaluate our proposed approach on four distinct datasets. The KITTI-2015 dataset comprises 200 stereo pairs depicting outdoor scenes. The Middlebury dataset contains 15 stereo pairs predominantly featuring indoor environments. The Booster dataset includes 228 stereo pairs with challenging non-Lambertian surfaces. For the Tanks and Temples dataset, we randomly sampled 116 image pairs from a multi-view dataset spanning four scenes.

11. Limitation

Since our method is based on diffusion sampling process, it is not suitable for real time application. Additionally, since we employ depth warping as a rendering technique and utilize photometric loss as an optimization objective, our approach exhibits sensitivity to significant illumination variations between stereo images. Potential solutions include applying color correction prior to image rendering or implementing left-right consistency as described in Section 7.2. We defer these improvements to future work.

12. Additional qualitative results

Additional qualitative results are presented in Figure 12, Figure 13, Figure 14.

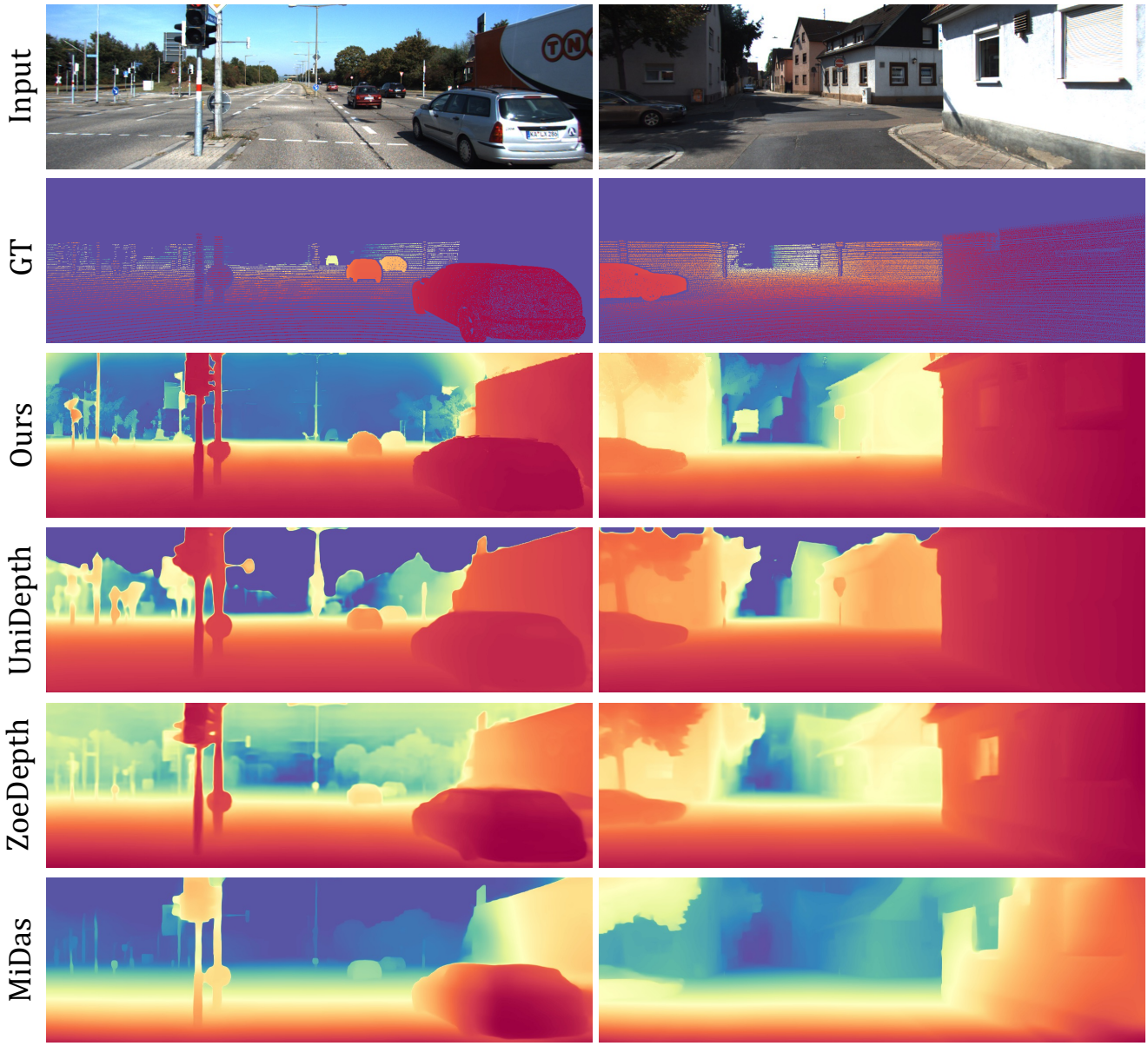

Figure 12. **Additional qualitative results.** Our method demonstrates superior depth quality in metric depth estimation, particularly in high-depth-range scenarios such as those found in the KITTI-2015 dataset [29], outperforming competing approaches.

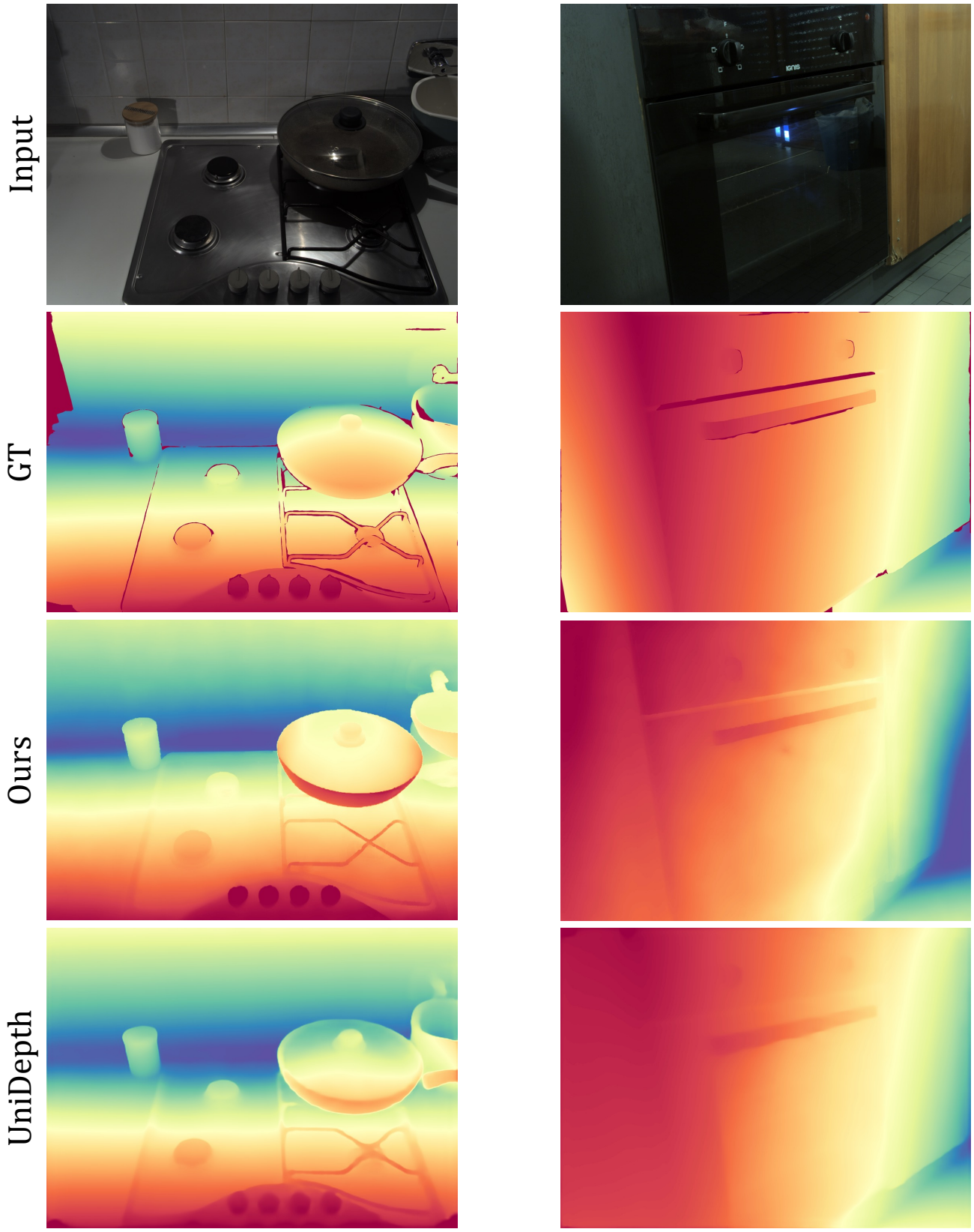

Figure 13. Additional qualitative results on Booster dataset [33]

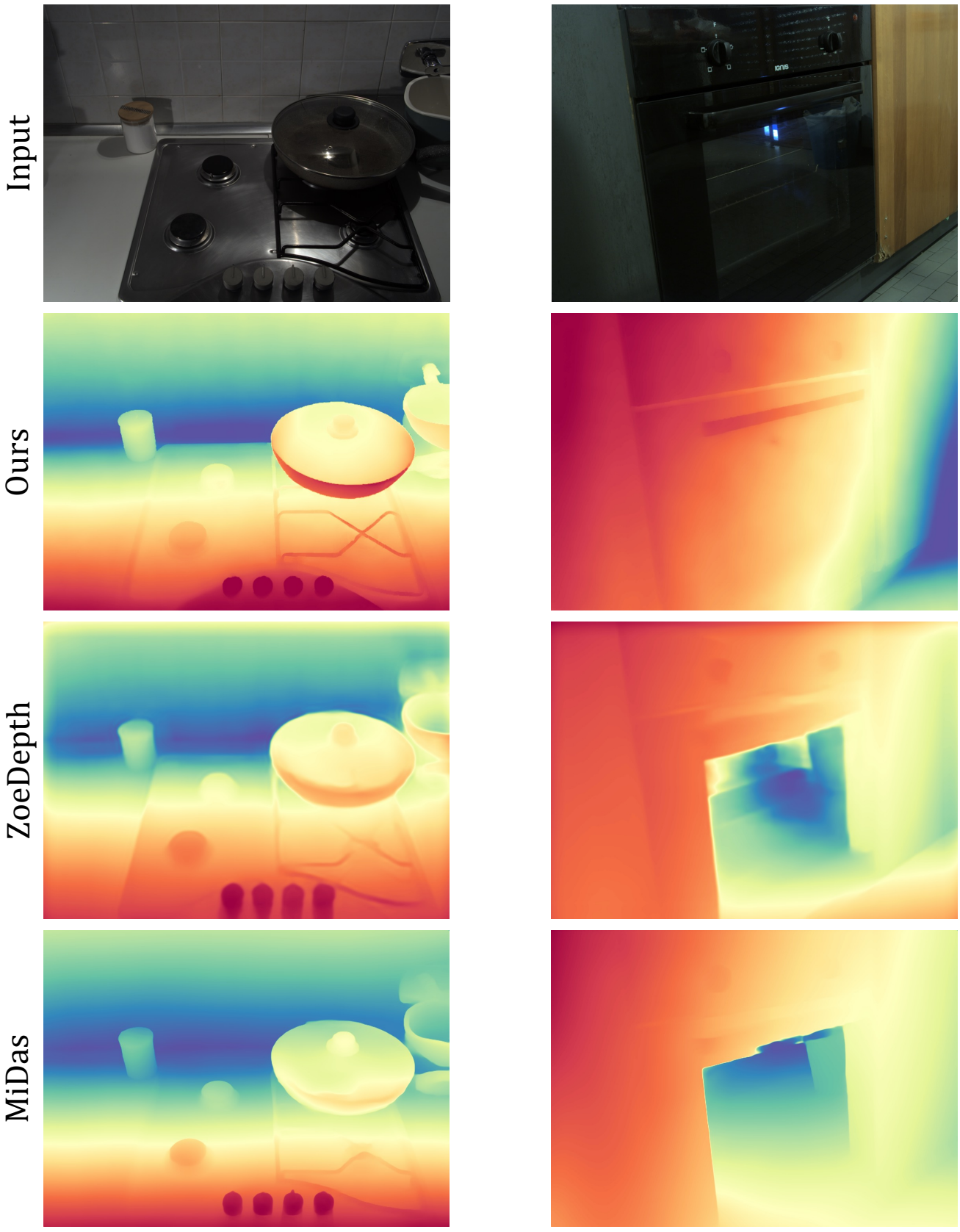

Figure 14. Additional qualitative results on Booster dataset [33]