CGI2022 manuscript No.
(will be inserted by the editor)

3D Terrain Estimation from a Single Landscape Image

Haruka Takahashi · Yoshihiro Kanamori · Yuki Endo

Abstract This paper presents the first technique to estimate a 3D terrain model from a single landscape image. Although monocular depth estimation also offers single-image 3D reconstruction, it assigns depth only to pixels visible in the input image, resulting in an incomplete 3D terrain output. Our method generates a complete 3D terrain model as a textured height map via a three-stage framework using deep neural networks. First, to exploit the performance of pixel-aligned estimation, we estimate terrain's per-pixel depth and color free from shadows or lights in the perspective view. Second, we triangulate the RGB-D data generated in the first stage and rasterize the triangular mesh from the top view to obtain an incomplete textured height map. Finally, we inpaint the depth and color in the missing regions. Because there are many possible ways to complete the missing regions, we synthesize diverse shapes and textures during inpainting using a variational autoencoder. Qualitative and quantitative experiments reveal that our method outperforms existing methods applying a direct perspective-to-top view transform as image-to-image translation.

Keywords Deep Learning · 3D reconstruction · Terrain

H. Takahashi
E-mail: harukaoceansouth@gmail.com
University of Tsukuba

Y. Kanamori
E-mail: kanamori@cs.tsukuba.ac.jp
University of Tsukuba

Y. Endo
E-mail: endo@cs.tsukuba.ac.jp
University of Tsukuba

1 Introduction

3D terrain models are widely used in film production and video games. Such terrain models are typically created by professional artists, but what if terrain models can be automatically generated from single landscape photos? We can enjoy 3D views of sightseeing spots in VR/AR applications. Single-view 3D reconstruction is possible with existing techniques of monocular depth estimation. However, they can only estimate the depth of the visible region in the input image, and hidden or occluded parts are left missing, resulting in an incomplete 3D model.

In this paper, we propose the first technique to estimate a complete 3D terrain model as a textured height map from a single landscape image. We face the following two challenges. First, we have to reconstruct the shape and texture of the regions that are occluded in the input image. Second, we should remove the shadows and shading from the 3D model's texture.

We overcome these challenges with a three-stage framework using convolutional neural networks (CNNs) via supervised learning. 1) We first employ monocular depth and texture estimation from the input image using CNNs, to exploit the advantage of pixel-aligned estimation. The texture is inferred such that our GAN-based approach removes the shadows and shading in the input image. The resulting RGB-D data is in the perspective view with missing regions, and thus we transform it to the top view, followed by inpainting. Namely, 2) we triangulate the RGB-D data and rasterize the triangular mesh in the top view via orthogonal projection to obtain incomplete height/texture maps. 3) We then inpaint the height and texture maps of the incomplete model using another CNN. Note that there are many possible ways to complete the missing regions. We thus

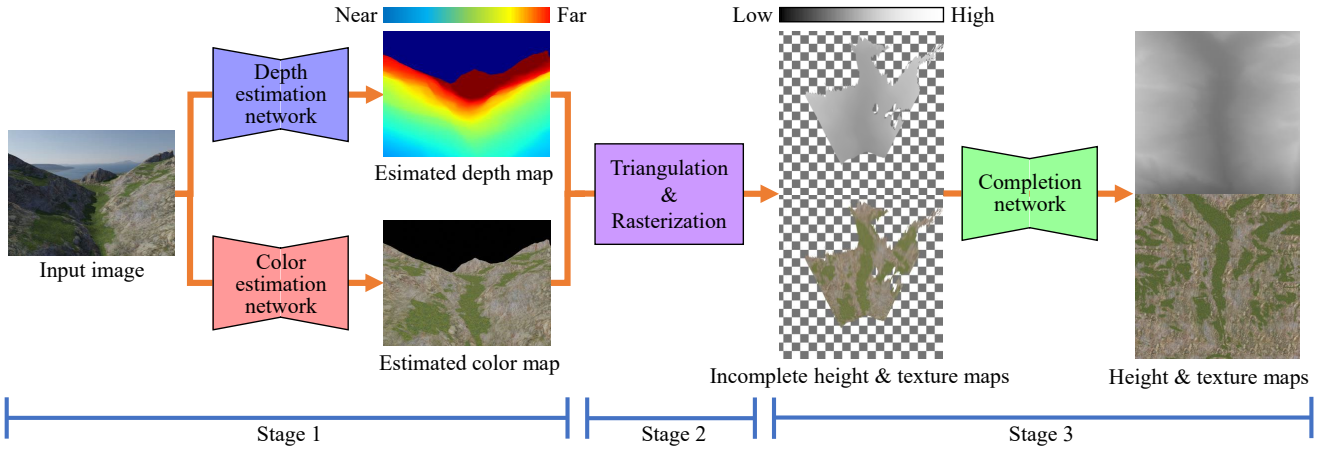

Fig. 1 Overview of our three-stage framework for single-image 3D terrain estimation. Stage 1 estimates the depth/color maps for visible pixels in the input image. Stage 2 then constructs a triangular mesh and rasterizes it in the top view via orthogonal projection. Stage 3 finally completes the missing regions to output complete height/texture maps.

synthesize diverse shapes and textures during inpainting using a *variational autoencoder* (VAE) [6]. For our supervised training, there are no public datasets of textured height maps representing terrains, to the best of our knowledge. We thus create a new dataset with measured height maps and synthetic textures.

To verify the effectiveness of our three-stage estimation, we compare our method qualitatively and quantitatively with existing methods for directly transforming an input image to another view and for image-to-image translation. We demonstrate that our method can estimate more plausible terrain 3D models.

2 Related Work

2.1 Terrain modeling and editing

Methods for modeling and editing 3D terrain models have a long history of research. Procedural methods [8–10] have been commonly used to form terrain geometry due to its fractal nature. They are advantageous in that they can define terrain geometry by simple recursive rules to randomly displace the terrain surface at relatively low computational cost. Simulation-based methods [7,10,11,16] can generate physically-plausible terrain models by simulating, e.g., the erosive effects of water flow. Sketch-based methods allow users to specify terrain geometry directly and intuitively with sketches from, e.g., the top view [24] or perspective view [18]. Our method differs from these methods in that our input is a single landscape image.

2.2 Monocular depth estimation

Monocular depth estimation has been actively studied in computer vision and can be applied to infer landscape geometry from a single image. Recent methods employ deep neural networks to achieve incredible performance [2,4,13,14,19,3,21]. However, these methods only handle visible regions in the input image and do not consider hidden or occluded parts of the terrain geometry.

2.3 View translation

Our method can be regarded as a multi-step view translation from the input view to another, for which there are several prior studies. For example, GAN-based methods [15,17] can directly translate aerial views to near-ground perspective views (or vice versa), which can be viewed as an extension of image-to-image translation. While they originally target RGB images as input, we can extend them to handle RGB-D data. However, view-transformed outputs tend to be inaccurate because they do not explicitly distinguish between visible and non-visible regions in the input image, unlike ours. See Section 4 for a comparison.

2.4 Other methods for terrain using deep learning

Besides the above-mentioned methods, recent techniques for targetting 3D terrain adopt deep learning. For example, there are methods for real-time terrain modeling from top-view sketches [5] and terrain super-resolution [1, 23]. To the best of our knowledge, there is no other

method for estimating a 3D terrain model from a single landscape image like ours.

3 Methodology

3.1 Overview

Figure 1 illustrates the overview of our three-stage framework to obtain a textured height map from the input landscape image. Stage 1 estimates the depth and color of the visible regions in the input image using respective networks. For the color estimation network, we employ GAN to remove shadows and shading due to the sunlight. We refer to the shadow- and shading-free color image extracted from the input image as a "*color map*." The resultant RGB-D map (i.e., the color and depth maps) lies in the same perspective view as the input image. Stage 2 triangulates the RGB-D map and rasterizes the triangular mesh from the top view via orthogonal projection to obtain incomplete height/texture maps. Finally, Stage 3 completes the missing regions of the rasterized depth and color maps simultaneously using another network. Because there are many possibilities for the reconstruction of the missing regions, we integrate a VAE to generate various shapes and textures for the missing regions. We train each network separately in a supervised manner.

We create a synthetic dataset for supervised learning. To simplify the problem, we restrict our target scenes to sunny landscapes containing mountains without water surfaces such as rivers and lakes. We also create binary masks to exclude sky regions for the training of the depth and color estimation in Stage 1. Binary masks for the test phase are created manually or using a commercial service[1]. Also, for simplification, we assume a calibrated camera for preparing both training and test data. We elaborate the dataset in Section 3.6.

Hereafter, we explain the components of our three-stage framework.

3.2 Stage 1: Depth estimation

While we can use any network architecture for monocular depth estimation in the first stage, we adopt one of the state-of-the-art architectures, *SARPN* [3]. It accounts for multi-scale feature information for handling the global scene structure with the low-resolution depth map and the detailed local shapes with the high-resolution depth map.

[1] `https://remove.bg/`

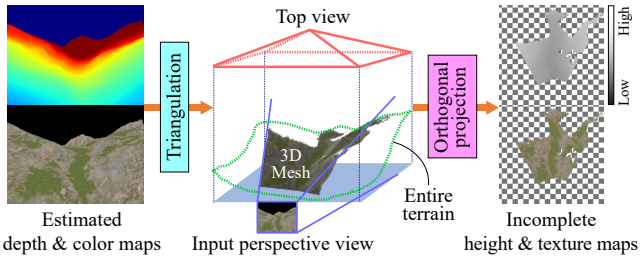

Fig. 2 Triangulation and rasterization for obtaining incomplete height and texture maps. We construct a tringular mesh from the estimated depth and color maps, and rasterize it from the top view via orthogonal projection.

We change the loss function of SARPN to improve the accuracy. Originally, SARPN uses the following loss function for training:

$$\mathcal{L} = \mathcal{L}_{depth} + \mathcal{L}_{grad} + \mathcal{L}_{normal}, \tag{1}$$

where $\mathcal{L}_{depth}$, $\mathcal{L}_{grad}$, and $\mathcal{L}_{normal}$ are loss functions for the inferred depth map, its gradient, and normal map, respectively. SARPN was proposed to infer indoor depth, and we found that the loss function of Eq. (1) is not suited well for landscape depth, where the depth range in our landscape dataset (from 100 to 4,500 meters) is much more significant than that of indoor scenes (from 0 to 10 meters). Specifically, $\mathcal{L}_{depth}$ is sensitive to outliers whose depth values are significantly large. We thus replace $\mathcal{L}_{depth}$ with the outlier-tolerant loss $\mathcal{L}_{ssitrim}$, used by Ranftl et al. [14]. Loss function $\mathcal{L}_{ssitrim}$ is defined as follows. First, with pixel i's depth z_i in estimated depth map Z, function q is defined as follows:

$$q(z_i, Z) = \frac{z_i - \text{median}(Z)}{\frac{1}{N}\sum_{i=1}^{N} |z_i - \text{median}(Z)|}, \tag{2}$$

where N is the number of pixels in Z and median(Z) yields the median value in Z. With function q, we define:

$$\mathcal{L}_{ssitrim} = \frac{1}{2N} \sum_{m \in U} \left| q(z_m, Z) - q(\hat{z}_m, \hat{Z}) \right|, \tag{3}$$

where $\hat{}$ denotes the corresponding ground truth, and U is a subset of pixels obtained by removing the top 20% of erroneous pixels. We also omit $\mathcal{L}_{normal}$ because it deteriorated the accuracy. In summary, we use the following loss function to train our depth estimation network:

$$\mathcal{L} = \mathcal{L}_{ssitrim} + \mathcal{L}_{grad}. \tag{4}$$

See Section 4.6 for the ablation study on loss functions. Note that, as explained in Section 3.1, sky regions are excluded in the loss function using binary masks.

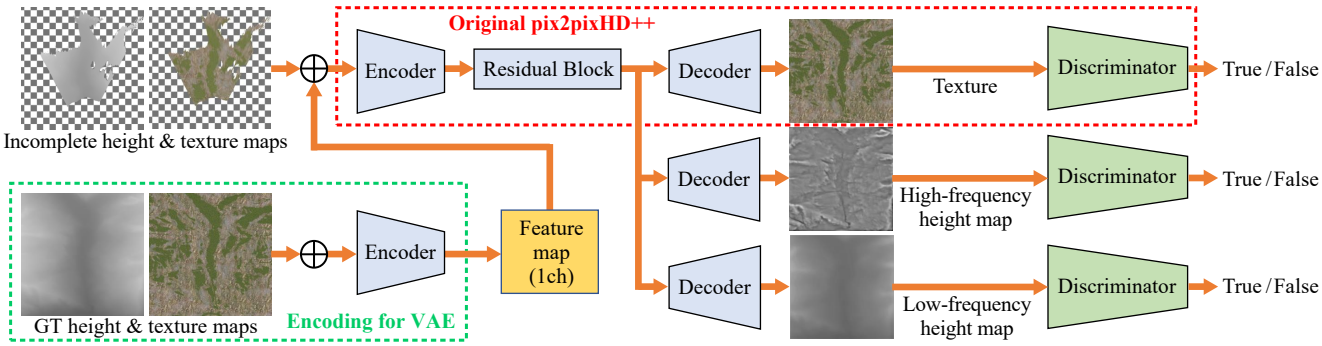

Fig. 3 Training overview for our height/texture completion network. Based on the network architecture of pix2pixHD++ [12], we integrate a VAE to diversify the completed height/texture of missing regions and extend to a multi-task network with three decoders for high-/low-frequency height and texture maps. $\oplus$ denotes channel-wise concatenation.

3.3 Stage 1: Color estimation

To estimate the color map from the input image, we adopt the network architecture of *SPADE* [12]. SPADE was originally proposed for semantic image synthesis, specialized image-to-image translation from a semantic mask to a photo-realistic image. The SPADE architecture injects the spatial information specified by the semantic input mask repeatedly at multi-scale. We adopt this architecture because we expect it as effective in our image-to-image translation based on conditional GAN.

For training, we used the same loss functions as SPADE's original setting; adversarial loss $\mathcal{L}_{GAN}$, feature matching loss $\mathcal{L}_{FM}$, and perceptual loss $\mathcal{L}_{vgg}$. Similar to depth estimation in Stage 1, we exclude sky regions in the input image from the loss calculation.

3.4 Stage 2: Rasterizing imcomplete RGB-D map

We transform the RGB-D map (i.e., color and depth maps) to a triangular mesh and rasterize it from the top view via orthogonal projection to obtain an incomplete textured height map (Fig. 2). For triangulation, we regard each pixel in the RGB-D map as a colored vertex and connect adjacent vertices[2] while omitting elongated triangles having large depth differences with a threshold of 100.

For rasterization, we require camera parameters of the input perspective view. The intrinsic and extrinsic camera parameters for training are provided from our dataset (Section 3.6). In the test phase, we assume that the intrinsic parameters are the same as those of our dataset, and, for the extrinsic parameters, the camera locates at the height of 1,000 meters, looking horizontally. We implemented rasterization with OpenGL and set the output resolution as 1024×1024.

[2] Specifically, we connect each pixel (i.e., a colored vertex) with its lower and right neighbors to define a triangle.

3.5 Stage 3: Completion of missing regions

When completing the incomplete textured height map, we account for the following two points: 1) There are many possibilities for the completed geometry/texture of the missing parts, and 2) the geometry and texture should correlate with each other; for example, a grass texture pattern should not appear on steep mountain slopes. For 1), we integrate a VAE to diversify the completed regions. For 2), we learn to complete geometry and texture simultaneously using a multi-task network. Furthermore, we separately learn the high- and low-frequency components of height maps to capture both global structures and local details accurately.

Fig. 3 illustrates the training procedure of our completion network. During training, the VAE's encoder (lower-left green dashed box in Fig. 3) encodes the concatenated ground-truth (GT) height/texture maps and yields a feature vector of channel size $(H \times W)$, where H and W are the height and width of GT height/texture maps. We obtain a 1-channel feature map of size $H \times W$ by reshaping the $(H \times W)$-channel vector. We then concatenate the resultant feature map and the incomplete height/texture maps channel-wise. From the concatenated input, our multi-task network outputs high-/low-frequency height and texture maps simultaneously. These three outputs are then fed to the respective discriminators. In the test phase, the VAE's encoder part is replaced by random sampling from the standard normal distribution.

For the network architecture, we base the architecture of *pix2pixHD++* [12] (top red dashed box in Fig. 3) because it supports high-resolution outputs, and modify it to support a VAE and multi-task outputs with

three decoders. In the decoder, the transposed convolution yields grid-like artifacts, so we replace it with upsampling followed by convolution. Regarding the loss functions, we used the same loss functions as those for the color estimation network (Section 3.3) and a KL-Divergence loss for the VAE.

3.6 Terrain landscape dataset

We created our dataset with measured geometry and synthetic texture using Blender. For the geometry, we used measured terrain height maps in *SRTM*[3]. For the texture, we generated ten different materials using a Blender add-on[4] and randomly assigned them. We first randomly cut out the textured terrains into 7,000-meter squares and, for each squared terrain, set up 36 perspective cameras looking at the center, equiangularly on a circumference with a radius of about 2,500 meters. The cameras' height is set to 75% of the maximum terrain height[5]. The cameras' pitch and roll angles were randomly set in the ranges of [-13, 0] (i.e., from looking-down to horizontal angles) and [-3, 3] degrees, respectively. The cameras' horizontal field-of-view angle is 60 degrees. For each camera, we rendered a shaded color image for input, GT color/depth maps, and a binary mask to exclude the sky, setting the maximum depth limit as 5,000 meters. We also extracted corresponding GT height and texture maps of 5,000-meter squares in front of each camera. We discard the data if the visible terrain area is less than 20% of the square area. For lighting, we set a sunlight and an environment light randomly selected from five 8k HDRIs obtained from *Poly Haven*[6].

Fig. 4 shows examples in our dataset. Our dataset has 11,440 views in total. The numbers for training, validation, and test samples are 10,000, 720, and 720, respectively. The image resolutions of depth/color maps and GT height/texture maps are 640×480 and 1024×1024, respectively.

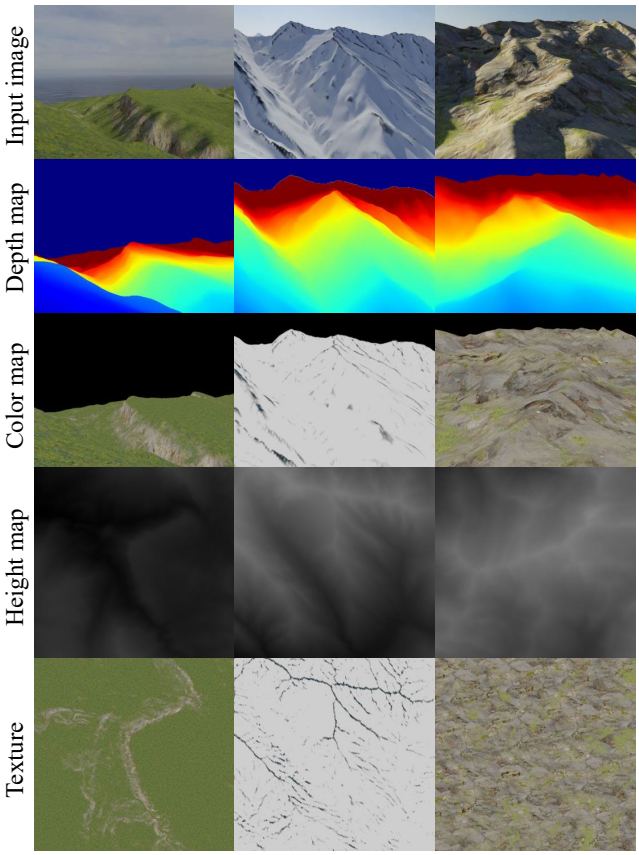

Fig. 4 Sample images from our terrain landscape datasets. From top to bottom, the terrain landscape images, depth maps, color maps, height maps, and textures are shown.

4 Experiments

4.1 Implementation details

We used Python and PyTorch to implement our method. For training, we assigned each network with one GPU from our heterogeneous GPU clusters, accounting for required GPU memories. Namely, we used Quadro RTX 6000 for the depth/color estimation networks and Quadro RTX 8000 for the completion network. We determined the training epochs using the validation data as 24 for the depth estimation network and 50 for the other networks. The training took about one day for each of the depth/color estimation networks and about two to three weeks for the completion network. For optimization, we used Adam for the training of all networks. For the depth estimation network, we set the momentum term as $\beta = (0.5, 0.999)$, the learning rate as 0.0001, the decay term as 0.00001. For the other networks, the momentum term was $\beta = (0, 0.999)$, the respective learning rates of the generators and discriminators were 0.0001 and 0.0004, and the decay term was 0. We scheduled the learning rate adjustment as follows. For the depth estimation network, we decayed the learning rate by 10% every 5 epochs. For the other networks, we decayed the learning rate linearly from 25 and terminated at 50 epochs with zero learning rate. The batch sizes were 4 for the depth/color estimation CNNs and 1 for the completion network.

[3] `http://viewfinderpanoramas.org/dem3.html`

[4] `https://www.blendermarket.com/products/true-terrain-`

[5] In case the cameras are buried in the ground, they are moved to the ground surface plus a 25-meter offset.

[6] `https://polyhaven.com/`

4.2 Experimental settings

Evaluation metrics. We used the following metrics for quantitative evaluation. For depth and height maps, we used absolute and relative errors. We also calculated the percentage of pixels for which the ratio δ between the estimated and GT values is less than a threshold. As thresholds, we used $\{1.25, 1.25^2, 1.25^3\}$, following existing depth estimation studies [4,13]. For evaluating textures, we used FID to measure the GAN output quality and LPIPS [22] as a perceptual evaluation measure.

Compared methods. We compared our method with SelectionGAN [17] and pix2pixHD++ [12]. SelectionGAN mutually translates aerial photographs and street views with semantic masks. We trained SelectionGAN and pix2pixHD++ from scratch using our dataset because their original tasks are different from ours. For each of SelectionGAN and pix2pixHD++, which originally output RGB images, we trained two respective models to generate height and texture maps directly from input images. We replaced the transposed convolution, as done in our network. Note that SelectionGAN requires a semantic mask as input corresponding to the output layout, for which we instead fed a three-channel noise image because our dataset does not contain such data. For pix2pixHD++, we resized the input images to 1024×1024, which is the input size of the network. Both SelectionGAN and pix2pixHD++ were trained using Adam up to 50 epochs with the same learning rate decay as our completion network. Adam's hyperparameters for pix2pixHD++ are the same as our completion network whereas those for SelectionGAN are the default values used in the original paper [17]. Our completion network yields multimodal outputs, so we generated 10 output candidates with random vectors for each input and selected the output with the best score[7]. We separately validated the faithfulness of 1) entire terrain outputs and 2) regions visible from the input viewpoints. For 2), we masked out the invisible regions from the output height and texture maps for calculating FID and LPIPS.

4.3 Qualitative comparisons

Fig. 5 shows a qualitative comparison of estimated terrains rendered from input views (upper rows) and overall views (lower rows). The overall views' orange points and dashed lines indicate the viewpoints and the field of views in the input images, respectively. We can see that our method reproduces the most plausible shapes and textures, particularly from the input views. This result indicates that our three-stage view translation is superior to the direct view translation approaches. However, our method sometimes yields unnatural repetitive patterns in estimated textures (Fig. 6).

[7] We ranked 10 output candidates for each metric, and counted the number of "being ranked as the top" for each candidate. We selected the candidate with the most top counts as the output.

4.4 Quantitative comparisons

Table 1 summarizes the quantitative comparison result. Regarding height maps, our method was the best for both the input and overall views. As for textures, our method was inferior to pix2pixHD++ for the overall views, probably due to the repetitive texture patterns in our method. In the input views, our method yielded the best quality textures. These results indicate that our method yields more plausible terrains than the compared methods, particularly in the input views.

4.5 Our multimodal terrain outputs

Fig. 7 shows our multimodal terrain outputs yielded by our completion network. Similar to the qualitative comparison, we show both the input and overall views. We can see that our terrains' shape and texture are consistently faithful to the ground truth in the input views, whereas they have rich variations in the regions that are invisible from the input views (highlighted with the red boxes).

4.6 Ablation study

We conducted ablation studies to validate 1) the loss functions for the depth estimation in Stage 1 and 2) the frequency decomposition in our completion network in Stage 2.

Validation of depth estimation losses. For training our depth estimation network, we tried the following four combinations of the loss functions explained in Section 3.2:

- $\mathcal{L}_{depth} + \mathcal{L}_{grad}$,
- $\mathcal{L}_{depth} + \mathcal{L}_{grad} + \mathcal{L}_{normal}$ (i.e., SARPN's original),
- $\mathcal{L}_{ssitrim} + \mathcal{L}_{grad}$, and
- $\mathcal{L}_{ssitrim} + \mathcal{L}_{grad} + \mathcal{L}_{normal}$.

Table 2 summarizes the quantitative comparison. Without $\mathcal{L}_{normal}$, the estimation accuracy improved when $\mathcal{L}_{ssitrim}$ is used rather than $\mathcal{L}_{depth}$. With $\mathcal{L}_{normal}$, the estimation accuracy improved when $\mathcal{L}_{depth}$ was used

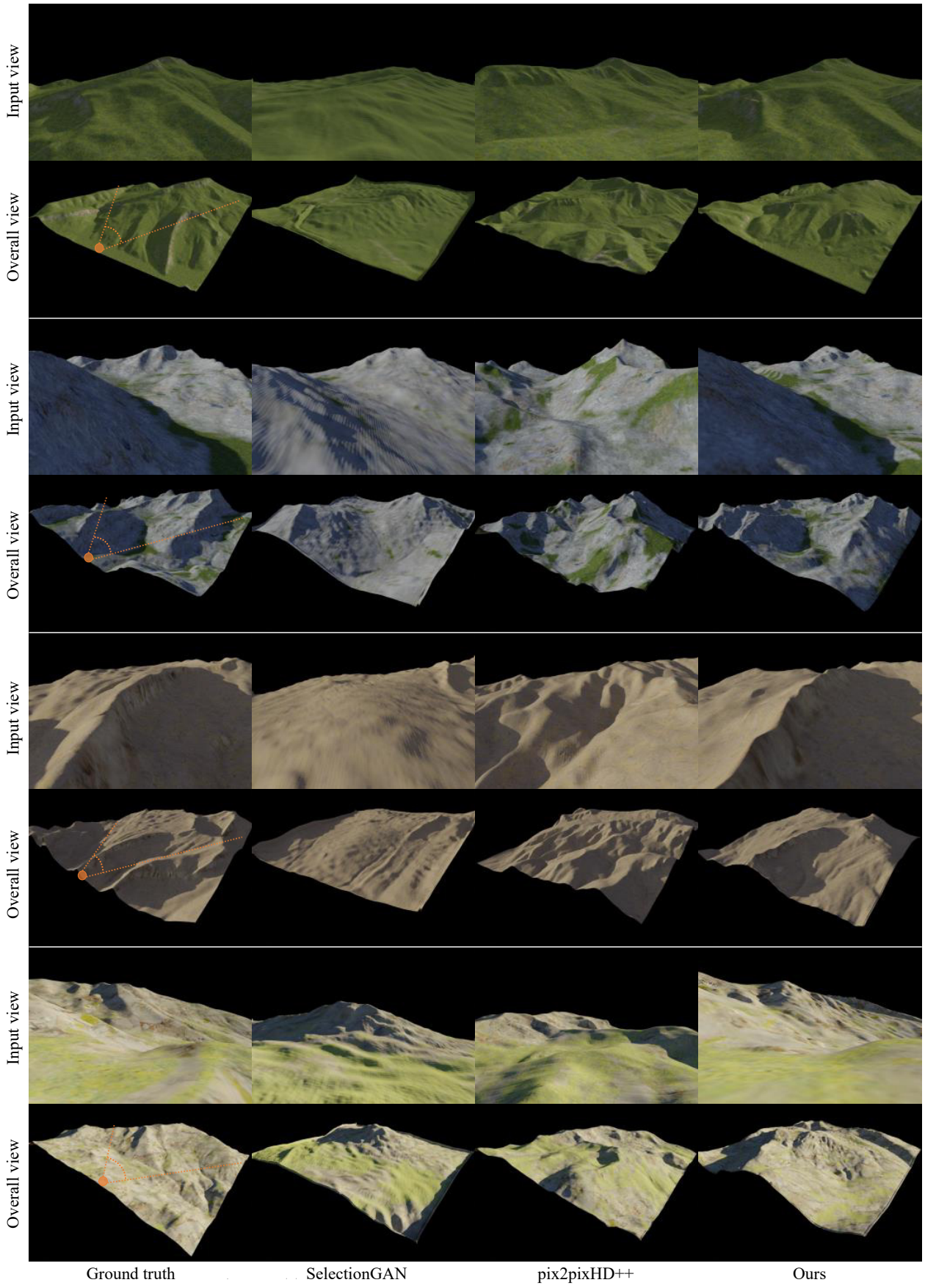

Fig. 5 Qualitative comparison of 3D terrain outputs rendered in the input views (upper rows) and overall views (lower rows). The orange points and the dashed lines indicate the viewpoints and the field of views in the input images, respectively.

Table 1 Quantitave comparison of our method and the other methods for both over and input views. The best score of each metric is highlighted by boldface.

Overall view	Height map					Texture	
	$\delta < 1.25 \uparrow$	$\delta < 1.25^2 \uparrow$	$\delta < 1.25^3 \uparrow$	MAE $\downarrow$	rel $\downarrow$	FID $\downarrow$	LPIPS $\downarrow$
SelectionGAN [17]	0.353	0.615	0.771	247	17.7	170	0.736
pix2pixHD++ [12]	0.354	0.616	0.774	237	9.46	**75.6**	0.379
Ours	**0.596**	**0.809**	**0.895**	**143**	**8.77**	82.2	**0.364**
Input view	Height map					Texture	
	$\delta < 1.25 \uparrow$	$\delta < 1.25^2 \uparrow$	$\delta < 1.25^3 \uparrow$	MAE $\downarrow$	rel $\downarrow$	FID $\downarrow$	LPIPS $\downarrow$
SelectionGAN [17]	0.372	0.642	0.793	237	1.36	77.1	0.0778
pix2pixHD++ [12]	0.386	0.650	0.799	226	0.718	28.0	0.0376
Ours	**0.765**	**0.909**	**0.952**	**88.4**	**0.618**	**24.1**	**0.0329**

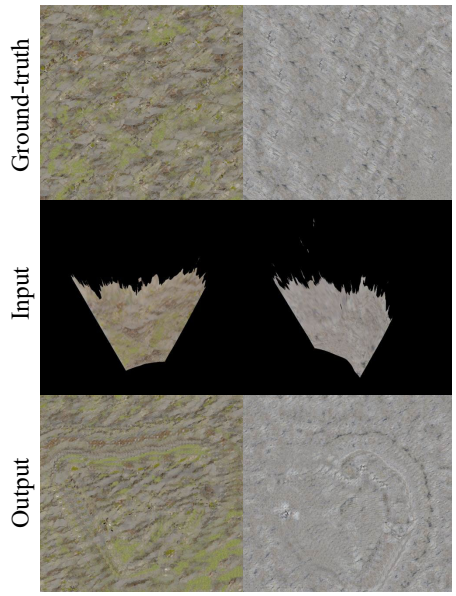

Fig. 6 Unnatural repetitive texture patterns sometimes appear in our completed textures.

but deteriorated when $\mathcal{L}_{ssitrim}$ was used. In summary, the best combination for all the evaluation metrics was $\mathcal{L}_{ssitrim} + \mathcal{L}_{grad}$.

Validation of frequency decomposition of height maps. For training our completion network (Section 3.5), we compared the results with and without the frequency decomposition of height maps. Table 3 shows the quantitative comparison result, indicating that the scores without frequency decomposition are consistently better than those with frequency decomposition. On the other hand, we found that rippled artifacts appear without frequency decomposition (Fig. 8). We thus adopted frequency decomposition for better visual quality.

4.7 Application to real photographs

We also applied our and other methods to real landscape photographs (Fig. 9), for which ground-truth 3D terrains are not available. In the first example of Fig. 9, our result looks more similar to the input image than the other methods. Theoretically, our method is advantageous over 3D visualizations of measured terrains (e.g., Google Earth) in that ours can capture the view at the moment of photo shooting, e.g., seasonal changes of landscapes. However, as is often the case with learning-based methods, our method heavily depends on the training data. Indeed, our estimated texture in the second example of Fig. 9 is quite different from the input image, although our estimated geometry is still better than those of the other methods. This defect comes from the limited texture patterns in our training dataset. Enriching our dataset for better generalizability is left as future work.

5 Limitations

Our method has the following limitations. First, as is often the case with learning-based approaches, our method cannot reproduce scenes that are largely different from those in our training dataset. In addition to a failure case with an inappropriate texture in Fig. 9, bottom, Fig. 10 shows another example where not only the texture but also the geometry are inferred inaccurately. Second, we have several assumptions, such as no water surfaces and a calibrated camera, for simplifying the problem, as explained in Section 3.1. These limitations can be overcome by, for example, enriching the dataset and inferring camera parameters as well. Also, because we represent terrains as textured height maps, our method cannot handle terrains with two or more ground surfaces per pixel in height maps, such as caves or overhangs. We will be able to support such complicated geometry by adopting implicit geometric representations [20].

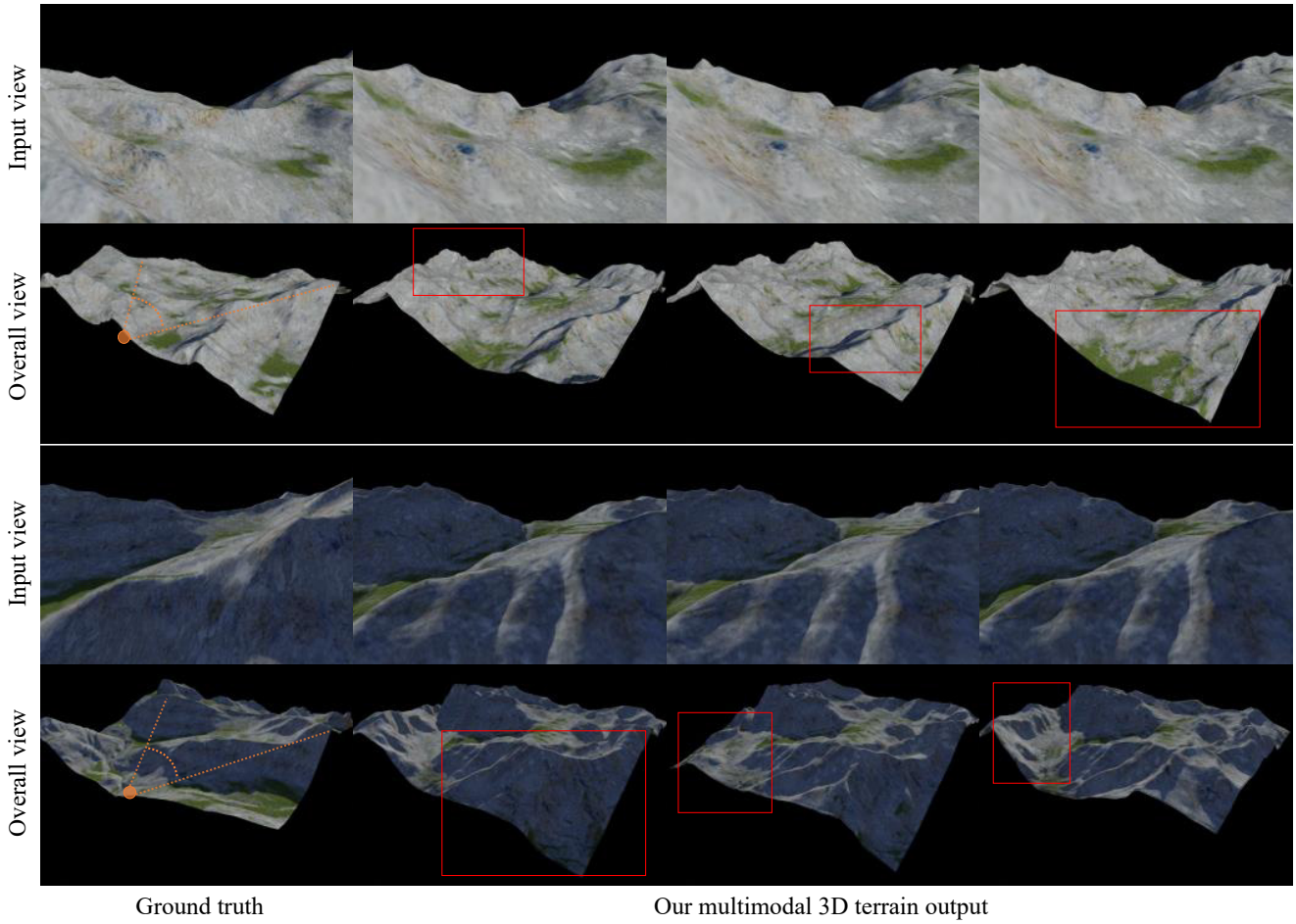

Fig. 7 Multimodal 3D terrain outputs of our completion network. Our method yields rich variations in geometry and texture for regions invisible from the input views (highlighted with the red boxes in the overall views) while retaining the input views.

Table 2 Quantitative comparison of the four combinations of loss functions (i.e., $\mathcal{L}_{depth}$, $\mathcal{L}_{grad}$, and $\mathcal{L}_{normal}$ from SARPN and $\mathcal{L}_{ssitrim}$ from MiDas) for the training our depth estimation network.

Combination of loss function	$\delta < 1.25 \uparrow$	$\delta < 1.25^2 \uparrow$	$\delta < 1.25^3 \uparrow$	MAE $\downarrow$	rel $\downarrow$
$\mathcal{L}_{depth} + \mathcal{L}_{grad}$	0.76	0.92	0.96	311.6	6.9×10^5
$\mathcal{L}_{depth} + \mathcal{L}_{grad} + \mathcal{L}_{normal}$	0.78	0.93	0.97	282.7	3.8×10^4
$\mathcal{L}_{ssitrim} + \mathcal{L}_{grad}$	**0.88**	**0.97**	**0.99**	**197.3**	**75**
$\mathcal{L}_{ssitrim} + \mathcal{L}_{grad} + \mathcal{L}_{normal}$	0.27	0.69	0.89	1064	83

Table 3 Quantitative comparison of results with and without frequency decomposition of height maps for the training of our multi-task completion network.

Overall view	Height map				
	$\delta < 1.25 \uparrow$	$\delta < 1.25^2 \uparrow$	$\delta < 1.25^3 \uparrow$	MAE $\downarrow$	rel $\downarrow$
Without frequency decomposition	**0.611**	**0.821**	**0.900**	**133**	**4.69**
With frequency decomposition	0.596	0.809	0.895	143	8.77
Input view	**Height map**				
	$\delta < 1.25 \uparrow$	$\delta < 1.25^2 \uparrow$	$\delta < 1.25^3 \uparrow$	MAE $\downarrow$	rel $\downarrow$
Without frequency decomposition	**0.780**	**0.918**	**0.956**	**82.2**	**0.253**
With frequency decomposition	0.765	0.909	0.952	88.4	0.618

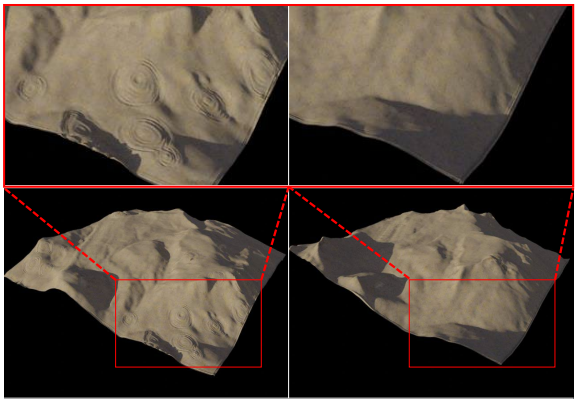

Fig. 8 Comparison of results without (left) and with (right) frequency decomposition of height maps. Rippled artifacts appear without frequency decomposition.

6 Conclusion

In this paper, we have proposed the first technique to estimate a 3D terrain model from a single landscape image. We achieved this via the following three stages. First, from the input image, we estimated depth and color maps using deep neural networks. We then triangulated the resultant RGB-D map (i.e., color and depth maps) and rasterized the triangular mesh from the top view via orthogonal projection. Finally, we completed the missing regions using a multi-task network, in which we also explored output variations using a VAE for missing regions and frequency decomposition of height maps for better visual quality. We created a novel dataset of 3D terrains with measured height maps and synthetic textures. Qualitative and quantitative evaluations demonstrated that our method yields more plausible 3D terrain models than the compared methods, more faithful particularly in the input views.

Our future work will be devoted mainly to improving visual quality and accuracy, including exploration of better network architectures and end-to-end training frameworks, and entrichment of our terrain dataset using, e.g., *Google Earth Engine*, for better generalizability. We also would like to explore a self-supervised approach that does not rely on synthetic datasets for better generalization performance.

References

1. Argudo, O., Chica, A., Andújar, C.: Terrain super-resolution through aerial imagery and fully convolutional networks. Computer Graphics Forum **37**(2), 101–110 (2018)
2. Bhat, S.F., Alhashim, I., Wonka, P.: AdaBins: Depth Estimation using Adaptive Bins. arXiv:2011.14141 (2020). URL `https://arxiv.org/abs/2011.14141`
3. Chen, X., Chen, X., Zha, Z.J.: Structure-Aware Residual Pyramid Network for Monocular Depth Estimation. In: Proceedings of the International Joint Conferences on Artificial Intelligence, pp. 694–700 (2019)
4. Fu, H., Gong, M., Wang, C., Batmanghelich, K., Tao, D.: Deep Ordinal Regression Network for Monocular Depth Estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2018), pp. 2002–2011 (2018)
5. Guérin, É., Digne, J., Galin, E., Peytavie, A., Wolf, C., Benes, B., Martinez, B.: Interactive example-based terrain authoring with conditional generative adversarial networks. ACM Transactions on Graphics **36**(6), 228:1–228:13 (2017)
6. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: ICLR 2014 (2014)
7. Kristof, P., Benes, B., Krivánek, J., Stava, O.: Hydraulic erosion using smoothed particle hydrodynamics. Computer Graphics Forum **28**(2), 219–228 (2009)
8. Lewis, J.P.: Generalized stochastic subdivision. ACM Transactions on Graphics **6**(3), 167–190 (1987). DOI 10.1145/35068.35069. URL `https://doi.org/10.1145/35068.35069`
9. Miller, G.S.P.: The definition and rendering of terrain maps. In: D.C. Evans, R.J. Athay (eds.) Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1986, Dallas, Texas, USA, August 18-22, 1986, pp. 39–48. ACM (1986)
10. Musgrave, F.K., Kolb, C.E., Mace, R.S.: The synthesis and rendering of eroded fractal terrains. In: J.J. Thomas (ed.) Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1989, Boston, MA, USA, July 31 - August 4, 1989, pp. 41–50. ACM (1989)
11. Neidhold, B., Wacker, M., Deussen, O.: Interactive physically based fluid and erosion simulation. In: P. Poulin, E. Galin (eds.) Proceedings of the Eurographics Workshop on Natural Phenomena, NPH 2005, Dublin, Ireland, 2005, pp. 25–32. Eurographics Association (2005)
12. Park, T., Liu, M., Wang, T., Zhu, J.: Semantic Image Synthesis with Spatially-Adaptive Normalization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2019), pp. 2337–2346 (2019)
13. Ramamonjisoa, M., Lepetit, V.: SharpNet: Fast and Accurate Recovery of Occluding Contours in Monocular Depth Estimation. The IEEE International Conference on Computer Vision (ICCV) Workshops on 3D Reconstruction in the Wild (2019)
14. Ranftl, R., Lasinger, K., Hafner, D., Schindler, K., Koltun, V.: Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. IEEE Transactions on Pattern Analysis and Machine Intelligence pp. 1–1 (2020)
15. Regmi, K., Borji, A.: Cross-view Image Synthesis Using Conditional GANs. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2018), pp. 3501–3510 (2018)
16. Roudier, P., Peroche, B., Perrin, M.: Landscapes synthesis achieved through erosion and deposition process simulation. Computer Graphics Forum **12**(3), 375–383 (1993)
17. Tang, H., Xu, D., Sebe, N., Wang, Y., Corso, J.J.J., Yan, Y.: Multi-Channel Attention Selection GAN With Cascaded Semantic Guidance for Cross-View Image Translation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2019), pp. 2417–2426 (2019)

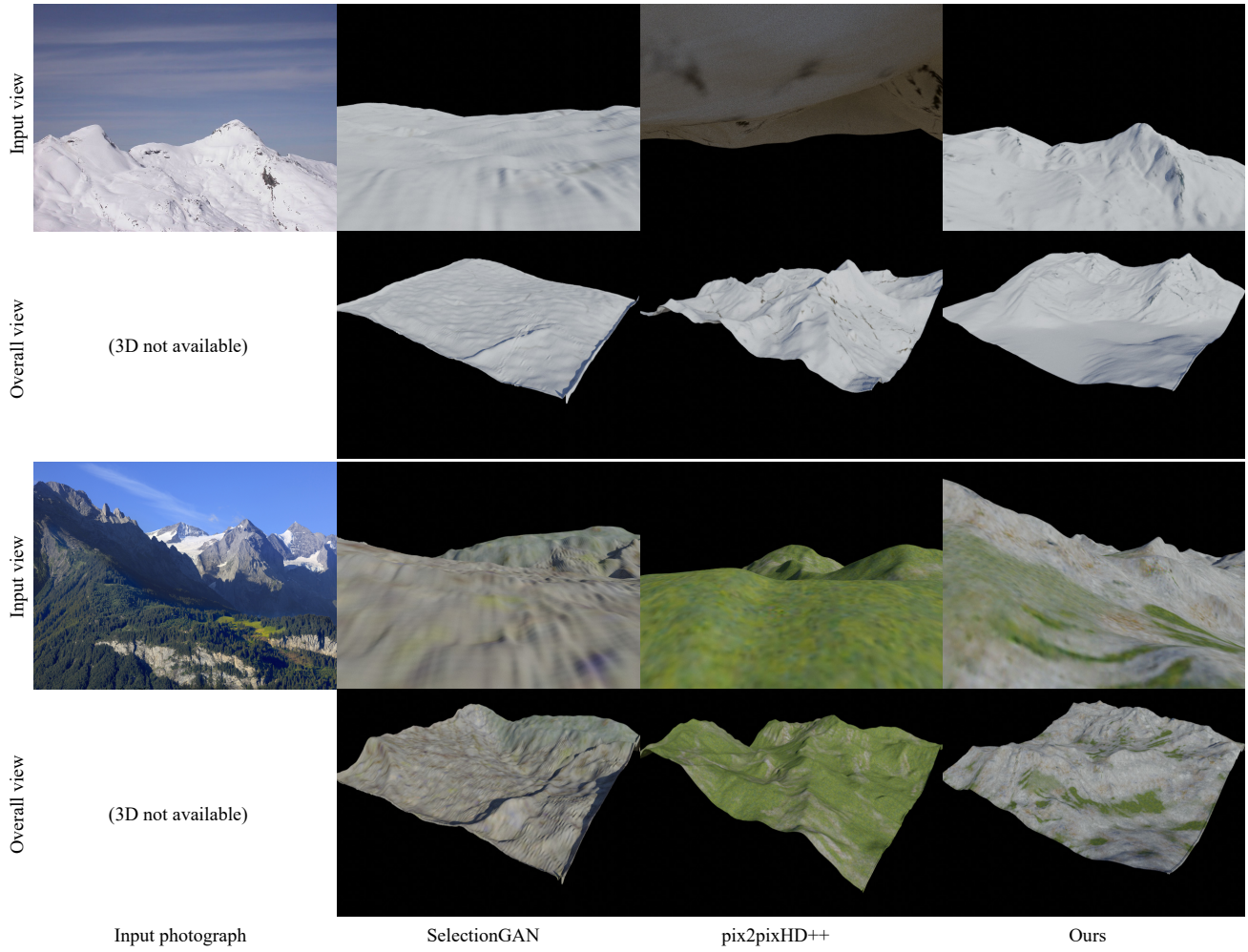

Fig. 9 Comparison with real photograph inputs. pix2pixHD++ failed to reproduce the input view in the first example, in which the camera is buried in the ground.

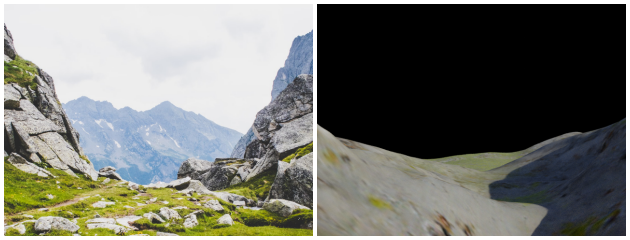

Fig. 10 Failure case where both texture and geometry are inferred inaccurately.

18. Tasse, F.P., Emilien, A., Cani, M., Hahmann, S., Bernhardt, A.: First person sketch-based terrain editing. In: P.G. Kry, A. Bunt (eds.) Graphics Interface 2014, GI '14, Montreal, QC, Canada, May 7-9, 2014, pp. 217–224. Canadian Information Processing Society / ACM (2014)
19. Wu, Y., Boominathan, V., Chen, H., Sankaranarayanan, A., Veeraraghavan, A.: PhaseCam3D — Learning Phase Masks for Passive Single View Depth Estimation. In: Proceedings of the IEEE International Conference on Computational Photography (ICCP 2019), pp. 1–12 (2019)
20. Xie, Y., Takikawa, T., Saito, S., Litany, O., Yan, S., Khan, N., Tombari, F., Tompkin, J., Sitzmann, V., Sridhar, S.: Neural fields in visual computing and beyond. Computer Graphics Forum (2022). DOI 10.1111/cgf.14505
21. Yin, W., Liu, Y., Shen, C., Yan, Y.: Enforcing Geometric Constraints of Virtual Normal for Depth Prediction. In: Proceedings of the IEEE Conference on Computer Vision (ICCV 2019), pp. 5683–5692 (2019)
22. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2018), pp. 586–595 (2018)
23. Zhao, Y., Liu, H., Borovikov, I., Beirami, A., Sanjabi, M., Zaman, K.A.: Multi-theme generative adversarial terrain amplification. ACM Transactions on Graphics **38**(6), 200:1–200:14 (2019)
24. Zhou, H., Sun, J., Turk, G., Rehg, J.M.: Terrain synthesis from digital elevation models. IEEE Transactions

on Visualization and Computer Graphics **13**(4), 834–848
(2007)